TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

COURSE FILE

Subject: WEB TECHNOLOGIES

Academic Year: 2022- 2023

Name of the Faculty: Mrs. A.Vijetha

Department : Information Technology

Branch & Year : IT III YEAR II SEMESTER

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Check List of the Course File

Department: IT **Date:**

Pre-Academic Session Review **Subject Code: 20CS6PC21**

Title of the Subject: Web Technologies.

S.No	Attributes	Yes/No
1	**Vision and Mission of institute**	
2	**Course handout& its contents**	
	a) Vision and Mission of department	
	b) PEOs of the program	
	c) Program Outcomes (POs)	
	d) Prerequisites	
	e) Course Outcomes (COs)	
	f) Detailed syllabus	
	g) Course Plan	
	h) Evaluation scheme	
3	CO-PO mapping	
4	Course material	
5	Teaching diary for the course	
6	Question bank prepared by faculty (unit wise)	
7	Quiz question bank	
8	Descriptive question bank for assignment	
9	Sets of copies of old question papers	
10	Analysis of student performance	
11	Answer book copies	

	a) Internal books	
	b) Assignment Copies	
	c) Laboratory records (if any)	
12	Whether remedial measures were taken by faculty members after completion of first module (with supporting documents)	
13	IQAC review report of teaching notes (pre + post)	

Name & Signatures IQAC members

1.

2.

3.

4.

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Teaching Notes Review Report

Department: IT **Date:**

Pre-Academic Session Review **Subject Code**: **20CS6PC21**

Title of the Subject: Web Technologies.

S. No	Observations	Excellent/Good/Fair	Suggestions/Remarks
1	Depth and Breadth of the subject to be covered		
2	Quality of the question bank		
	a) Quiz question bank		
	b) Descriptive assignment		
3	Degree of relevance to attainment of POs and PSOs by the course content		
4	Whether the course content is designed in view of bridging the gap for attainment to POs and PSOs with meaningful Course Outcomes (C.O.)		
5	Past result analysis for reference and identifying remedial measures to be carried out to attainment of improvement.		

Committee members of the Department

1.

2.

3.

4.

Members of the academic committee

1. Principal

2 Head of the Department

3. Subject Expert

4. Department IQAC Coordinator

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Teaching Notes Review Report

Department: IT **Date:**

Post-Academic Session Review **Subject Code: 20CS6PC21**

Title of the Subject: Web Technologies

S. No	Observations	Excellent/Good/Fair	Suggestions/Remarks
1	Depth and Breadth of the subject to be covered		
2	Quality of the question bank		
	a) Quiz question bank		
	b) Descriptive assignment		
3	Degree of relevance to attainment of POs and PSOs by the course content		
4	Whether the course content is designed in view of bridge the gap for attainment to POs and PSOs with meaningful Course Outcomes (C.O.)		
6	Past result analysis for reference and identifying remedial measures to be carried out to attainment of improvement.		

Committee members of the Department

1.

2.

3.

4.

Members of the academic committee

1. Principal

2 Head of the Department

3. Subject Expert

4. Department IQAC Coordinator

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

COURSE FILE

COURSE DESCRIPTION / COURSE INFORMATION SHEET

Name of the Dept: Information Technology

Course Title	**Web Technologies**			
Course Code		**Programme**	**Information Technology**	
Regulation	**R20**	**Year/Semester**	**III/II**	
Course Structure	**Lectures**	**Tutorials**	**Practical**	**Credits**
	2	**0**	**0**	**2**
Course Teacher	**Mrs. A.Vijetha**			
Email	**vijethait@tkrec.a.in**			
Phone No	**9441346327**			
No of Hours Allotted per Week	**Lectures**	**Tutorial**	**Practical**	
	2	**0**	**0**	

I. COURSE OVERVIEW:

1. Vision & Mission of the Institution

Vision	Imparting Knowledge and instilling skills to the aspiring students in the field of Engineering, Technology, Science and Management to face the emerging challenges of the society.
Mission	➢ Encouraging scholarly activities that transfer knowledge in the areas of Engineering, Technology, Science and Management. ➢ Ensuring students of all levels, well trained to meet the needs of education and their future endeavors. Inculcating human values and ethics into the education system for the all-round development of the students.

2. Course Handout

a) Vision & Mission of the Department	
Vision	The program aims at creating capable engineering professionals to meet the flourishing needs of the Industry and society in the field of Information Technology.
Mission	➢ Impart adequate employability skills to make the students industry ready with global standards. ➢ Inculcate ethical values and leadership qualities in addressing the societal needs using Information Technology.
b) Program Educational Objectives (PEOs)	**PEO1**. The graduates will be prepared to adopt emerging technologies for professional growth. **PEO2**. The graduates will be able to pursue research in upcoming

	technologies related to Information Technology with ethics. **PEO3**. The graduates will be able to apply their knowledge through lifelong learning to meet the challenges of the society.
c) Program Outcomes & Program Specific Outcomes (POs)& (PSOs)	**PO1. Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. **PO2. Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. **PO3. Design / development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. **PO6. The engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. **PO7. Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. **PO9. Individual and team work**: Function effectively as an individual, and

	as a member or leader in diverse teams, and in multidisciplinary settings. **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. **PSO1.**Students should be able to apply creativity in support of the design, simulation, implementation & oversight of more advanced technologies. **PSO2.** An ability to recognize the importance of professional developments by pursuing postgraduate studies (or) to participate & succeed in competitive examinations that offer challenging & rewarding careers.

Prerequisites	➢ Basic programming knowledge of C. ➢ Programming knowledge of JAVA.
e) Course Outcomes (COs)	**CO1**. Apply the concepts of PHP in creating web pages and connecting to database(My sql) **CO2.** Apply the concepts of XML for structuring the web pages. **CO3.** Make use of Servlets to create dynamic web pages in client-server architecture. **CO4.** Make use of JSP to develop interactive web pages. **CO5.** Apply the techniques of Java script in client side scripting.

f) Detailed Syllabus:

R20 B.TECH IT III YEAR

20CS6PC21: WEB TECHNOLOGIES

III Year B.Tech. IT II-Sem L T P C

2 0 0 2

COURSE OBJECTIVES

- To understand the technologies used in Web Programming.
- To know the importance of object-oriented aspects of Scripting.
- To understand creating database connectivity using JDBC.
- To learn the concepts of web-based application using sockets.

COURSE OUTCOMES: The student will able to

- Apply the concepts of PHP in creating web pages and connecting to database(My sql)
- Apply the concepts of XML for structurizing the web pages.
- Make use of Servlets to create dynamic web pages in client-server architecture.
- Make use of JSP to develop interactive web pages.
- Apply the techniques of Java script in client side scripting.

UNIT- I

HTML Common tags- List, Tables, images, forms, Frames; Cascading Style sheets**.** **Introduction to PHP:**Declaring variables, data types, arrays, strings, operators, expressions, control structures, functions, Readingdata from web form controls like text boxes, radio buttons, lists etc., Handling File Uploads. Connecting todatabase (MySQL as reference), executing simple queries, handling results, Handling sessions and cookies File

Handling in PHP: File operations like opening, closing, reading, writing, appending, deleting etc. on text andbinary files, listing directories.

UNIT- II

XML: Introduction to XML, Defining XML tags, their attributes and values, Document Type Definition, XMLSchemes, Document Object Model, XHTML Parsing XML Data – DOM and SAX Parsers in java.

UNIT - III

Introduction to Servlets: Common Gateway Interface (CGt), Life cycle of a Servlet, deploying a servlet, TheServlet API, Reading Servlet parameters, Reading Initialization

parameters, Handling Http Request &Responses, Using Cookies and Sessions, connecting to a database using JDBC.

UNIT - IV

Introduction to JSP: The Anatomy of a JSP Page, JSP Processing, Declarations, Directives, Expressions,Code Snippets, implicit objects, Using Beans in JSP Pages, Using Cookies and session for session tracking,connecting to database in JSP.

UNIT - V

Client-side Scripting: Introduction to Javascript, Javascript language – declaring variables, scope of variables,functions. Event handlers (onclick, onsubmit etc.), Document Object Model, Form validation.

TEXT BOOKS

1. Harvey Deitel, Abbey Deitel, Internet and World Wide Web: How To Program 5th Edition.
2. Herbert Schildt, Java - The Complete Reference, 7th Edition. Tata McGraw- Hill Edition.
3. Michael Morrison XML Unleashed Tech media SAMS.

REFERENCE BOOKS

1. John Pollock, Javascript - A Beginners Guide, 3rd Edition – Tata McGraw-Hill Edition.
2. Keyur Shah, Gateway to Java Programmer Sun Certification, Tata McGraw Hill, 2002.

g) Course Plan (Theory)

Department of INFORMATION TECHNOLOGY		Theory:	2
PROGRAM(UG):	INFORMATION TECHNOLOGY	Practical:	0
Course Code:	20CS6PC21	Credits:	2
Course Name:	WEB TECHNOLOGIES		
Regulation:	R20		
Class	**Section**	**Name of the Faculty**	
III Year - II Sem	IT	Mrs. A.Vijetha	

COURSE OUTCOMES:

After successful completion of the course, the student will be able to,

- Apply the concepts of PHP in creating web pages and connecting to database(My sql)
- Apply the concepts of XML for structurizing the web pages.
- Make use of Servlets to create dynamic web pages in client-server architecture.
- Make use of JSP to develop interactive web pages.
- Apply the techniques of Java script in client side scripting.

UNIT - I

Introduction to PHP: Declaring variables, data types, arrays, strings, operators, expressions, control structures, functions, Reading data from web form controls like text boxes, radio buttons, lists etc., Handling File Uploads, Connecting to database (MySQL as reference), executing simple queries, handling results, Handling sessions and cookies

File Handling in PHP: File operations like opening, closing, reading, writing, appending, deleting etc. on text and binary files, listing directories

Objectives: To introduce PHP language for server side scripting.

Outcome: Apply the concepts of PHP in creating web pages and connecting to database(My sql)

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

UNIT NO.	TOPIC NAME	BOOKS REFERENCED	PROPOSED NO. OF PERIODS	PROPOSED DATE	BLOOM'S TAXANOMY LEVEL	TEACHING AID
I	**HTML TAGS**					
1.1	Lists		1	30/01/2023	Explain	Chalk and Board
1.2	Tables		1	31/01/2023	Explain	Chalk and Board
1.3	images		1	01/02/2023	Understand	Chalk and Board
1.4	forms		1	02/02/2023	Remember	Chalk and Board
1.5	frames		1	03/02/2023	Discuss	Chalk and Board
1.6	Cascading Style Sheets	**T1,T2,R5,R6**	1	04/02/2023	Remember	Chalk and Board
1.7	**INTRODUCTION TO PHP**		1	06/02/2023	Understand	Chalk and Board
1.8	Declaring Variables		1	07/02/2023	Explain	Chalk and Board
1.9	Data types		1	08/02/2023	Knowledge	Chalk and Board
1.10	arrays		1	09/02/2023	Explain	Chalk and Board
1.11	strings		1	10/02/2023	Discuss	Chalk and Board
1.12	operators		1	13/02/2023	Explain	Chalk and

						Board
1.13	expressions		1	14/02/2023	Understand	Chalk and Board
1.14	Control structures		1	15/02/2023	Understand	Chalk and Board
1.15	Functions		1	16/02/2023	Understand	Chalk and Board
1.16	Reading data from web form control like Text boxes, radiobuttons,lists etc		1	17/02/2023	Examining	Chalk and Board
1.17	Handling File Uploads		2	20/02/2023 , 21/02/2023	Examining	Chalk and Board
1.18	Connecting to database(MYSQL as reference)		2	22/02/2023, 23/02/2023	Examining	Chalk and Board
1.19	Executing simple queries		1	24/02/2023	Examining	Chalk and Board
1.20	Handling results		1	25/02/2023	Examining	Chalk and Board
1.21	Handling sessions and cookies		1	27/02/2023	Examining	Chalk and Board
1.22	File Handling in PHP		1	28/02/2023	Examining	Chalk and Board
1.23	Operaters like opening,closing,reading,writing,,appending,deleting		1	1/03/2023	Examining	Chalk and Board
1.24	On text		1	2/03/2023	Examining	Chalk and

						Board
1.25	Binary files		1	3/03/2023	Examining	Chalk and Board
1.26	Listing directories		1	4/03/2023	Examining	Chalk and Board

UNIT - II

XML: Introduction to XML, Defining XML tags, their attributes and values, Document Type Definition, XML Schemas, Document Object Model, XHTML.

Parsing XML Data - DOM and SAX Parsers in java.

Objectives: To introduce XML and processing of XML Data with Java.

Outcome: Apply the concepts of XML for structurizing the web pages.

UNIT NO.	TOPIC NAME	BOOKS REFERENCED	PROPOSED NO. OF PERIODS	PROPOSED DATE	BLOOM'S TAXANOMY LEVEL	TEACHING AID
II	**XML**					
2.1	Introduction to xml	**T1,R1,R4,R5,R6**	1	6/03/2023	Remember	Chalk and Board
2.2	Defining XML tags		1	7/03/2023	Discuss	Chalk and Board
2.3	XML attributes and values		1	9/03/2023	Discuss	Chalk and Board
2.4	Document Type Definition(DTD)		2	10/03/2023, 13/03/2023	Explain	Chalk and Board
2.5	XML schemas		1	14/03/202	Explain	Chalk and

				3		Board
2.6	Document Object Model(DOM)		1	15/03/2023	Explain	Chalk and Board
2.7	XHTML Parsing		2	16/03/2023, 17/03/2023	Discuss	Chalk and Board
2.8	DOM And SAX Parsers in Java		2	18/03/2023, 20/03/2023	Remember	Chalk and Board

UNIT - III

Introduction to Servlets: Common Gateway Interface (CGI), Lifecycle of a Servlet, deploying a servlet, The Servlet API, Reading Servlet parameters, Reading Initialization parameters, Handling Http Request & Responses, Using Cookies and Sessions, connecting to a database using JDBC.

Objectives: To introduce Server side programming with Java Servlets and JSP.
Outcome: Make use of Servlets to create dynamic web pages in client-server architecture.

UNIT NO.	TOPIC NAME	BOOKS REFERENCED	PROPOSED NO. OF PERIODS	PROPOSED DATE	BLOOM'S TAXANOMY LEVEL	TEACHING AID
III	**Servlets**					
3.1	Common Gateway Interface(CGI)	T1,R1,R4,R5	1	21/03/2023	Remember	Chalk and Board
3.2	Life cycle of servlet		1	23/03/2023	Understand	Chalk and Board
3.3	Deploying of servlet		1	24/03/2023	Explain	Chalk and Board

3.4	The Servlet API		1	25/03/2023	Remember	Chalk and Board
3.5	Reading Sevlet-Parameters		1	3/04/2023	Understand	Chalk and Board
3.6	Reading Intialization parameters		1	5/04/2023	Explain	Chalk and Board
3.7	Handling http Request and Responces		2	6/04/2023	Discuss	Chalk and Board
3.8	Using cookies and Sessions		2	10/04/2023	Examining	Chalk and Board
3.9	Connecting ta a database using JDBC		2	11/04/2023	Examining	Chalk and Board

UNIT - IV

Introduction to JSP: The Anatomy of a JSP Page, JSP Processing, Declarations, Directives, Expressions, Code Snippets, implicit objects, Using Beans in JSP Pages, Using Cookies and session for session tracking, connecting to database in JSP.

Objectives: To introduce Server side programming with Java Servlets and JSP.

Outcome: Make use of JSP to develop interactive web pages.

UNIT NO.	TOPIC NAME	BOOKS REFERENCED	PROPOSED NO. OF PERIODS	PROPOSED DATE	BLOOM'S TAXANOMY LEVEL	TEACHING AID
IV	**Introduction of JSP**	**T1,R1,R2**	**1**	15/04/2023	Understand	Chalk and Board

4.1	The Anatomy of a JSP page		1	17/04/2023	Understand	Chalk and Board
4.2	JSP processing		2	18/04/2023 , 19/04/2023	Explain	Chalk and Board
4.3	Declarations		2	20/04/2023 , 21/04/2023	Discuss	Chalk and Board
4.4	Directives		2	24/04/2023 , 25/04/2023	Remember	Chalk and Board
4.5	Expressions		2	26/04/2023 , 27/04/2023	Understand	Chalk and Board
4.6	Code Snippets		2	28/04/2023 , 1/05/2023	Explain	Chalk and Board
4.7	Implicit objects		2	2/05/2023, 3/05/2023	Understand	Chalk and Board
4.8	Using Beans of JSP Pages		2	4/05/2023, 5/05/2023	Understand	Chalk and Board
4.9	Using Cookies and session tracking		1	8/05/2023	Understand	Chalk and Board
4.10	Connecting to database in JSP		2	9/05/2023, 10/05/2023	Examining	Chalk and Board

UNIT- V

Client side Scripting: Introduction to Javascript: Javascript language - declaring variables, scope of variables, functions, event handlers (onclick, onsubmit etc.), Document Object Model, Form validation. Simple AJAX application.

Objectives: To introduce Client side scripting with Javascript and AJAX.

Outcome: Apply the techniques of Java script in client side scripting.

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

UNIT NO.	TOPIC NAME	BOOKS REFERENCED	PROPOSED NO. OF PERIODS	PROPOSED DATE	BLOOM'S TAXANOMY LEVEL	TEACHING AID
V	**Client Side Scripting**	**T1,R1,R4**	**1**	15/05/2023	Understand	Chalk and Board
5.1	Introduction to Javascript		2	16/05/2023 , 17/05/2023	Understand	Chalk and Board
5.2	Javascript language-declaring variables		2	18/05/2023 , 19/05/2023	Understand	Chalk and Board
5.3	JS-scope of variables		2	22/05/2023 , 23/05/2023	Understand	Chalk and Board
5.4	Functions		2	25/05/2023 , 26/05/2023	Understand	Chalk and Board
5.5	Event handlers(onclick,onsubmit.etc)		2	30/05/2021 , 31/05/2023	Examining	Chalk and Board
5.6	Document Object Model		1	6/06/2023	Examining	Chalk and Board
5.7	Form Validation		1	9/06/2023	Examining	Chalk and Board

TEXT BOOKS:

1. Web Technologies, Uttam K Roy, Oxford University Press
2. The Complete Reference PHP – Steven Holzner, Tata McGraw-Hill

REFERENCE BOOKS:

1. Web Programming, building internet applications, Chris Bates 2nd edition, WileyDreamtech
2. Java Server Pages –Hans Bergsten, SPD O'Reilly
3. Java Script, D. Flanagan, O'Reilly,SPD.
4. Beginning Web Programming-Jon Duckett WROX.
5. Programming World Wide Web, R. W. Sebesta, Fourth Edition, Pearson.
6. Internet and World Wide Web – How to program, Dietel and Nieto, Pearson.

h)Scheme of Evaluation

Theory

Evaluation Criteria		Marks
Midterm-1	Assignment I	05
	Descriptive Paper	20
	Total	25
Midterm-2	Assignment II	05
	Descriptive Paper	20
	Total	25
	Average of Midterm-1 and Midterm-2	**25**
End-Examination		**75**
Total		**100**

3. Mapping of CO-PO&PSO

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO-1	2				2								1	1
CO-2	2	1			2								1	1

CO-3	1				2				2		1		1	2
CO-4	1				2				2		2		1	2
CO-5	1	2		2	2								1	2
Weight age	1.4	1.5		2.0	2.0				2.0		1.5		1.0	1.6

Contribution of course to program outcomes & Program Specific outcomes

Type	Course Code, Title	PO 1	PO 2	PO 3	PO 4	PO 5	PO6	PO 7	PO 8	PO 9	PO1 0	PO 11	PO 12	PSO 1	PSO 2
Theory	Web Technologi es	1.3	1.5		2.0	2.0				2.0		1.5		1.0	

Delivery Methodology

Course Delivery Methods/Modes:

1. Classroom lectures: Yes

2. Presentations: Yes
3. Laboratory sessions: Yes
4. Demos: No
5. Assignments: Yes
6. Case studies: No
7. Seminars: Yes
8. Projects: No
9. E-Learning Resources: Yes

Assessment Methodology

Course Outcome	Assessment Tool		Activity aligned to the Outcome
CO1	Unit-1	Mid I	Conducted Mid exams and Unit tests.
CO2	Unit-2		
CO3	Assignment & Unit-3		Given problems and questions to solve and conducted unit test.
CO4	Unit-4	Mid II	Conducted Mid exams and Unit tests.
CO5	Unit-5		

Note - Framed Rubrics for internal assessments and Laboratory exams.

E-Learning Resources:

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

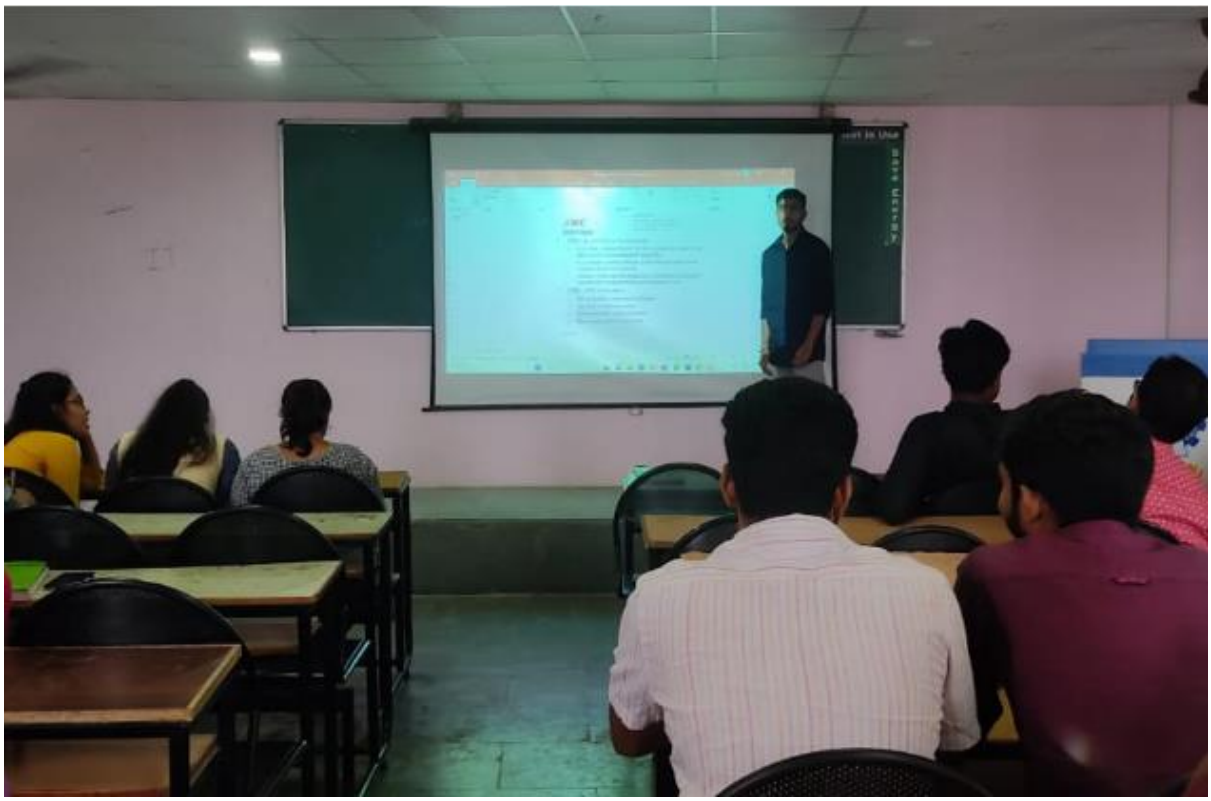
Sno	Unit	Topic	Reference link
1	1	Connecting to database in php	https://www.javatpoint.com/php-mysql-connect
2	2	Connecting to a database using jdbc	https://www.youtube.com/watch?v=y_YxwyYRJek
3	5	Event handlers	https://www.youtube.com/watch?v=OTNYOdoy7B8

Seminar topics are given to students:

S.No	Student Name	Roll Number	Topic	Signature
1	Aanchal Thakur	20R91A1201	Handling File Uploads, Connecting to database	
2	Akavaram Tejaswini	20R91A1202	Handling sessions and cookies	
3	Akula Meghana	20R91A1203	File Handling in PHP	
4	B Sathwika	20R91A1204	HTML Tags	
5	Bejjenki Chaithanya	20R91A1206	PHP variables,Data types,Operators	
6	C Ankitha	20R91A1208	PHP Arrays,String,Control,Structures,Functions	
7	Chillara Mahesh	20R91A1212	XML tags,Attributes,Values	
8	D Aadityaa	20R91A1213	Document Type Definition,XML Schemas	
9	Damidi Maheshwar	20R91A1214	Document Object Model,XHTML	
10	Enukonda Harshavardha	20R91A1217	Difference between DOM,SAX	

	n Reddy			
11	Godala Mahitha Reddy	20R91A1219	CGI,Life cycle of a Servlet,deploying a Servlet.	
12	Gunreddy Raveena	20R91A1220	The Servlet API, Reading Parameters, Initialization Parameters	
13	Inapanuri Kavya	20R91A1221	Handling Http Request & Responses	
14	K Sai Kumar	20R91A1224	JSP Processing,Declarations,Directives,Expressions	
15	Kumbam Niharika	20R91A1225	Connecting a database using JDBC.	
16	Mamidi Sai Nikhitha	20R91A1229	Connecting to database in JSP	
17	Nara Taruni	20R91A1236	Javascript,variables,functions	
18	Neerati Uday Kumar	20R91A1237	Event Handlers	
19	Nethi Sathvika	20R91A1238	How Web technology usefull inreal time.	
20	Racharla Naresh	20R91A1242	Static web page using HTML	
21	Vavilla Sandeep	21R95A1207	Connecting databade using JDBC	

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology



Collaborative learning:

Outcome of the Collaborating Learning:

With the help of collaborative learning, students have improved in the development of high-level thinking oral communication self-management leadership skills presentation skills which will be helpful in their higher studies and development of project life cycle. The students have given a presentation on "Future of Internet" which has given a complete picture to the rest of the class and it has motivated even other students to participate in the upcoming activities.

Think-Pair-Share Activity

Description: Think-Pair-Share (TPS) is a collaborative learning strategy in which students work together to solve a problem or answer a question about an assigned reading. This technique requires students to (1) think individually about a topic or answer to a question; and (2) share ideas with classmates.

Significance: Teacher can understand the different thought processes of the students while listening to the pairs and when the students share their view at the end. The interaction with students at personal level is intended to motivate those students who may not be generally interested in the topic.

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Topic : The Impact of Artificial Intelligence on Web Development
Learning Outcome : Inspiration for Innovation

S.No	Topic	Team Lead	Team members	Team Member Sign
1	The Impact of Artificial Intelligence on Web Development	Bejjenki Chaithanya	Aanchal Thakur	
			Akavaram Tejaswini	
			Akula Meghana	
			B Sathwika	
			Bejjenki Chaithanya	
			Bhukya Vishnuvardhan	
2		D Aadityaa	C Ankitha	
			Ch Akshay Kumar	
			Ch Bhavya Siva Sai Kiran	
			Chelimela Keerthana	
			Chillara Mahesh	
			D Aadityaa	
3		Godala	Damidi Maheshwar	
			Dharavath Vishnuvardhan	
			Eerla Eshwar Prasad	
			Enukonda Harshavardhan Reddy	

			Gaddam Avinash	
			Godala Mahitha Reddy	
4		Gunreddy Raveena	Gunreddy Raveena	
			Inapanuri Kavya	
			Jaya Vardhan	
			Jolge Ajay	
			Kuchipudi Saikumar	
			Kumbam Niharika	
5		Mamidi Sai Nikhitha	Kurakula Rahul	
			Kusuma Vishwesh	
			Neela Sunil	
			Mamidi Sai Nikhitha	
			Maroju Sathish	
			Midde Varun Kumar	
6		Neerati Uday Kumar	Mohammad Abdhur Rahman	
			Mohammed Muzammil Hussain	
			Nagamalla Rohith Kumar	
			Nallala Krishna Chaitanya	
			Nara Taruni	
			Neerati Uday Kumar	

7		Racharla Naresh	Nethi Sathvika	
			Oruganti Vinod	
			Polasa Vyshnavi	
			Pundra Ragasree	
			Racharla Naresh	
			S Nivas	
8		Yanagandhula Varun	S Sindhu	
			Sirandasu Sairaj	
			V Sampath Kumar	
			Yash Wasnik	
			Yanagandhula Varun	
			Manne Jayanth Kumar	
9		Vavilla Sandeep	B Manohar Reddy	
			Kothapally Sai Sumanth	
			P Sri Harsha Kumar	
			Pyata Tarun	
			Ramavath Venkatesh	
			Vavilla Sandeep	

S.No	Title	Description
1	Concept	• Know about AI-Powered Web Personalization. • Gain the knowledge about AI-Enhanced Content Creation. • Learn about different AI for Security and Cyber Threat Detection.
2	Challenges Faced	• In conventional method below average students can't able to understand the concept of AI.
3	Reason for choosing the activity for the topic	• It is important for understanding various aspects like Ubiquity of Web Development, Cutting-Edge Technology, Interdisciplinary Nature, Job Opportunities. Preparation for the Future.
4	Implementation	• Activity is planned for 50 minutes. • Discussed the concept in the class before conducting the activity. Students were formed into teams with one of the member as team leader to lead the team and check for effective time utilization during the discussions. They should then share the ideas that they have recollected. Later some among the students (pair)

		were asked to share their ideas with the entire class.
5	Feedback from Learners (Consolidated)	• The entire session was exciting and innovative. Interaction with students helped investigating the knowledge in an effective way. Students gained knowledge in this topic.

Photos

Course Material

Question Bank

Short Answer Questions

UNIT-I

1. What is PHP?

2. What is the difference between $name and $$name?
3. What are the differences between Get and post methods.
4. How can we create a database using PHP and MySQL?
5. What is the use of header() function in PHP ?
6. How can we get second of the current time using date function?
7. List out the predefined classes in PHP?
8. What type of inheritance that PHP supports?
9. What are the advantages/disadvantages of MySQL and PHP?
10. What is the difference between PHP and Javascript?

UNIT-II

1. Define XML? What are the advantages of xml?
2. List the XML syntax rules in detail.
3. Define an xml scheme show how an XML Scheme can be
4. created?
5. Explain a brief note on XML parsers?
6. Define how it is different from HTML?
7. Explain the purpose of XML schema?
8. List out the advantages of schema over DTD?
9. Explain about XML parsing done with SAX?
10. List out the three flavours of Document Type declaration?

UNIT-III

1. List out difference between web server and application server?
2. Which HTTP method is non-idempotent?
3. Explain difference between GET and POST method?
4. List out MIME Types?
5. Discuss the web application and what is its directory structure?
6. Explain about Servlet?
7. List out various phases of Servlet life cycle?
8. Build a Servlet program to illustrate parameter reading and parameter initializing. ?
9. Explain how to override service () method?
10. List the methods defined in HttpServletRequest?
11. How do you get ServletContext reference inside Servlet?
12. Which open source tag library have you used?
13. What are the differences between GET and POST method in HTTP protocol?
14. List different types of statements in JDBC?
15. Explain different types of JDBC drivers?

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

UNIT-IV

1. What are the differences between custom JSP tags and Serlets?
2. Explain the difference between JSP include directive and JSP include action.
3. Explain about Scriptlet tag?
4. Explain how to use JavaBeans from JSP pages.
5. Explain about various implicit objects?
6. How many JSP scripting elements and what are they?
7. How JSP pages the preferred API for creating a web-based client program?
8. Define Tag library descriptor (TLD)?
9. Explain the categories of JSP tags - Directives, Scripting elements, Actions?
10. List out differences between including action and include directive in JSP?
11. How do you define application wide error page in JSP?
12. Explain how to load the drivers?
13. Explain how to insert an image file (or other raw data) into a database?
14. List the Java packages which contains JDBC classes and interfaces, Java.SQL, Javax.SQL
15. Define how to open a database connection using JDBC.

UNIT-V

1. Explain how to embed JavaScript code in an HTML document.
2. Define arrays in JavaScript?
3. List the differences between Client side JavaScript Server side JavaScript?
4. Define how to create a Date Object?
5. Explain dynamic html? What is the main difference between
6. DHTML and HTML?
7. Explain the various control statements available with JavaScript.
8. Explain about a function using function constructor?
9. Explain about the Accessing Elements using JavaScript?
10. Define a boolean operator that a JavaScript support?
11. Explain about String object in JavaScript?

Long Answer Questions

UNIT-I

1. What are the different types of errors in PHP?
2. What is the functionality of the function strstr and stristr?
3. Explain about various datatypes in PHP.
4. Explain about Arrays in PHP.
5. List and Explain the string functions in PHP.
6. List the statements that are used to connect PHP with MySQL.
7. How PHP is different from PHP Script? Explain.
8. Explain PHP form processing with an example.
9. How can I retrieve values from one database server and store them in other database server using PHP?

10. What are the differences between Get and post methods in form submitting. Give the case where we can use get and we can use post methods?

UNIT-II

1. Explain and show how XML is useful in defining data for web applications.
2. Explain the various terms related to Document Type Definition.
3. Design an XML schema for hospital information management.Include every feature available with schema.
4. Explain how styling XML with cascading style sheets is done for the library information domain.
5. List and Explain the important features of XML which make it more suitable than HTML for creating web related services.
6. Define an xml scheme to show how an XML Scheme can be created
7. Define Attributes in XML .Also different types of attributes
8. List the elements in XML .Also different types of content of Elements.
9. How do you define the elements of an XML document in an XML Schema?
10. How do you set default and fixed values for simple Elements?

UNIT-III

1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
2. What is the security issues related to Servlets.
3. Explain how HTTP POST request is processed using Servlets
4. Explain how cookies are used for session tracking?
5. Explain about Tomcat web server.
6. What is Servlet? Explain life cycle of a Servlet?
7. What are the advantages of Servlets over CGI
8. What is session tracking? Explain different mechanisms of session tracking?
9. What is the difference between Servlets and applets?
10. What is the difference between doGet() and doPost()?

UNIT-IV

1. Explain about JSP Elements?
2. List the different Action Tags used in JSP with their functionality
3. Explain the types of Scripting tags and Directive tags in JSP.
4. Explain about the usage of JavaBean Component in JSP.
5. Explain briefly about the Problem with Servlets
6. Describe the Anatomy of JSP Page
7. Explain the MVC architecture and write a JSP program which prints the current date?
8. List the types of JSP Implicit Objects.
9. How application data can be shared in JSP. Explain.
10. Explain sharing and application data in JSP application Development
11. List the methods in request object.
12. Explain about the JSP Directive Elements? Explain each one of them in detail?
13. Explain JSP application design with suitable example?

UNIT-V

1. Build a JavaScript program to convert distance in kilometers, miles to meters or inches
2. Build a java script to verify a phone number, email-id and date formats.
3. Compare and contrast HTML and DHTML with suitable examples.
4. Explain the need for scripting languages in web programming.
5. Explain the features of Java Script.
6. What is JavaScript? Write the features of JavaScript?
7. Write the code in JavaScript to open a new window when a
8. Explain any three objects of JavaScript
9. What is form validation? Explain with example?
10. What is an event? How can we handle events in JavaScript?

Quiz Question bank
UNIT-I

1 Which of the following type of variables are whole numbers, without a decimal point, like 4195?
a). Integers
b). Doubles
c). Booleans
d). Strings

2 Which of the following function returns selected parts of an array?
a). array_reverse()
b). array_search()
c). array_shift()
d). array_slice()

3 Which of the following provides content type of the uploaded file in PHP?
a). $_FILES['file']['tmp_name']
b). $_FILES['file']['name']
c). $_FILES['file']['size']
d). $_FILES['file']['type']

4 Which of the following method can be used to create a MySql database using PHP?
a). mysql_connect()
b). mysql_query()
c). mysql_close()
d). mysql_Query()

5 Which of the following is used to get information sent via get method in PHP?
a). $_GET
b). $GET
c). $GETREQUEST
d). $get

6 Which of the following tags is not a valid way to begin and end a PHP code block?
a). <? ?>
b). <% %>
c). <?=?>
d). <?php ?>

7 What is the difference between print () and echo ()?
a). print() can be used as part of an expression, while echo() can't
b). echo() can be used as part of an expression, while print() can't
c). echo() can be used in the CLI version of PHP, while print() can't
d). print() can be used in the CLI version of PHP, while echo() can't

8 Which of the following type of variables are floating-point numbers, like 3.14159 or 49.1?
a). Integers
b). Doubles
c). Booleans
d). Strings

9 Variable name in PHP starts with
a). ! (Exclamation)
b). $ (Dollar)
c). & (Ampersand)
d). # (Hash)

10 Which of the following is the default file extension of PHP?
a). .php
b). .hphp
c). .xml
d). .html

11 Which of the following is used to display the output in PHP?
a). **echo**
b). write
c). scan
d). get

12 Which of the following is correct to add a comment in php?
a). & …… &
b). // **……**
c). /*/ …… /*/
d). *.....*

13 Which of the following is the use of strlen() function in PHP?
a). The strlen() function returns the type of string

b). The strlen() function returns the length of string
c). The strlen() function returns the value of string
d). The strlen() function returns both value and type of string

14 Which of the following is used for concatenation in PHP?
a). + (plus)
b). * (Asterisk)
c). . (dot)
d). append()

15 Which of the following is the correct way of defining a variable in PHP?
a). $variable name = value;
b). $variable_name = value;
c). $variable_name = value
d). $variable name as value;

16 What is the use of fopen() function in PHP?
a). The fopen() function is used to open folders in PHP
b). The fopen() function is used to open remote server
c). The fopen() function is used to open files in PHP
d). The fopen() function is used to open remote files in server

17 Which of the following is the correct use of the strcmp() function in PHP?
a). The strcmp() function is used to compare the strings excluding case
b). The strcmp() function is used to compare the uppercase strings
c). The strcmp() function is used to compare the lowercase strings
d). The strcmp() function is used to compare the strings including case

18 Which of the following is the correct way to open the file "sample.txt" as readable?
a). fopen("sample.txt", "r");
b). fopen("sample.txt", "r+");
c). fopen("sample.txt", "read");
d). fopen("sample.txt");

19 Which of the following function displays the information about PHP and its configuration?
a). php_info()
b). phpinfo()
c). info()
d). php()

20 Which of the following function is used to sort an array in descending order?
a). sort()
b). asrot()
c). dsort()
d). rsort()

21 HTML stands for
a). Hypertext Markup List
b). Hyperprocessor Markup Language
c). Hypertext Media Language
d). Hypertext Markup Language

22 Which of the following is NOT a super global variable in PHP?
a). $_GET
b). $GLOBAL
c). $_POST
d). $_LOCAL

23 CSS stands for
a). Choice based style sheet
b). Common Style sheets
c). cross Style sheet
d). Cascading Style sheets

24 PHP is _____ scripting language.
a). Server-side
b). Clint-side
c). Middle-side
d). Out-side

25 By using which tag we are able to insert table on to the webpage
a). <radiobutton>
b). <textarea>
c).
d). <table>

26 Which of the following is not the scope of Variable in PHP?
a). Local
b). Global
c). Static
d). Extern

27 In PHP a variable needs to be declare before assign
a). Depends on website
b). FALSE
c). Depends on server
d). TRUE

28 Which of the following keyword is used to return a value from a function?
a). set
b). send
c). pass
d). return

29 PHP scripts are executed on
a). ISP Computer
b). Client Computer
c). Server Computer
d). It depends on PHP scripts

30 How PHP files can be accessed?
a). Through Web Browser
b). Through HTML files
c). Through Web Server
d). Through Client

UNIT-II

1 What is the correct syntax of the declaration which defines the XML version?
a). <xml version="A.0" />
b) <?xml version="1.0"?>
c). <?xml version="A.0" />
d) <?xml Version="1.0"/>

2 What does XML stand for?
a). eXtra Modern Link
b) eXtensible Markup Language
c). Example Markup Language
d) X-Markup Language

3 Which statement is true?
a). All the statements are true
b) All XML elements must have a closing tag
c). All XML elements must be lower case
d) All XML documents must have a DTD

4 Is it easier to process XML than HTML?
a). Yes
b) No
c). Sometimes
d) Can't say

5 Which of the following programs support XML or XML applications?
a). Internet Explorer 5.5

b) Opera
c). RealPlayer.
d) Crome
6 Kind of Parsers are
a). well-formed
b) well-documented
c). non-validating and validating
d) well-validate
7 Well-formed XML document means
a). it contains a root element
b) it contain an element
c). it contains one or more elements
d) must contain one or more elements and root element must contain all other elements
8 Comment in XML document is given by
a). <? -- -->
b) <! -- --!>
c). <! -- -->
d) </-- -- >
9 When processing an output XML, "new line" symbols
a). are copied into output "as is", i.e. "CR+LF" for Windows, CR for Macintosh, LF for UNIX.
b) are converted to single LF symbol
c). are converted to single CR symbol
d) are discarded
10 Which of the following strings are a correct XML name?
a). _myElement
b) my Element
c). #myElement
d) $myElement
11 Which of the following strings are a correct XML name?
a). xmlExtension
b) xslNewElement
c). XMLElement#123
d) XSLNewElement
12 Which of the following XML fragments are well-formed?
a). <? xml ?>
b) <?xml version="A.0"?>
c). <?xml encoding="JIS"?>
d) <?xml encoding="JIS" version="A.0"?>
13 Valid XML document means (most appropriate)

a). the document has root element
b) the document contains at least one or more root element
c). the XML document has DTD associated with it & it complies with that DTD
d) Each element must nest inside any enclosing element property
14 XML uses the features of
a). HTML
b) XHTML
c). VML
d) SGML
15 XML document can be viewed in
a). IE C.0
b) IE B.0
c). IE 6.0
d) IE X.0
16 What does DTD stand for?
a). Direct Type Definition
b) Document Type Definition
c). Do The Dance
d) Dynamic Type Definition
17 XML is designed to ____ and ____ data.
a). design, style
b) design, send
c). store, style
d) store, transport
18 What is the full form of XSD
a). XHTML Schema Definition
b) XML Schema Definition
c). XSLT Schema Definition
d) XSL Schema Definition
19 An XML document is a string of ____.
a). HTML character codes
b) XML codes
c). ASCII codes
d) Characters
20 In an XML document, a tag is a markup construct that starts with ___ and ends with.
a). <, >
b) <!--, -->
c). <#, >
d) @, @
21 A Document Type Definition (DTD) is a set of ____ which is used to define the type of

document for an SGML-family markup language.

a). markup definition
b) markup document
c). main declarations
d) markup declarations

22 An XML element can have ____.

a). multiple attributes
b) only two unique attributes
c). multiple unique attributes
d) only one unique attribute

23 Which is the correct syntax to link XML file with CSS?

a). <?xml type="text/css" href="file.css"?>
b) <?xml type="text/css" src="file.css"?>
c). <?xml-stylesheet type="text/css" href="file.css"?>
d) <?xml-stylesheet type="text/css" src="file.css"?>

24 Which of the following attributes is used to define a namespace.

a). xmlns
b) Xml-ns
c). Name-space
d) ns

25 ____is used to read XML documents and provide access to their content and structure.

a). XML Processor
b). XML Pre-processor
c). XML Compiler
d). XML Interpreter

UNIT-III

1 What type of servlets use these methods doGet(), doPost(),doHead, doDelete(), doTrace()?

a). Genereic Servlets
b). **HttpServlets**
c). Httpreq
d). Httpres

2 Which of the following is the correct order of servlet life cycle phase methods?

a). **init, service, destroy**
b). initialize, service, destroy
c). init, execute, destroy
d). init, service, delete

3 What is javax.servlet.Servlet?
a). **interface**
b). abstract class
c). concreate class
d). inheritance
4 Which of the following method can be used to get the value of form parameter?
a). **request.getParameter**
b). request.getParameterValues
c). request.getParameterNames
d). request.getParameterNumber
5 Which of the following code can be used to write a cookie?
a). request.addCookiecookie
b). **response.addCookiecookie**
c). Header.addCookiecookie
d). Header.addCookie
6 Which of the following code can be used to force any content in the buffer to be written to the client?
a). request.flushBuffer
b). response.flush
c). response.flushBuffer
d). request.flush
7 Which of the following code can be used to redirect user to different url?
a). request.sendRedirectlocation
b). **response.sendRedirectlocation**
c). header.sendRedirectlocation
d). header.sendRedirect
8 Which of the following is the correct order of filter life cycle phase methods?
a). **init, service, destroy**
b). initialize, service, destroy
c). init, doFilter, destroy
d). init, service, delete
9 Which element of web.xml is used to specify the error handler in servlets?
a). **error-page**
b). error-handler
c). exception
d). exception-handler
10 Which of the following code is used to get PrintWriter object in servlet?
a). **response.getWriter**
b). request.getWriter
c). new PrintWriter
d). getWriter
11 Which of the following code is used to get session in servlet?
a). **request.getSession**
b). response.getSession

c). new Session
d). getSession

12 Which of the following code retrieves the context of the request?
a). new ClassContextPath
b). **request.getContextPath**
c). response.getContextPath
d). getContextPath

13 Which of the following code checks whether this request was made using a secure channel, such as HTTPS?
a). response.isSecure
b). **request.isSafe**
c). Header.isSecure
d). Header.isSafe

14 When destroy() method of a filter is called?
a). **The destroy() method is called only once at the end of the life cycle of a filter**
b). The destroy() method is called after the filter has executed doFilter method
c). The destroy() method is called only once at the begining of the life cycle of a filter
d). The destroyer() method is called after the filter has executed

15 Which method is used to send the same request and response objects to another servlet in RequestDispacher ?
a). **forward()**
b). sendRedirect()
c). send()
d). Redirect()

16 Which method in session tracking is used in a bit of information that is sent by a web server to a browser and which can later be read back from that browser?
a). HttpSession
b). URL rewriting
c). Cookies
d). Hidden form fields

17 Which are the examples of Application Server?
a). Apache
b). Tomcat
c). JBoss
d). Weblogic

18 How many techniques are used in Session Tracking?
a). **4**
b). 3
c). 2
d). 5

19 Which method is used to specify before any lines that uses the PintWriter?
a). setPageType()
b). setContextType()
c). setContentType()
d). setResponseType()

20 Which of the following code encodes the specified URL by including the session ID in it?

a). **response.encodeURL(url)**
b). request.encodeURL(url)
c). Header.encodeURL(url)
d). response.encodeurl(url)

21 Which package provides interfaces and classes for writing Servlets.
a). javax.servlet
b). javax.java.servlet
c). javax.awt.servlet
d). javax.swing.servlet

22 How many arguments does the service() methos
a). 1
b). **2**
c). 3
d). 4

23 Which of the following method is used to add a cookie to the response
a). **addCookie()**
b). sendCookie()
c). installCookie()
d). inserCookie()

24 Which method can be used to access the ServletConfig object?
a). getServletInfo()
b). **getServletConfig()**
c). getInitParameters()
d). getConfig()

25 A deployment descriptor describes
a). Web component request and response objects
b). Web component request settings
c). **Web component settings**
d). Web component response settings

UNIT-IV

1 Which page directive should be used in JSP to generate a PDF page?
a). **contentType**
b). generatePdf
c). typePDF
d). contentPDF

2 Which tag should be used to pass information from JSP to included JSP?
a). **Using <%jsp:page> tag**
b). Using <%jsp:param> tag
c). Using <%jsp:import> tag
d). Using <%jsp:useBean> tag

3 Application is instance of which class?

a). javax.servlet.Application
b). javax.servlet.HttpContext
c). javax.servlet.Context
d). **javax.servlet.ServletContext**

4 Which option is true about session scope?
a). Objects are accessible only from the page in which they are created
b). **Objects are accessible only from the pages which are in same session**
c). Objects are accessible only from the pages which are processing the same request
d). Objects are accessible only from the pages which reside in same application

5 Which one is the correct order of phases in JSP life cycle?
a). Initialization, Cleanup, Compilation, Execution
b). Initialization, Compilation, Cleanup, Execution
c). Compilation, Initialization, Execution, Cleanup
d). Cleanup, Compilation, Initialization, Execution

6 "request" is instance of which one of the following classes?
a). Request
b). HttpRequest
c). HttpServletRequest
d). ServletRequest

7 What is the full form of JSP
a). Java Servlet Pages
b). Java Server Pages
c). Java Small Pages
d). Java Special Pages

8 Which of the following is not a method of JSP's servlet?
a). _jspService()
b). jspDestroy()
c). jspService()
d). jspInit()

9 Which is not a directive?
a). include
b). page
c). export
d). useBean

10 Which is mandatory in <jsp:useBean /> tag?
a). **id, class**
b). id, type
c). type, property
d). type,id

11 Which one of the following is correct for directive in JSP?
a). %@directive%
b). <%!directive%>
c). <%directive%>
d). <%=directive%>

12 Which of the following action variable is used to include a file in JSP?
a). jsp:setProperty
b). jsp:getProperty
c). jsp:include
d). jsp:plugin

13 What is a Jsp page translated into?
a). CGI
b). Servlet
c). Applet
d). JavaBean

14 Which type of driver makes a JDBC-ODBC bridge.
a). Type1 Driver
b). Type2 Driver
c). Type3 Driver
d). Type4 Driver

15 Which of the following scopes is not valid with respect to JavaBean in JSP
a). response
b). session
c). request
d). application

16 Which attribute uniquely identification element?
a). **ID**
b). Class
c). Name
d). Scope

17 "out" is implicit object of which class?
a). javax.servlet.jsp.PrintWriter
b). javax.servlet.jsp.SessionWriter
c). javax.servlet.jsp.SessionPrinter
d). **javax.servlet.jsp.JspWriter**

18 Which object stores references to the request and response objects?
a). sessionContext
b). **pageContext**
c). HttpSession
d). sessionAttribute

19 What temporarily redirects response to the browser?
a). <jsp:forward>
b). %@directive%
c). response.sendRedirect(URL)
d). response.setRedirect(URL)

20 Which tag is used to set a value of a JavaBean?
a). **<c:set>**
b). <c:param>
c). <c:choose>
d). <c:forward>

21 Java code is embedded under which tag in JSP?
a). Declaration
b). **Scriptlet**
c). Expression
d). Comment

22 Which of the following is not a directive in JSP?
a). page directive
b). include directive
c). taglib directive
d). **command directive**

23 Which JDBC driver Type(s) can be used in either applet or servlet code?
a). Type 1 and Type 2
b). Type 3 and Type 4
c). Type 4 only
d). Type 1 and Type 3

24 In JSP,a Canvas object provides access to a Graphics object via one of its method called…
a). getCanvas()
b). paint()
c). getPaint()
d). getGraphics()

25 What method is used to specify a container's layout in JSP
a). setContainerLayout()
b). setLayout()
c). setConLayout()
d). layout()

UNIT-V

1 Which type of JavaScript language is ___
a). Object-Oriented
b). Object-Based

c). Assembly-language
d). High-level

2 Which of the following is the correct syntax to create a cookie using JavaScript?
a). document.cookie = 'key1 = value1; key2 = value2; expires = date';
b). browser.cookie = 'key1 = value1; key2 = value2; expires = date';
c). window.cookie = 'key1 = value1; key2 = value2; expires = date';
d). navigator.cookie = 'key1 = value1; key2 = value2; expires = date';

3 Which of the following is the correct syntax to redirect a url using JavaScript?
a). document.location='http://www.newlocation.com';
b). browser.location='http://www.newlocation.com';
c). navigator.location='http://www.newlocation.com';
d). window.location='http://www.newlocation.com';

4 Which of the following is the correct syntax to print a page using JavaScript?
a). window.print;
b). browser.print;
c). navigator.print;
d). document.print;

5 Which built-in method returns the character at the specified index?
a). characterAt
b). getCharAt
c). charAt
d). getChar

6 Which built-in method combines the text of two strings and returns a new string?
a). append
b). concat
c). attach
d). delete

7 The _____ to the directory or web page that set the cookie
a). Secure
b). Expires
c). Domain
d). Path

8 JavaScript code between a pair of "script" tags are called _____
a). Non-inline
b). External
c). Referenced
d). Inline

9 _____ JavaScript is also called client-side JavaScript.
a). Microsoft
b). Navigator
c). LiveWire
d). Native

10 _____ JavaScript statements embedded in an HTML page can respond to user events such as mouse-clicks, form input, and page navigation.
a). Client-side

b). Server-side
c). Local
d). Native

11 The main purpose of JavaScript in web browser is to …
a). Creating animations and other visual effects
b). User Interface
c). Visual effects
d). User experience

12 When are the keyboard events fired?
a). When user manually calls the button
b). When user clicks a key
c). When the user calls the modifier
d). When the user calls the instruction

13 Which property is used to specify the key type when pressed?
a). keyCode
b). keyType
c). keyName
d). keyProperty

14 In general, event handler is nothing but _____
a). function
b). interface
c). event
d). handler

15 When will the browser invoke the handler?
a). Program begins
b). Any event occurs
c). Specified event occurs
d). Webpage loads

16 Which property specifies the property of the event?
a). Type
b). Target
c). Manner
d). Program

17 The process by which the browser decides which objects to trigger event handlers on is _____
a). Event Triggering
b). Event Listening
c). Event Handling
d). Event propagation

18 ……. is universally supported also it works in all browsers including IE, and works for all handlers, regardless of how they are registered.
a). Event bubbling
b). Event handling
c). Event capturing
d). Event registering

19 …… objects have altKey, ctrlKey, metaKey, and shiftKey properties, which are set to True if the corresponding modifier key is held down when the event occurs.
a). Key down
b). Key event
c). Key up
d). Keypress

20 Which is the opposite to the load event in JavaScript?
a). dontload
b). postload
c). preload
d). unload

21 When will the browser invoke the handler?
a). Program begins
b). Any event occurs
c). Specified event occurs
d). Webpage loads

22 The events that represents occurrences related to the browser window are
a). Window
b). Element
c). Display
d). Handlers

23 ……, only works with event handlers registered with addEventListner() when the third argument is True.
a). Event bubbling
b). Event handling
c). Event capturing
d). Event registering

24 …….. allows the same event handler function to be registered more than once. When an event of the specified type occurs, the registered function will be invoked as many times as it was registered.
a). addEvent()
b). addMultipltEvent()
c). attachEvent()
d). reattachEvent()

25 Invoking ……. more than once on the same object with the same arguments has no effect, the handler function remains registered only once, and repeated invocation does not alter the order in which handlers are invoked.
a). addEvent()
b). addMultipltEvent()
c). attachEvent()
d). addEventListener()

Descriptive question bank for assignment

1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
2. What is the security issues related to Servlets.
3. Explain how HTTP POST request is processed using Servlets
4. Explain how cookies are used for session tracking?
5. Explain about Tomcat web server.
6. What is Servlet? Explain life cycle of a Servlet?
7. What are the advantages of Servlets over CGI.
8. What is session tracking? Explain different mechanisms of session tracking?
9. What is the difference between Servlets and applets?
10. What is the difference between doGet() and doPost().
11. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
12. List and explain the classes and interfaces of javax.servlet.http package.
13. Build a Servlet that handles HTTP get Request
14. Describe about session tracking with relevant code snippet.
15. "Servlet offer several advantages over CGI". Justify.
16. Explain about Security Issues in Servlet
17. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
18. Build a Servlet program to illustrate parameter reading and parameter initializing.
19. Explain Cookies session tracking with relevant code snippet.
20. List the methods defined in HttpServletRequest.

Sets of copies of old question papers

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

R16

Code No: 136EN

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year II Semester Examinations, May - 2019
WEB TECHNOLOGIES
(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Note: This question paper contains two parts A and B.
Part A is compulsory which carries 25 marks. Answer all questions in Part A. Part B consists of 5 Units. Answer any one full question from each unit. Each question carries 10 marks and may have a, b, c as sub questions.

PART - A

(25 Marks)

1.a) How do you reverse a string in PHP without using any built in functions? [2]
b) What are the storage engines used by MySQL? [3]
c) What is Document Object Model? [2]
d) How does one link external style sheet in a XHTML document? [3]
e) Write the purpose of cookies. [2]
f) What is session tracking? Explain. [3]
g) List down the advantages of Java beans. [2]
h) How JSP page is compiled? [3]
i) What is the scope of variables in java script? [2]
j) How the keyword "new" is used to create objects in java script? [3]

PART - B

(50 Marks)

2.a) Discuss about various functions used in PHP with examples.
b) Write PHP code to create a login page for a web application. [5+5]

OR

3.a) Discuss about various types of PHP interpreters.
b) Write a program in PHP to find out length of the string "This is my first program". [5+5]

4.a) Explain about XML core tags and flow control tags.
b) Show how an XML schema can be created. [5+5]

OR

5.a) Explain the advantages of XML schemas over DTDs.
b) Differentiate between DOM and SAX parsers in java. [5+5]

6.a) What is CGI? List the CGI environmental variables.
b) Explain the life cycle of a Servlet with a neat sketch. [5+5]

OR

7.a) Discuss the process of deploying a web application.
b) How to handle http request and responses? Explain. [5+5]

8.a) Write about the components of JSP and explain.
b) How to connect database connection through JSP? Illustrate with example. [5+5]

OR

9.a) Write in brief about JSP tag extensions and libraries.
b) How to create and make use of beans in JSP pages? Demonstrate with example. [5+5]

10.a) Explain about objects, methods and events in java scripts.
b) Write a java script to change text color of HTML elements. [5+5]

OR

11.a) What is the functioning of the java script keyword "this" and "dot" operator? Explain.
b) Write a java script to validate a form consisting of a hall ticket number as username and mobile number as password. Also navigate to another web page after validation. [5+5]

---ooOoo---

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

R18

Code No: 155DN

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year I Semester Examinations, March - 2021
WEB TECHNOLOGIES
(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

1. a) Explain different types of arrays used in PHP with examples.
   b) Write a PHP script to read and write into a file. [7+8]

2. a) What is XML DOM ? How DOM parses the XML file? [8+7]
   b) Explain different data types in XML schema.

3.a) Demonstrate the use of cookies in servlets with an example.
   b) Describe how an HTTP Servlet handles its client requests. [7+8]

4.a) What are JSP Code snippets? Develop a JSP program to display current date and time.
   b) Discuss various implicit objects in JSP? [7+8]

5.a) Write a JavaScript to display whether given number prime or not.
   b) How do you create a function using function overloading? [7+8]

6.a) Differentiate between for and foreach statements in PHP with examples.
   b) Discuss the use of frames in creation of HTML document. [7+8]

7.a) Explain with suitable examples, difference between get and post in servlets.
   b) Describe the procedure for validating XML documents against a schema. [7+8]

8.a) How to create a Date object using JavaScript?
   b) What is CSS? Describe various methods to include CSS in webpage. [7+8]

---ooOoo---

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

R16

Code No: 136EN

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year II Semester Examinations, July/August - 2021
WEB TECHNOLOGIES
(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

1. Explain the steps involved in connecting MySQL database from PHP. Write an example PHP script illustrating how to retrieve and display records from database. [15]

2.a) Write PHP program to copy the content of one file to another. [7+8]
b) Explain about DOM based XML processing.

3.a) Give a brief note on XML schemas.
b) Define SAX. How SAX parses the XML file? Explain. [7+8]

4. What is Servlet? Write servlet program for displaying "Hallow World". [15]

5. What is cookies ? Discuss with an example how to use them. [15]

6.a) How to use Scripting Elements in JSP ? Explain.
b) Write the steps in connection database to the JSP page. [7+8]

7.Discuss the Document Object Model in JavaScript in detail. [15]

8. What is AJAX? Explain how to implement AJAX with example. [15]

---ooOoo---

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

R16

Code No: 136EN

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech III Year II Semester Examinations, November/December - 2021
WEB TECHNOLOGIES
(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

1. How to connect MySQL using PHP? Write a program to executing simple queries. [15]

2.a) Discuss the differences between XML and XHTML. [8+7]
b) Give a brief note on DOM.

3.a) Write a servlet program to read the name and values of parameters of the client request.
b) Mention and explain the uses of the servlet. [8+7]

4.a) Write the steps in connection database to the JSP page. [7+8]
b) List out the classes and interfaces in javax.servlet.* package.

5. Give a brief note on built-in Objects in JavaScript with example. [15]

6. Write a PHP program to find factorial of a given number using functions. [15]

7.a) Explain how an XML Schema is created. [8+7]
b) Illustrate the XML attributes and values.

8. List and explain the web servers that support CGI programming. [15]

---ooOoo---

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Analysis of student performance in the course

Performance Index (Theory)

S.No	Hall Ticket No	Name of the Student	Mid-1	Mid-2	Avg
1	20R91A1201	Aanchal Thakur	25	25	25
2	20R91A1202	Akavaram Tejaswini	25	25	25
3	20R91A1203	Akula Meghana	16	22	19
4	20R91A1204	B Sathwika	25	25	25
5	20R91A1205	Barla Sai Kiran Reddy	5	0	2.5
6	20R91A1206	Bejjenki Chaithanya	25	25	25
7	20R91A1207	Bhukya Vishnuvardhan	15	25	20
8	20R91A1208	C Ankitha	23	25	24
9	20R91A1209	Ch Akshay Kumar	5	25	15
10	20R91A1210	Ch Bhavya Siva Sai Kiran	9	23	16
11	20R91A1211	Chelimela Keerthana	25	25	25
12	20R91A1212	Chillara Mahesh	16	23	19.5
13	20R91A1213	D Aadityaa	25	25	25
14	20R91A1214	Damidi Maheshwar	25	25	25
15	20R91A1215	Dharavath Vishnuvardhan	8	25	16.5
16	20R91A1216	Eerla Eshwar Prasad	8	19	13.5
17	20R91A1217	Enukonda Harshavardhan	21	21	21
18	20R91A1218	Gaddam Avinash	23	25	24
19	20R91A1219	Godala Mahitha Reddy	16	23	19.5
20	20R91A1220	Gunreddy Raveena	25	25	25
21	20R91A1221	Inapanuri Kavya	25	25	25
22	20R91A1222	Jaya Vardhan	22	25	23.5
23	20R91A1223	Jolge Ajay	5	5	5
24	20R91A1224	Kuchipudi Saikumar	25	25	25
25	20R91A1225	Kumbam Niharika	25	25	25
26	20R91A1226	Kurakula Rahul	5	25	15
27	20R91A1227	Kusuma Vishwesh	16	25	20.5
28	20R91A1228	Neela Sunil	15	22	18.5
29	20R91A1229	Mamidi Sai Nikhitha	24	25	24.5
30	20R91A1230	Maroju Sathish	25	23	24

31	20R91A1231	Midde Varun Kumar	25	22	23.5
32	20R91A1232	Mohammad Abdhur Rahman	21	25	23
33	20R91A1233	Mohammed Muzammil	5	10	7.5
34	20R91A1234	Nagamalla Rohith Kumar	20	25	22.5
35	20R91A1235	Nallala Krishna Chaitanya	21	25	23
36	20R91A1236	Nara Taruni	25	25	25
37	20R91A1237	Neerati Uday Kumar	25	25	25
38	20R91A1238	Nethi Sathvika	25	25	25
39	20R91A1239	Oruganti Vinod	6	25	15.5
40	20R91A1240	Polasa Vyshnavi	25	22	23.5
41	20R91A1241	Pundra Ragasree	25	25	25
42	20R91A1242	Racharla Naresh	25	25	25
43	20R91A1243	S Nivas	25	25	25
44	20R91A1244	S Sindhu	25	25	25
45	20R91A1245	Sirandasu Sairaj	20	25	22.5
46	20R91A1246	V Sampath Kumar	18	20	19
47	20R91A1247	Yash Wasnik	5	5	5
48	20R91A1248	Yanagandhula Varun	25	25	25
49	20R91A1249	Manne Jayanth Kumar	20	25	22.5
50	21R95A1201	B Manohar Reddy	22	25	23.5
51	21R95A1202	Kothapally Sai Sumanth	22	25	23.5
52	21R95A1203	P Sri Harsha Kumar	25	25	25
53	21R95A1204	Pyata Tarun	25	23	24
54	21R95A1205	Ramavath Venkatesh	25	23	24
55	21R95A1207	Vavilla Sandeep	25	24	24.5

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

Answer book copies

Internal Papers(MID-I & MID-II)

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
B.Tech V (III-II)Semester MID-I Examinations, January -2023

Name of the subject: ***WEB TECHNOLOGIES*** ***Date:***

Time: 01:20min. **Max. Marks: 20**

All questions carry equal marks

Q. No	Questions	Marks	Bloom's Levels	CO Map
Q.1	Write a HTML program on Table creation.	5	3	4
Q.2	How to connect MySQL using PHP? Write a program for executing simple queries.	5	3	4
Q.3	Define DTD in XML.Difference between internal and external DTDs.	5	3	5
Q.4	Explain about XML Schema.	5	2	5

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
Approved by AICTE, Affiliated by JNTUH, Accreditated by NAAC- 'A' Grade
Medbowli, Meerpet, Balapur, Hyderabad, Telangana- 500097
Mob: 9393959597. Email: info@tkrec.ac.in, deanacademics@tkrec.ac.in
Department of Information Technology

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE
(UGC-Autonomous)
B. Tech VI (III-II) Semester MID-II Examinations, June-2023

Name of the subject:Web Technologies ***Date:16-06-2023***

Time: 01:20min. **Max. Marks: 20**

All questions carry equal marks

Q. No	Questions	Marks	Bloom's Levels	CO Map
Q.1	Explain JDBC in detail with an example using servlets.	5	5	CO2
Q.2	Write about JSP processing.	5	`1	CO2
Q.3	Explain Functions in JavaScript with example.	5	5	CO2
Q.4	Explain the following: a. Declarations in JSP b, Scriplets in JSP	5	3	CO2

Assignment Copies

Assignment I

S.No	Roll Number	Student Name	Assignment Question
1	20R91A1201	Aanchal Thakur	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
2	20R91A1202	Akavaram Tejaswini	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
3	20R91A1203	Meghana. Akula	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
4	20R91A1204	Bingi Sathwika	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
5	20R91A12	Barla	1. What is the difference between Servlets and

	05	Saikiranreddy	applets?
			2. What is the difference between doGet() and doPost().
6	20R91A1206	Bejjenki Chaithanya	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
7	20R91A1207	Bhukya Vishnuvardhan	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
8	20R91A1208	C. Ankitha	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
9	20R91A1209	Ch. Akshay Kumar	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
10	20R91A1210	Chintada Bhavya Siva Saik	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
11	20R91A1211	Chelimela Keerthana	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
12	20R91A1212	Chillara. Mahesh	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
13	20R91A1213	D. Aadityaa	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
14	20R91A1214	D. Maheshwar Naidu	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
15	20R91A1215	Dharavath Vishnuvardhan	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
16	20R91A1216	Erla Eswar Prasad	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
17	20R91A12	Enukonda	1. Explain how HTTP POST request is processed

	17	Harshavardhan Re	using Servlets
			2. Explain how cookies are used for session tracking?
18	20R91A1218	Gaddam Avinash	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
19	20R91A1219	Godala Mahitha Reddy	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
20	20R91A1220	G. Raveena	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
21	20R91A1221	Inapanuri Kavya	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
22	20R91A1222	Jaya Vardhan	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
23	20R91A1223	Jolge Ajay	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
24	20R91A1224	Kuchipudi Sai Kumar	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
25	20R91A1225	K Niharika Reddy	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
26	20R91A1226	Kurakula Rahul	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
27	20R91A1227	Kusuma Vishwesh	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
28	20R91A1228	Neela Sunil	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
29	20R91A1229	Mamidi. Sainikhitha	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?

30	20R91A1230	Maroju Sathish	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
31	20R91A1231	Midde Varun Kumar	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
32	20R91A1232	Mohammed Abdhur Rahman	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
33	20R91A1233	Mohammed Muzammil Hussain	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
34	20R91A1234	Nagamalla Rohith Kumar	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
35	20R91A1235	Nallala Krishna Chaitanya	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
36	20R91A1236	Nara Taruni	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
37	20R91A1237	Neerati Uday Kumar	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
38	20R91A1238	Nethisathvika	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
39	20R91A1239	Oruganti Vinod	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
40	20R91A1240	Polasa Vyshnavi	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
41	20R91A1241	Pundra Raga Sree Reddy	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.

			2. What is the security issues related to Servlets.
42	20R91A1242	Racharla Naresh	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
43	20R91A1243	S. Nivas	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
44	20R91A1244	Singireddy Sindhu	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
45	20R91A1245	Sirandasu Sairaj	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
46	20R91A1246	V. Sampath Kumar	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
47	20R91A1247	Yash Wasnik	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
48	20R91A1248	Yanagandhula Varun	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?
49	20R91A1249	Manee Jayanth Kumar	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
50	21R95A1201	B Manohar Reddy	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().
51	21R95A1202	Kothapally Sai Sumanth	1. Define a session tracker that tracks the number of accesses and last access data of a particular web page.
			2. What is the security issues related to Servlets.
52	21R95A1203	P Sri Harsha Kumar	1. Explain how HTTP POST request is processed using Servlets
			2. Explain how cookies are used for session tracking?
53	21R95A1204	Pyata Tarun	1. Explain about Tomcat web server.
			2. What is Servlet? Explain life cycle of a Servlet?

54	21R95A1205	Ramavath Venkatesh	1. What are the advantages of Servlets over CGI
			2. What is session tracking? Explain different mechanisms of session tracking?
55	21R95A1207	Vavilla Sandeep	1. What is the difference between Servlets and applets?
			2. What is the difference between doGet() and doPost().

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Define a session tracker that tracks the number of accesses and last access data of a particular web page.	2.5	L2	3
2	What is the security issues related to Servlets.	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Explain how HTTP POST request is processed using Servlets	2.5	L2	3
2	Explain how cookies are used for session tracking?	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Explain about Tomcat web server.	2.5	L2	3
2	What is Servlet? Explain life cycle of a Servlet?	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	What are the advantages of Servlets over CGI	2.5	L2	3
2	What is session tracking? Explain different mechanisms of session tracking?	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	What is the difference between Servlets and applets?	2.5	L2	3
2	What is the difference between doGet() and doPost().	2.5	L3	3

Assignment II

S.No	Roll Number	Student Name	Assignment Question
1	20R91A1201	Aanchal Thakur	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.

			2. List and explain the classes and interfaces of javax.servlet.http package.
2	20R91A1202	Akavaram Tejaswini	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
3	20R91A1203	Meghana. Akula	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
4	20R91A1204	Bingi Sathwika	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
5	20R91A1205	Barla Saikiranreddy	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
6	20R91A1206	Bejjenki Chaithanya	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
7	20R91A1207	Bhukya Vishnuvardhan	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
8	20R91A1208	C. Ankitha	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
9	20R91A1209	Ch. Akshay Kumar	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
10	20R91A1210	Chintada Bhavya Siva Saik	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
11	20R91A1211	Chelimela Keerthana	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
12	20R91A1212	Chillara. Mahesh	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
13	20R91A12	D. Aadityaa	1. "Servlet offer several advantages over CGI".

	13		Justify.
			2. Explain about Security Issues in Servlet
14	20R91A1214	D. Maheshwar Naidu	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
15	20R91A1215	Dharavath Vishnuvardhan	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
16	20R91A1216	Erla Eswar Prasad	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
17	20R91A1217	Enukonda Harshavardhan Re	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
18	20R91A1218	Gaddam Avinash	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
19	20R91A1219	Godala Mahitha Reddy	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
20	20R91A1220	G. Raveena	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
21	20R91A1221	Inapanuri Kavya	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
22	20R91A1222	Jaya Vardhan	3. Build a Servlet that handles HTTP get Request
			4. Describe about session tracking with relevant code snippet.
23	20R91A1223	Jolge Ajay	5. "Servlet offer several advantages over CGI". Justify.
			6. Explain about Security Issues in Servlet
24	20R91A1224	Kuchipudi Sai Kumar	7. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			8. Build a Servlet program to illustrate parameter reading and parameter initializing.
25	20R91A1225	K Niharika Reddy	9. Explain Cookies session tracking with relevant code snippet.

			10. List the methods defined in HttpServletRequest.
26	20R91A1226	Kurakula Rahul	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
27	20R91A1227	Kusuma Vishwesh	3. Build a Servlet that handles HTTP get Request
			4. Describe about session tracking with relevant code snippet.
28	20R91A1228	Neela Sunil	5. "Servlet offer several advantages over CGI". Justify.
			6. Explain about Security Issues in Servlet
29	20R91A1229	Mamidi. Sainikhitha	7. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			8. Build a Servlet program to illustrate parameter reading and parameter initializing.
30	20R91A1230	Maroju Sathish	9. Explain Cookies session tracking with relevant code snippet.
			10. List the methods defined in HttpServletRequest.
31	20R91A1231	Midde Varun Kumar	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
32	20R91A1232	Mohammed Abdhur Rahman	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
33	20R91A1233	Mohammed Muzammil Hussain	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
34	20R91A1234	Nagamalla Rohith Kumar	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
35	20R91A1235	Nallala Krishna Chaitanya	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
36	20R91A1236	Nara Taruni	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
37	20R91A1237	Neerati Uday Kumar	1. Build a Servlet that handles HTTP get Request

			2. Describe about session tracking with relevant code snippet.
38	20R91A1238	Nethisathvika	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
39	20R91A1239	Oruganti Vinod	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
40	20R91A1240	Polasa Vyshnavi	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
41	20R91A1241	Pundra Raga Sree Reddy	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
42	20R91A1242	Racharla Naresh	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
43	20R91A1243	S. Nivas	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
44	20R91A1244	Singireddy Sindhu	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
45	20R91A1245	Sirandasu Sairaj	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
46	20R91A1246	V. Sampath Kumar	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
47	20R91A1247	Yash Wasnik	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
48	20R91A1248	Yanagandhula Varun	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
49	20R91A12	Manee Jayanth	1. Explain about Servlet? Explain lifecycle of a

	49	Kumar	Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
50	21R95A1201	B Manohar Reddy	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.
51	21R95A1202	Kothapally Sai Sumanth	1. Build a Servlet that generates HTML page and explain the process of generation of HTML page.
			2. List and explain the classes and interfaces of javax.servlet.http package.
52	21R95A1203	P Sri Harsha Kumar	1. Build a Servlet that handles HTTP get Request
			2. Describe about session tracking with relevant code snippet.
53	21R95A1204	Pyata Tarun	1. "Servlet offer several advantages over CGI". Justify.
			2. Explain about Security Issues in Servlet
54	21R95A1205	Ramavath Venkatesh	1. Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.
			2. Build a Servlet program to illustrate parameter reading and parameter initializing.
55	21R95A1207	Vavilla Sandeep	1. Explain Cookies session tracking with relevant code snippet.
			2. List the methods defined in HttpServletRequest.

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Build a Servlet that generates HTML page and explain the process of generation of HTML page.	2.5	L4	3
2	List and explain the classes and interfaces of	2.5	L3	3

	javax.servlet.http package.			

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Build a Servlet that handles HTTP get Request	2.5	L2	3
2	Describe about session tracking with relevant code snippet.	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	"Servlet offer several advantages over CGI". Justify.	2.5	L2	3
2	Explain about Security Issues in Servlet	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Explain about Servlet? Explain lifecycle of a Servlet. Illustrate with an example program.	2.5	L2	3
2	Build a Servlet program to illustrate parameter reading and parameter initializing.	2.5	L3	3

Q.No	Questions	Marks	Bloom's Level	CO Map
1	Explain Cookies session tracking with relevant code snippet.	2.5	L2	3

2	List the methods defined in HttpServletRequest.	2.5	L3	3

UNIT TEST1

Q.No	Questions	Marks	Bloom's Level	CO Map
1	How do you create an ordered and unordered list? Give examples. Explain the difference between HTML attributes and elements. Describe the and tags and their uses. How can you add comments in HTML?	10	4	1
2	Explain the use of arithmetic, comparison, and logical operators in PHP. What are conditional statements in PHP, and give examples of if, else, and switch statements. How do you write a for loop and a while loop in PHP?	10	4	1

UNIT TEST2

Q.No	Questions	Marks	Bloom's Level	CO Map

1	How are XML elements and tags defined? What are attributes in XML, and how are they used? Explain the structure of an XML document with opening and closing tags. Provide an example of an XML element with attributes and values.	10	4	2
2	How can you parse an XML document using the DOM parser in Java? Provide an example of traversing and manipulating XML data using DOM.How does SAX parsing handle large XML documents?	10	4	2

UNIT TEST3

Q.No	Questions	Marks	Bloom's Level	CO Map

1	Describe the life cycle phases of a servlet. What methods are invoked during the initialization and destruction phases of a servlet's life cycle?	10	3	3
2	Describe the key methods in the HttpServletRequest and HttpServletResponse classes. How can servlets set HTTP response headers and content types?Explain the purpose of HTTP status codes in servlet responses.	10	4	3

UNIT TEST4

Q.No	Questions	Marks	Bloom's Level	CO Map
1	What are JavaBeans, and how can they be used in JSP pages? Describe the steps to use a JavaBean in a JSP page. Explain the advantages of using JavaBeans for separating logic and presentation.	10	3	4

2	How can you establish a database connection in a JSP page? Explain the security implications of connecting directly to a database in JSP.	10	4	4

UNIT TEST5

Q.No	Questions	Marks	Bloom's Level	CO Map
1	What are event handlers in JavaScript, and how do they facilitate interactivity? Provide examples of commonly used event handlers, such as onclick, onsubmit, and onmouseover.	10	4	5
2	Provide examples of form validation techniques, such as checking required fields and valid email addresses. How can you display error messages to the user during form validation?	10	4	5

Laboratory records(if any)

Students whose academic performance is not satisfactory

S. No	Roll No	Student Name	Remedial measures taken by teacher

Teacher self assessment (at the completion of course)

All five units are completed and 80% of students understood the subject with the help of lab practical's students and the various examples given in the classroom.

UNIT-I

PHP

PHP:

The PHP Hypertext PreProcessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

* PHP is a recursive acronym for "PHP: Hypertext PreProcessor".

* PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

Common Uses Of PHP:

* PHP performs system functions, i.e., from files on a system it can create, open, read, write and close them.

* PHP can handle forms i.e., gather data from files, save data to a file, through email you can send data, return data to the user.

* You can add, delete, modify elements within your database through PHP.

* Access cookies variables and set cookies.

* Using PHP, you can restrict users to access some pages of your website.

* It can encrypt data.

Advantages of PHP:

* PHP runs on various platforms (windows, Linux, Unix, Mac osx, etc..)
* PHP is compatible with almost all servers today (Apache, IIS etc)
* PHP supports a wide range of databases
* PHP is free.
* PHP is easy to learn and it runs effeciently on the server side

Syntax for PHP:

```
<? php
  - - -
  - - -
?>
```

* DECLARING VARIABLES:

In PHP, a variable starts with the '$' sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

* After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

PHP Variables:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

* A variable starts with a $ sign, followed by the name of the variable.
* A variable name must start with a letter or the underscore character.
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-Z, 0-9 and _).
* Variable names are case-sensitive ($age and $AGE are two different variables).

PHP is a Loosely Typed Language :

In the example above, notice that we did not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value. Other languages such as C, C++ and Java, the programmer must declare the name and type of the variable before using it.

PHP Variable Scope :

In PHP, variables can be declared anywhere in the script.

* The scope of the variable is the part of the part of the script where the variable can be referenced / used.

* PHP has three different variable scope.

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
$x=5; // global scople
function myTest()
{
    // using x inside this function will not generate an error
    echo "<p> Variable x inside function is : $x </p>";
}
```

```
myTest();
echo "<p> Variable x outside function is : $x </p>";
?>
```

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function :

```
<?php
function myTest()
{
    $x = 5; // local scope
    echo "<p> Variable x inside function is : $x </p>";
}
myTest();
// using x outside the function will generate an error
echo "<p> Variable x outside function is : $x </p>";
?>
```

* PHP DATA TYPES :

Variables can store data of different types, and different data types can do different things.

* PHP supports the following data types :

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

1) PHP string :

A string is a sequence of characters, like "Hello world!".

* A string can be any text inside quotes. You can use single or double quotes.

Example :

```
<?php
$x = "Hello world!"
$y = 'Hello world!';

    echo $x;
    echo "<br>";
    echo $y;
?>
```

Output :

```
Hello world!
Hello world!
```

2) PHP Integer:

An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.

Rules for integers:

* An integer must have atleast one digit (0-9)
* An integer cannot contain comma or blanks
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based prefix with 0x) or octal (8-based prefix with 0)

Example:

```
<?php
$x = 5985;
var_dump($x);
?>
```

Output:

```
int (5985)
```

3) PHP Float:

A float (floating point number) is a number with decimal point or a number in exponential form. In the following example $x is a float. The PHP var_dump() function returns the datatype and value.

Example:

```
<?php
$x = 10.365;
var_dump ($x);
?>
```

Output

```
float (10.365)
```

4) PHP Boolean:

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

5) PHP Array:

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the datatype and value:

Example:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
    var_dump($cars);
?>
```

Output:

```
[0] => string(5) "volvo"
[1] => string(3) "BMW"
[2] => string(6) "Toyota"
```

6) PHP object:

An object is a data type which stores data and information on how to process that data.

* In PHP, an object must be explicitly declared.

* First we must declare class of object. For this, we use a class keyword. A class is a structure that can obtain properties and methods.

Example :

```
<?php
class car
{
    function car()
    {
        $this -> model = " VW";
    }
}
// Create an object
  $x = new car();
// Show object properties
  echo $x -> model ;
?>
```

Output:

VW

7) PHP NULL value :

NULL is a special data type which can have only one value : NULL.

* A variable of datatype NULL is a variable that has no value assigned to it.

Note: If a variable is created without a value, it is automatically assigned a value of NULL.

* Variables can be emptied by setting a value to NULL

Example:

```
<? php
  $x = "Hello world!";
  $x = null
  var_dump ($x);
  ?>
```

Output :

NULL

8) PHP Resource :

The special resource type is not an actual data type. It is a storing of the reference and resources external to PHP.

* A common example of using the resource datatype is a database call.

* PHP Array

An array is a special variable, which can hold more than one value at a time.

* There are three different kinds of arrays and each array value is accessed using an ID which is called array index.

Numeric array:

An array with a numeric index. Values are stored and accessed in linear fashion.

Associative array:

An array with strings as index. This stores element values in association with key values in association rather than in a strict linear index order.

Multidimensional array:

An array containing one or more arrays and values are accessed using multiple indices.

Numeric Array:

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example:

Following is the example showing how to create and access numeric arrays.

Here we have used array() function to create array. This function is explained in function reference.

```
<? php
  $numbers = array(1,2,3,4,5);
  foreach ($numbers as $value)
  {
      echo "Value is $value <br/>";
  }
?>
```

This will produce the following result –

```
Value is 1
value is 2
value is 3
value is 4
value is 5
```

Associative Arrays:

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between keys and values.

* To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE: Don't keep associative array inside double quote while printing otherwise it would not return any value.

```
<? php
$salaries = array ("Rohit" => 2000, " peter" => 1000,
                    " kelvin" => 500);
    echo "salary of Rohit is". $salaries['Rohit'].
                                "<br/>";
    echo "salary of peter is". $salaries['peter'].
                                "<br/>";
    echo "salary of kelvin is". $salaries ['kelvin'].
                                "<br/>";
?>
```

This will produce the following result –

Salary of Rohit is 2000
salary of peter is 1000
salary of kelvin is 500

Therefore from above example we came to know that
[Salary of Rohit is high
salary of peter is medium
salary of kelvin is low.]

Multidimensional Arrays:

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example :

In this example we create a two dimensional array to store marks of three students in three subjects – This example is an associative array, you can create array in the same fashion.

```
<?php
$marks = array (" Rohit" => array ("physics" => 35,
                "maths" => 30, "chemistry" => 39 ),
                " peter" => array ("physics" => 30,
                "maths" => 32 , "chemistry" => 29),
                " kelvin" => array ("physics" => 31,
                "maths" => 22 , "chemistry" => 39));

/* Accessing multi-dimensional array values*/
echo "Marks for Rohit in physics: ";
echo $marks ['Rohit'] ['physics'] . "<br/>";

echo " Marks for peter in maths : ";
echo $marks ['peter'] ['maths'] . "<br/>";

echo " Marks for kelvin in chemistry : ";
echo $marks ['kelvin'] ['chemistry'] . " <br/>";

?>
```

This will produce the following result –

Marks for Rohit in physics : 35
Marks for peter in maths : 32
Marks for kelvin in chemistry : 39

* STRINGS

A string is a sequence of characters, like "Hello world!".

Get the Length of a String:

The PHP strlength i.e., strlen() function returns the length of a string.

* The example below returns the length of the string "Hello world!";

```
<?php
    echo strlen("Hello world!"); // outputs 12
?>
```

Count the Number of Words in a String:

The PHP str_word_count() function counts the number of words in a string:

```
<?php
    echo str_word_count("Hello world!"); // outputs 2
?>
```

The output of the code above will be: 2.

String Concatenation Operator:

To concatenate two string variables together, we use dot (.) operator -

```
<?php
$string1 = "Hello world";
$string2 = "1234";
echo $string1." ".$string2;
?>
```

This will produce the following result -

Hello world 1234

Reverse a String:

The PHP strrev() function reverses a string:

```
<?php
   echo strrev("Hello world"); //outputs dlrowolleH
?>
```

Using the strpos() function

The strpos() function is used to search for a string or character within a string.

* If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

* Let's see if we can find the string "world" in our string -

```
<?php
   echo strpos("Hello World!", "world");
?>
```

This will produce the following result — 6

In PHP we can assign values into the string variables in 3 ways.

→ using single quotation
→ using double quotation
→ heradoc style

* In PHP, we have approximately 100 functions.
* we introduce each function but we have to implement some of the functions of a string.

1) determining string length
2) comparing two strings
3) manipulating string case
4) alternatives for regular expression functions
5) converting string to and form HTML.
6) padding and stripping a string
7) counting characters and works.

1) determining string length:
here we are using strlen() function and this function returns the length of the string.

Eg: int strlen(string str)

2) Comparing two strings:
php provides four functions for performing this task.

a) strcmp( )
b) strcasecmp( )
c) strlen( ) strspn( )
d) strcmp strcspn( )

a) <u>strcmp( )</u> :

the strcmp( ) function performs a binary numbers and compares two strings.

<u>Syntax:</u> (case - sensitive)

int strcmp (string str1, string str2)

→ 0, if str1 and str2 are equal (s1==s2)

→ -1, if str1 is less than str2 (s1<s2)

→ 1, if str2 is less than str1 (s2<s1)

<u>Example:</u>

```
<? php
  $pwd = "abitcse";
  $pwd2 = "abitcse2";
  if (strcmp($pwd, $pwd2)! = 0)
  {
     echo "pwd do not match";
  }
  else
  {
     echo "pwd match";
  }
?>
```

b) stricasecmp():

The stricasecmp() function operates exactly like stricmp() but it is case-insensitive.

Syntax:

int stricasecmp(string str1, string str2)

Example:

```
<?php
    $gmail1 = "abit@gmail.com";
    $gmail2 = "ABIT@gmail.com";
    if (!stricasecmp($gmail1, gmail2)
        echo "the gmail addresses are identical";
?>
```

c) strspn():

Calculating the similarity between two strings.

syntax:

int strspn(string str1, string str2[, int start, int length)

Example:

```
<?php
    $pwd = "abc123";
    if (strspn($pwd, "1234567890") == strlen($pwd))
        echo "the pwd cannot consist solely of numbers!";
?>
```

d) strcspn():

calculating difference between two strings.

Syntax:

int strcspn(string str1, string str2, int start, int length)

Example:

```
<?php
  $pwd = "abc123";
if (strcspn ($pwd, "1234567890")==0)
{
   echo "pwd can't consist solely of numbers!";
}
?>
```

3) manipulating string case:

In this we have mainly four functions

a) strtolower()

b) strtoupper()

c) ucfirst()

d) ucwords()

a) strtolower():

converting a string to lower case.

Syntax:

strtolower (string)

Output:

bangaram

Example:

```
<?php
  $name = "Bangaram";
  echo str_tolower ($name);
?>
```

b) stritoupper( ):

converting a string to uppercase

Syntax:

stritoupper (string)

Example:

```
<?php
  $name = " John";
  echo stritoupper ($name);
?>
```

Output:

JOHN

c) ucfirst( ):

converts the first character of a string to uppercase

Syntax:

ucfirst (string)

Example:

```
<?php
  $name = "gopi";
  ~~echo strucfirst (string~~
  echo strucfirst ($name);
?>
```

Output:

Gopi

d) ucwords( ):

converts the first character of each word in a string to uppercase.

Syntax:

ucwords (string)

Example:

```
<?php
  echo ucwords("hello world!");
?>
```

output:

Hello World!

4) Alternativies for regular expression function:

In this we have to describe different types of functions. They are:

a) strtok()
b) explode()
c) implode()
d) strops()
e) strpos()
f) str_replace()
g) strstr()
h) substr()
i) substr_count()
j) substr_replace()

a) strtok():

this function parses the string based on a predefined list of characters.

Syntax:

string strtok(string str, string tokens)

Example:

```
<?php
$into = "abit : abit@gmail.com\siddavaram, kdp";
$tokens = ":\,";
$tokenized = strtok($into, $tokens);
while ($tokenized)
{
    echo "elements= $tokenized <br>";
```

```
$tokenized = strtok($tokens);
}
?>
```

b) explode();

this function devides the string str into an array of substrings, in this we have to mainly concentrate on areas: size of() and strip-tags() to determine the total no. of words.

Example:

```
<?php
$summary == <<< summary
```

php is a server side scripting language.

```
summary;
$words = size of (explode (' ', strip-tags ($summary)));
echo "total words in summary : $words";
?>
```

c) Implode()

we concentrate array elements to form a single ~~defined~~ delimited string using the implode() function.

Example:

```
<?php
$cities = array("kdp", "anlpur", "tirupathi");
echo implode("\", $cities);
?>
```

d) strops():

In this function finds the position of the first case-sensitive occurance of a substring in a string.

e) strrpos():

in this function finds the last occurance of a string returning its numerical position.

f) str_replace():

this function case sensitively replaces all instance of a string with another.

Example:

```
<?php
  $gmail = "abit@gmail.com";
  $gmail = str_replace("@", "(is)", $gmail);
     echo "college mail is $gmail";

?>
```

g) strstr:

this function returns the remainder of a string begining with the first occurance of a predefined string.

Example:

```
<?php
  $gmail= "abit@gmail.com";
  echo ltrim (strstr ($gmail, "@"), "@");

?>
```

h) substr():

this function returns the part of a string located between a predefined string offset and length positions.

Eg:
```
<?php
$car = "1994ford";
echo substr ($car5);
?>
```

Eg 2:
```
<?php
$car = "1994 ford";
echo substr($car, 0, 4);
?>
```

i) substr_count():

this function returns the no. of times one string excuss another.

Example:
```
<?php
$intu= array("php", "xAMPP");
$talk = <<< talk
PHP is a scripting language and php is server side
programming language. xAMPP is a web server
Talk:

foreach($info as $it)
{
    echo "the word $it appears". substr_count
                        ($talk, $it). "time (s)<br/>";
}
?>
```

j) substr_replace():

replace the portion of a string with another string.

Example:

```
<?php
$name = "Abit college";
echo substr_replace ($name, "engg", 0,4);
?>
```

5) Converting string to HTML form:

In this we are using different types of converting function. they are

* Converting newline characters to HTML break tags.

Example:

```
<?php
$info = " aaaaaaaaaaaaaaa
          Bbbbbbbbbbbbbbb
          ccccccccccccccc
          dddddddddddddddd ";
echo n12br1( $into);
?>
```

Here we do not use <br> statement.

* Using special HTML characters for other purpose.

In this we use htmlspecialchars() function.

- & — &
- " — "
- ' —
- < — <
- > — >

Example:

```
<?php
$input = "<php is "scripting" language>";
echo htmlspecialchars ($input);
?>
```

6) padding and stripping a string:

php provides no. of functions. They are

a) ltrim()

b) rtrim()

c) trim()

d) str_pad()

a) ltrim():

this function removes various characters from the begining of a string including white space, horizontal tab (\t), newline (\n), carriage return (\v), null (\0).

~~String ltrim (string str, char~~

Syntax:

ltrim (string, charlist)

b) rtrim():

this function removes various characters from the end of the string and except designed characters.

String rtrim (string str [, string charlist])

c) trim():

both ltrim and rtrim.

d) str_pad():

this function pads a string with a specified number of characters

Example:
```
<?php
    Echo str_pad ("salad", 10). "is good";
?>
```

Output:

Salad is good.

7) Counting characters and words:

its mainly used for to determine the total number of characters or words in a given string. Php provides two functions.
there are count_chars() and str_word_count.

* PHP Operators :

Operators are used to perform operations on variables and values.

* PHP language supports following type of operators

a) Arithmetic operators
b) Assignment operators
c) Comparision operators
d) Increment / Decrement Operators
e) Logical Operators
f) Array operators

a) Arithmetic Operators:

The PHP arithmetic operators are used with numeric values to perform common arithmetic operations such as addition, subtraction, multiplication etc..

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y^th power.

b) Assignment Operators:

The PHP assignment operators are used with numeric values to write a value to a variable.

* The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression to the right.

Assignment	Same as..	Description
$x=y$	$x=y$	The left operand gets set to the value of the expression on the right.
$x+=y$	$x=x+y$	Addition
$x-=y$	$x=x-y$	subtraction
$x*=y$	$x=x*y$	Multiplication
$x/=y$	$x=x/y$	Division
$x\%=y$	$x=x\%y$	Modulus

c) Comparision Operators:

The PHP comparision operators used to compare two values (number or string).

Operator	Name	Example	Result
==	Equal	$x==$y	returns true if $x is equal to $y
===	Identical	$x===$y	returns true if $x is equal to $y, and they are of same type.
!=	not equal	$x!=$y	returns true if $x is not equal to $y

Operator	Name	Example	Result
<>	not equal	$x <> $y	returns true if $x is not equal to $y
!==	not identical	$x !== $y	returns true if $x is not equal to $y, or they are not of same type
>	Greater than	$x > $y	returns true if $x is greater than $y
<	less than	$x < $y	returns true if $x is less than $y
>=	greaterthan or equal to	$x >= $y	returns true if $x is greater than or equal to $y
<=	less than or equal to	$x <= $y	returns true if $x is less than or equal to $y.

d) Increment / Decrement Operators:

The PHP increment operators are used to increment a variables value

* The PHP decrement operators are used to decrement a variables value

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one.
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one.

e) Logical Operators:

The PHP logical operators are used to combine conditional statements

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	! $x	True if $x is not true.

f) Array Operators:

PHP array operators are used to compare arrays

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	returns true if $x and $y have same key/value pairs
===	Identity	$x === $y	returns true if $x and $y have the same key/value pairs in the same order and of the same types.
!=	Inequality	$x != $y	returns true if $x is not equal to $y
<>	Inequality	$x <> $y	returns true if $x is not equal to $y
!==	Non-Identity	$x !== $y	returns true if $x is not identical to $y

* CONTROL STRUCTURES

php supports different types of statements like

1) if statement
2) else statement
3) switch statement
4) while statement
5) do...while statement
6) for statement
7) for each statement
8) break and goto statement.
9) continue statement

1) if statement :

Use the if statement to execute some code only if the specified condition is true.

Syntax :

```
if (expression)
{
    statement
}
```

Example :

```
<?php
    $secretnumber = 143;
    if ($_POST['guess'] == $secretnumber)
    {
        echo "<p> Congratulation! </p>";
    }
?>
```

2) else statement:

if the condition is true then if statements are executed otherwise else statements will be executed.

Example:

```
<?php
$secretnumber = 143;
if ($_POST['guess'] == $secretnumber)
{
    echo "<p> Congratulation!</p>";
}
else
{
    echo "<p> Sorry! </p>";
}
?>
```

3) switch statement:

Use the switch statement to select one of many blocks of code to be executed.

* switch statement can compare "=" operations only.

Example:

```
<?php
    $x=1;
    switch ($x)
    {
        case 1: echo "number1";
                break;
```

```
    case 2: echo "number2";
            break;

    case 3: echo "number3";
            break;

    Default:
            echo "no number b/w 1 and 3";
  }
?>
```

4) while Statement :

while loop checks the condition then only executes the statements if condition is true.

Syntax :

```
while (expression)
{
    statements
}
```

Example:

```
<?php
  $count=1;
  while ($count<5)
  {
      printf("%dsquared = %d <br>", $count,
                                  pow($count,2));
      $count++;
  }
?>
```

Output :

```
1 squared=1
2 squared=4
3 squared=9
4 squared=16
```

5) do... while statement :

It will execute the statement atleast once even if condition is false (or) true.

Syntax :

```
do
{
    statements

}while (expression);
```

Example :

```
<?php
   $count =11;
   do
   {
      printf (" %d squared = %d <br/> ", $count,
                                    pow($count,2));
   }
   while ($count <10);

?>
```

6) for statement :

By using this loop we can run number of iteration

Syntax :

```
for (exp1; exp2; exp3)
{
    statements;

}
```

There are few rules to keep in mind when using php's for loops

- The first expression exp1, is evaluated by default at the first iteration of the loop.
- The second expression exp2, is evaluated at the begining of each iteration. This expression determines whether loop will continue.
- The third expression exp3, is evaluated at conclusion of even loop.

Example:

```
<?php
  for ( $kilometers =1; $ kilometers <=3; $kilometers ++)
  {
     printf (" % kilometers = %.f miles <br/> ",
                        $kilometers, $kilometers * 0.62140);
  }
?>
```

Output:

1 kilometers = 0.6214 miles

2 kilometers = 1.2428 miles

3 kilometers = 1.8642 miles.

7) for each statement:

loops through a block of code for each element in an array.

* The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax:

```
foreach ($array as $value)
{
    code to be executed;
}
```

Example:

```
<?php
  $colors = array ("red", "blue", "white", "black");
  foreach ($colors as $value)
  {
      echo "$value <br>";
  }
?>
```

Output:

red
blue
white
black

8) break and goto statements:

break statement: break statement is end execution of a do while, for, foreach, switch, while block.

goto statement: In php goto statement, "BREAK" features was extend to support labels. This means we can suddenly jump to a specific location outside of a looping or conditional construct.

9) **Continue statement**:

continue statement execute the current loop iteration to the end.

* PHP Functions:

A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page loads. A function will be executed by a call to the function.

Create a User Defined Function in PHP.

A user defined function declaration starts with a word "function".

Syntax:

```
function functionName()
{
    code to be executed;
}
```

Note: A function name can start with a letter or underscore (not a number)

* Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the begining of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello World!". To call the function, just write its name:

Example :

```
<? php
  function writeMsg( )
  {
      echo "Hello World!";
  }
  writeMsg( ); // call the function
?>
```

PHP Function Arguments :

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just seperate them with a comma.

* The following example has a function with one argument ($fname). When the familyName( ) function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name.

Example :

```
< ? php
function familyName($fname)
{
    echo "$fname Refsnes. <br>";
}
```

```
familyName ("Jani");
familyName ("Hege");
familyName ("Stale");
familyName ("Kai Jim");
familyName ("Borge");
?>
```

PHP Default Argument Value :

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument.

Example :

```
<?php
    function setHeight ($minheight = 50)
    {
        echo "The height is : $minheight <br>";
    }
    setHeight (350);
    setHeight ( ); // will use the default value of 50
    setHeight (135);
    setHeight (80);
?>
```

Output :

The height is : 350
The height is : 50
The height is : 135
The height is : 80

PHP Functions – Returning Values

To let a function return a value, use the return statement.

Example:

```
<? php
  function sum($x,$y)
  {
      $z=$x+$y;
      return $z;
  }
  echo "5+10 =" . sum (5,10). "<br>";
  echo "7+13 =" . sum (7,13). "<br>";
  echo "2+4 =" . sum (2,4). "<br>";
?>
```

Output:

5+10 = 15
7+13 = 20
2+4 = 6

Array Functions:

* **Count**: it returns total no. of elements

Example :

```
<?php
    $arr = array(10,20,30);
    echo count($arr);
?>
```

Output : 3

* **Sort**: it returns the elements of an array in ascending order.

Example :

```
<?php
    $arr = array(60,20,30)
    sort($arr);
    print_r($arr);
?>
```

output : 20, 30, 60

* **rsort**: it returns the elements of an array in descending order.

Example :

```
<?php
    $arr = array(101,104,102);
    rsort_r($arr);
    print_r($arr);
?>
```

output : 104, 102, 101

* asort: it returns the original keys with ascending order

Example:
```
<?php
$arr = array(104 => 40, 101 => 20, 108 => 50);
asort ($arr);
print_r($arr);
?>
```

Output: 101 => 20, 104 = 40, 108 = 50

* ksort: it returns the array in ascending order with based on the "keys".

Example:
```
<?php
$arr = array (104 => 40, 101 => 20, 108 => 50)
ksort($arr);
print_r ($arr);
?>
```

Output: [101] = 20, [104] = 40, [108] = 50

* array push(): this function adds an element into the end of the array and returns the total no. of elements in that array.

Example:
```
<?php
$arr = array (10, 20, 30)
echo array_push ($arr, 40);
print_r ($arr);
?>
```

Output:
```
10
20
30
40
```

* array pop() : remove the last element and return the value to that element.

Example :
```
<?php
$arr = array(10, 20, 30);
echo array_pop($arr);
print_r($arr);
?>
```

* array_change_key_case() : it converts all keys of an array into lower case.

Example :
```
<?php
$arr = array('ABC' => 10, 20, 30);
print_r(array_change_key_case($arr));
?>
```

* array_chunk(): splits an array into chunck of an array.

Example :
```
<?php
$arr = array(10, 20, 30, 40, 50, 60);
print_r(array_chunk($arr, 2));
?>
```

* array_keys() : it returns new array with keys as value of another array.

Example :
```
<?php
$arr = array('abc' => 10, 20, 30, 40);
print_r(array_keys($arr));
?>
```

* array-values() : return array with the values of an array.

Example :
```
<?php
$arr = array('ABC' = 10,20,30,40,50,60)
print-r (array-values($arr));
?>
```

* array-flip() : exchanges all keys with their associated values in array.

Example :
```
<?php
$arr = array('ABC' -> 10,20,30,40);
// $arr = array (10, 200,400);
printf-r (array-flip ($arr));
?>
```

* array-merge() : merges one or more arrays into one array.

Example :
```
<?php
$arr = array ('ABC' => 10,20,30,40);
$arr1 = array (100,200, 300);
printf-r (array-merge ($arr, $arr1));
?>
```

* Shuffle() : shuffle the elements of an array

Example :
```
<?php
$arr = array('ABC' => 10,20,30,40);
Shuffle ($arr);
print-r ($arr);
?>
```

* READING DATA FROM WEB FORM

$_REQUEST :

It is used to collect data after submitting an HTML form

* when user submits the data by clicking on "submit", the form data is sent to the file specified in the action attribute of the <form> tag.

* In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field.

Example:

```
<html>
<body>
<form method="post" action="<?php
            echo $_SERVER['PHP_SELF']; ?>">
Name ; <input type="text" name="fname">
<input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"]== "POST")
{
    $name = htmlspecialchars($_REQUEST['fname']);
```

```
    if (empty ($name))
    {
        echo "Name is empty";
    }
    else
    {
        echo " $name ;
    }
}
?>
</body>
</html>
```

$_POST :

It is widely used to collect form data after submitting an HTML form with method = "POST". $_POST is also widely used to pass variables

$_GET :

This can also be used to collect form data after submiting an HTML form with method = "get".

* $_GET can also collect data sent in the URL.

* HANDLING FILE UPLOADS:

It can be used to upload the files.

a) name
b) type
c) tmp_name
d) error
e) size.

Example:

Uploading file to server.

```
<html>
<body>
<form action = "upload.php" method = "post">
<input type = "file" name = "file1">
<input type = "submit" name = "submit" value =
                                    "upload">
</form>
</body>
</html>

<?php
$file_name = $_FILES ['file1']['name'];
$file_type = $_FILES ['file1']['type'];
$file_path = $_FILES ['file1']['tmp_name'];
$file_size = $_FILES ['file1']['size'];
$file_error = $_FILES ['file1']['error'];
$file_loc = "upload /". $file_name ;
```

?>

Note :

mov_uploaded_file

↓

This is the function which uploads your file from client to server

* CONNECTING TO DATABASE

With PHP, you can connect to and manipulate databases.

* MySQL is the most popular database system used with PHP

What is MySQL ?

* MySQL is a database system that runs on a server.
* MySQL is very fast, reliable and easy to use.
* MySQL compiles on a number of platforms
* MySQL is developed, distributed and supported by oracle corporation

PHP connecting to MySQL

1) mysqli
2) PDO (PHP Data Objects)

Establishing a connection

Object-oriented :

```
<?php
$conn = new mysqli ("localhost", "userid", "password");
if ($conn -> connect_error)
{
    die ("connection failed :". $conn -> connect_error);
}
else
{
    echo "connection is established";
}
?>
```

Procedure - Oriented :

```
<?php
$conn=mysqli_connect("localhost", "userid",
                                  "password");
if (!$conn)
{
   die ("connection failed", mysqli_connect_error());
}
else
{
    echo " " ;
}
?>
```

Closing the connection :

Object - oriented

```
$conn -> close();
```

Procedure - oriented

```
mysqli_close ($conn);
```

Execution of Query:

Object Oriented

```
<?php
$sql= "select * from emp";
if ( $conn -> query ($sql) == TRUE)
{
    echo "successful";
}
```

```
else
{
    echo "error: " . $conn -> error ;
}
?>
```

Procedure - Oriented :

```
<?php
$sql = "      ";
if (mysqli_query ($conn , $sql)
{
    echo " successful";
}
else
{
    echo " error : " mysqli_error ($conn);
}
?>
```

* EXECUTING SIMPLE QUERIES

Database Queries:

A Query is a question or a request

Create a MySQL Table.

The "create table" statement is used to create a table in mySQL

Example: create table student (sid int(10),
sname varchar(20) ....);

Insert Data into MySQL

After a database and a table have been created, we can start adding data in them.

* Here are some syntax rules to follow

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted.
- Numeric values must not be quoted.
- The word NULL must not be quoted.

The "insert into" statement is used to add new records to a MySQL table.

Example:

insert into student values (value1, value2, ...);

Creation of database:

Syntax:

```
create database databasename;
```

Example:

```
create database cse;
```

Select Data From a MySQL Database.

The "select" statement is used to select data from one or more tables:

```
select column_name(s) from table_name
```

or we can use the * character to select ALL columns from a table.

```
select * from table_name
```

Delete Data From MySQL

The "Delete" statement is used to delete records from a table:

```
delete from table_name where some_column =
                                some_value
```

Update Data in MySQL

The "update" statement is used to update existing records in a table

```
update table_name set column1 = value1, column2 =
                value2 where some_column=
                                some_value
```

* HANDLING RESULTS:

```
<?php
  $servername = "localhost";
  $username = "username";
  $password = "password";
  $dbname = "myDB";

// create connection
$conn = new mysqli($servername, $username,
                   $password, $dbname);
// check connection
if ($conn -> connect_error)
{
    die("connection failed:".$conn -> connect_error);
}
$sql = "SELECT id, firstname, lastname FROM
                                MyGuests";
$result = $conn -> query($sql);
if ($result -> num_rows > 0)
{
    //output data of each row
    while ($row = $result -> fetch_assoc())
    {
        echo "id:". $row["id"]. " - Name:".$row
              ["firstname"]." ".$row["lastname"]."<br>";
    }
}
else
{
    echo "0 results";
}
$conn -> close();
?>
```

* HANDLING SESSIONS AND COOKIES :

$_COOKIE :

* A cookie is often used to identify a user.
* cookies are stored at client side.
* Cookie takes less space to store data.
* It consists of name and a textual value.
* In every http communication between browser and server, a header is included.
* A header stores information about this message (user id's or names)

Create cookies with PHP.

A cookie is created with the setcookie() function.

Syntax :

```
setcookie (name, value, expire, path, domain,
                                  secure, httponly);
```

Only the name parameter is required. All other parameters are optional.

PHP create / Retrieve a cookie :

The following example creates a cookie named "user" with the value "John".

* The cookie will expire after 30 days (86400 x 30).

* The '/' means that the cookie is available in entire website.

* We then retrieve the value of the cookie "user" (using the global variable $_COOKIE)

* we also use isset() function to find out if the cookie is set.

Example :

```
<?php
$cookie_name = "user";
$cookie_value = "John";
setcookie ($cookie_name, $cookie_value,
                    time() + (86400 * 30), "/");

?>
<html>
<body>
<?php
if (!isset ($_COOKIE [$cookie_name]))
{
    echo ....

}
else
{
    echo ....

}
$_COOKIE [$cookie_name];

?>
</body>
</html>
```

$_SESSION:

A session is a way to store information (in variables) to be used across multiple pages.

* Unlike a cookie, the information is not stored on the users computer.

Start a PHP session:

A session is started with the session_start() function

Example:

" demo-session.php "

```
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["favcolor"] = "blue";
$_SESSION["favanimal"] = "dog";
echo "session variables are set.";
?>
</body>
</html>
```

Note:

The session_start() function must be the very first thing in your document. Before any html tags.

Destroy a PHP session:

To remove all global session variables and destroy the session, use session_unset() and session_destroy()

Example:

```
<?php
session_start();
?>
<html>
<body>
<?php
    session_unset();
    session_destroy();
?>
</body>
</html>
```

Chapter - 2

FILE HANDLING IN PHP

File Handling:

File Handling is an important part of web application.

Operations on the file:

1) open
2) read
3) write
4) close.

1) Open a file:

A better method to open files is within the fopen() function. This function takes two parameters

* The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

* The file may be opened in one of the following modes.

modes	description
r	open a file for read only
w	open a file for write only erases the contents of the file and file pointer starts at begining of the file
a	open a file for read only The existing data in the file is preserved. File pointer starts at the end of the file
x	creates a new file for write only. returns FALSE and an error if file already exists
r+	open a file for read/write.
w+	open a file for read/write
a+	open a file for read/write
x+	creates a new file for read/write

Example :
```
<?php
$myfile = fopen ("abc.txt", "w");
?>
```

2) read a file :

The fread() function reads from an open file.

* The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum no. of bytes to read.

3) writing a file:

The fwrite() function is used to write to a file.

* The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

Example:

```
<?php
$myfile = fopen("abc.txt", "w");
$txt = "Good morning\n";
fwrite ($myfile, $txt);
fclose ($myfile);
```

4) Close a file:

The fclose() function is used to close an open file.

Example:

```
<?php
$myfile = fopen("abc.txt", "r");
fclose ($myfile);
?>
```

Note:

* The fgets() function is used to read a single line from a file.

* The feof() function checks if the "end-of-file" (EOF) has been reached.

* The fgetc() function is used to read a single character from a file.

Example:

```
<?php
$myfile = fopen("abc.txt", "r");
while (!feof($myfile))
{
    $txt = fgets($myfile);
    echo $txt . '<BR>';
}
fclose($myfile);
?>
```

* Listing Directories :

Function	Description
chdir()	changes the current directory
chroot()	changes the root directory
closedir()	close a directory handle
dir()	returns an instance of the directory class
getcwd()	returns the current working directory
opendir()	opens a directory handle
readdir()	returns an entry from a directory handle
scandir()	returns an array of files and directories of a specified directory.

UNIT-II

XML

* INTRODUCTION TO XML

what is XML:

- XML stands for Extensible Markup Language.
- XML is a markup language much like HTML.
- XML was designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is designed to be self-descriptive.
- XML is W3C Recommendation.

Representing Web Data : XML

XML stands for eXentensible Markup Language, developed by W3C in 1996. XML 1.0 was officially adopted as a W3C recommendation in 1998. XML was designed to carry data, not to display data. XML is designed to be self-descriptive. XML is a subset of SGML than can be defined in your own tags. A meta Language and tags describe the content. XML supports CSS, XSL, DOM.

Advantages:

* XML is a simple scripting language whereas humans can easily read.
* XML document is language natural that means one language programming code can generate an XML document and these

documents can be passed by other languages.

Goals of XML:

* The user must be able to define and use his own tags.
* Allows the user to build his own tag library, based on his web requirement.
* Allow user to define the formatting rules for the user defined tags.
* XML must support storage or transport of data.

* DEFINING XML TAGS, THEIR ATTRIBUTES AND VALUES

Tags and Elements :

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. XML-elements names are enclosed by triangular brackets < > as shown below:

<element>

Syntax Rules for Tags and Elements

Element Syntax: Each XML-element needs to be closed either with start or with end elements as shown below:

<element> ... </element>

Or in simple-cases, just this way:

<element/>

Nesting of elements :

An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

Following example shows incorrect nested tags.

```
<?xml version = "1.0" ?>
<contact-info>
  <company> IARE
</contact-info>
  </company>
```

Following example shows correct nested tags:

```
<?xml version = "1.0" ?>
<contact-info>
<company> IARE </company>
</contact-info>
```

Let us learn about one of the most important part of XML, the XML tags. XML tags form the foundation of XML. They define the scope of an element in the XML. They can also be used to insert comments, declare settings required for parsing the environment and to insert special instructions.

* We can broadly categorize XML tags as follows:

<u>Start Tag</u>.

The begining of every non-empty XML element is marked by a start-tag. An example of start-tag is:

```
<address>
```

End Tag:

Every element that has a start tag should end with an end-tag. An example of end-tag is:

</address>

* Note that the end tags include a solidus ("/") before the name of an element.

Empty Tag:

The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as below:

(1) A start tag immediately followed by an end-tag as shown below:

<hr></hr>

(2) A complete empty-element tag is as shown below:

<hr/>

* Empty-element tags may be used for any element which has no content.

XML Tags Rules.

Following are the rules that need to be followed to use XML tags:

Rule 1:

XML tags are case-sensitive. Following line of code is an example of wrong syntax

</Address>,

because of the case difference in two tags, which is treated as erroneous syntax in XML.

<address> This is wrong syntax </Address>

Following code shows a correct way, where we use the same case to name the start and the end tag.

<address> This is correct syntax </address>

Rule 2:

XML tags must be closed in an appropriate order, i.e., an XML tag opened inside another element must be closed before the outer element is closed. For example:

```
<outer_element>
    <internal_element>
        This tag is closed before the outer_element
    </internal_element>
</outer_element>
```

XML elements

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.

* Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax:

Following is the syntax to write an XML element:

```
<element-name attribute1 attribute2>

    .... content

</element-name>
```

where,

- element-name is the name of the element. The name its case in the start and end tags must match

- attribute1, attribute2 are attributes of the element seperated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as:

name = "value"

The name is followed by an = sign and a string value inside double (" ") or single (' ') quotes.

Empty Element:

An empty element (element with no content) has following syntax:

```
<name attribute1 attribute2 .../>
```

Example of an XML document using various XML element:

```
<?xml version="1.0"?>
<contact-info>
<address category="residence">
<name> Tanmay Patil </name>
<company> TutorialsPoint </company>
<phone> (011) 123-4567 </phone>

<address/>
</contact-info>
```

XML Elements Rules:

Following rules are required to be followed for XML elements.

- An element name can contain any alphanumeric characters. The only punctuation marks allowed in names are the hyphen(-), underscore (_) and period (.).

- Names are case-sensitive. For example, Address, address and ADDRESS are different names.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements as seen in the above example.

Root element :

An XML document can have only one root element. For example, following is not a correct XML document, because both the x and y elements occur at the top level without a root element.

```
<x> ... </x>
<y> ... </y>
```

The following example shows a correctly formed XML document:

```
<root>
<x> ... </x>
<y> ... </y>
</root>
```

Document Type Definition (DTD):

DTD is an XML technique used to define the structure of a XML document.

* DTD is a text based document with the extension of .dtd.

* A DTD defines the structure and the legal elements and attributes of an XML document

DTD - XML Building Blocks:

The main building blocks of both XML and HTML documents are elements.

* Seen from a DTD point of view, all XML documents are made up by the following building blocks:

a) Elements
b) Attributes
c) Entities
d) PCDATA
e) CDATA

a) Elements:

Declaring Elements

In a DTD, XML elements are declared with the following syntax:

```
<!ELEMENT element-name category>
```

or

```
<!ELEMENT element-name (element-content)>
```

Elements with Parsed character Data

Elements with only parsed character data are declared with #PCDATA inside parentheses:

```
<!ELEMENT element-name (#PCDATA)>
```

Example :

```
<!ELEMENT student (#PCDATA)>
```

Elements with Children (sequences)

Elements with one or more children are declared with the name of the children elements inside parentheses:

```
<!ELEMENT element-name (child1)>
```

or

```
<!ELEMENT element-name (child1, child2..)>
```

Example :

```
<!ELEMENT student (branch, srno, name, marks)>

<!ELEMENT student (branch, srno, name, marks)>
<!ELEMENT branch (#PCDATA)>
<!ELEMENT srno (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
```

b) Attributes :

Declaring Attributes

```
<!ATTLIST element-name attribute-name
                attribute-type  attribute-value>
```

DTD example :

```
<!ATTLIST payment type CDATA "check">
```

XML example :

<payment type = "check" />

A Default Attribute Value

DTD : <! ELEMENT square EMPTY>
<! ATTLIST square width CDATA "0">

Valid XML : <square width = "100" />

#REQUIRED

DTD : <! ATTLIST person number CDATA #REQUIRED>

Valid XML : <person number = "5677" />

Invalid XML : <person />

#IMPLIED :

DTD : <! ATTLIST contact fax CDATA #IMPLIED>

Valid XML : <contact fax = "555-667788" />

Invalid XML : <contact />

#FIXED :

DTD : <! ATTLIST college name CDATA #FIXED "NNRG">

Valid XML : <college name = "NNRG" />

Invalid XML : <college name = "JNTUH" />

Enumerated Attribute Values

DTD : <! ATTLIST payment type (check | cash) "cash">

XML example : <payment type = "check" />

or <payment type = "cash" />

Example :

```
//abc.dtd

<!ELEMENT student (branch, rno, name, marks)>
<!ELEMENT branch (#PCDATA)>
<!ELEMENT rno (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
<!ATTLIST branch dept CDATA #REQUIRED>

//abc.xml
<?xml version = "1.0"?>
<!DOCTYPE student SYSTEM "abc.dtd">
<student>
    <branch dept = "CSE">
    <rno> 501 </rno>
    <name> naveen </name>
    <marks> 65 </marks>
    </branch>
</student>
```

Linking DTD to XML:

DTD declarations either internal XML document or make external DTD file, after linked to a XML document.

* Internal DTD You can write rules inside XML document using <!DOCTYPE ..> declaration. Scope of this DTD within this document. Advantages is document Validated by itself without external reference.

* External DTD You can write rules in a seperate file (with .dtd extension). later this file linked to a XML document. This way you can linked several XML documents refer same DTD rules.

Internal DTD:

Internal DTD you can declare inside your XML file. In XML file top <!DOCTYPE ...> declaration to declare the DTD.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE root_element [
  ...
  ...
]
```

Following internal DTD example define root element <student> and other element are second level element along with discipline attribute.

* DTD rules must be placed specifies top of the XML element (root element) in the document.

```
<?xml version = "1.0" ?>
<!DOCTYPE root_element [
    <!ELEMENT student (branch, rno, name,
                                marks)>

    <!ELEMENT branch (#PCDATA)>
    <!ELEMENT rno (#PCDATA)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT marks (#PCDATA)>
    <!ATTLIST branch dept CDATA
                        #REQUIRED>]

<student>
<branch dept = "CSE">
<rno> 501 </rno>
<name> naveen </name>
<marks> 65 </marks>
</branch>
</student>
```

External DTD:

External DTD are shared between multiple XML documents. Any changes are update in DTD document effect or updated come to a all XML documents

* External DTD are of two types.

a) Private DTD
b) Public DTD

a) Private DTD:

Private DTD identify by the SYSTEM keyword. Access for single or group of users.

* You can specify the rules in the external DTD file with .dtd extension. Later in XML file <!DOCTYPE..> declaration is present to link the DTD file

Syntax:

```
<!DOCTYPE root-element SYSTEM "dtd-file-location">
```

Example:

```
<!DOCTYPE root-element SYSTEM "abc-dtd">
```

b) Public DTD:

Public DTD identify by the PUBLIC keyword. Access any users and our XML editor are known to DTD

* Some common DTD : webDTD , XHTML , MathML etc.

Syntax:

```
<!DOCTYPE root_element PUBLIC "dtd_name"
                    "dtd_file_location">
```

Example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
                    1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1
                    -transitional.dtd">
```

XML Schema :

XML schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to database schema that describes the data in a database.

Syntax :

You need to declare a schema in your XML document as follows –

```
<xs: schema>
 - -
 - -
</xs: schema>
```

Example :

The following example shows how to use schema

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<xs: schema xmlns: xs = "http://www.w3.org/2001/
                                XMLSchema">
 <xs: element name = "contact">
  <xs: complexType>
   <xs: sequence>
    <xs: element name = "name" type = "xs: string"/>
    <xs: element name = "company" type = "xs: string"/>
    <xs: element name = "phone" type = "xs: int"/>
  </xs: sequence>
 </xs: complexType>
</xs: element>
</xs: schema>
```

The basic idea behind XML schemas is that they describe the legitimate format that an XML document can take.

Elements:

As we saw in XML - Elements chapter, elements are the building blocks of XML document. An element can be defined within an XSD as follows –

```
<xs:element name="x" type="y"/>
```

Definition Types

You can define XML schema elements in the following ways –

Simple Type:

Simple type element is used only in the context of the text. Some of the predefined simple types are:

xs:integer, xs:boolean, xs:string, xs:date.

For example –

```
<xs:element name="phone_number" type="xs:int"/>
```

Complex Type:

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents.

For example :

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string"/>
      <xs:element name = "company" type = "xs:string"/>
      <xs:element name = "phone" type = "xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In the above example, Address element consists of child elements. This is a container for other <xs:element> definitions, that allows to build a simple hierarchy of elements in the XML document.

Global Type :

With the global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the person and company for different address of the company. In such case, you can define a general type as follows –

```
<xs:element name = "AddressType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string"/>
      <xs:element name = "company" type = "xs:string"/>
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
```

Now let us use this type in our example as follows –

```
<xs:element name="Address1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address" type="AddressType"/>
      <xs:element name="phone1" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Address2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address" type="AddressType"/>
      <xs:element name="phone2" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

Attributes:

Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below -

```
<xs:attribute name="x" type="y"/>
```

DOCUMENT OBJECT MODEL

" The W3C (DOM) Document object Model is a platform and language - netural interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document".

* The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents the HTML document as a tree-structure.

* In XML, DOM is a standard for how to get, change, add or delete XML elements.

* In XML DOM we use few properties and methods.

XML Properties

1) nodeName
2) nodeValue
3) parentNode
4) childNode
5) attributes

Methods

1) getElementsByTagName (" ")
2) appendChild (node);
3) removeChild (node);

Get the value of an XML Element

```
txt = xmlDoc.getElementsByTagName("title")[0].
                    childNodes[0].nodeValue;
```

Example: xml program

```
<student>
  <Branch  Br="CSE">
    <Rno> 501 </Rno>
    <title> XYZ </title>
  </Branch>
  <Branch  Br="ECE">
    <Rno> 401 </Rno>
    <title> abc </title>
  </Branch>
</student>
```

XML DOM nodes:

According to the XML DOM, everything in an XML document is a node.

* The entire document is a document node.
* Every XML element is an element node.
* The text in the XML elements are text nodes.
* Every attribute is an attribute node.
* Comments are comment nodes.

Insert a new node in XML:

For inserting a new element we use appendChild which insert a childnode to a specified tag.

```
x = document.appendChild("marks")
```

Example:

```
getElementByTagName('dept')[1].childNode[0].
                                nodeValue = 240;
```

program:

```
<college>
  <dept Branch = 'CSE'>
      <student> 120 </student>
      <faculty> 30 </faculty>
  </dept>
  <dept Branch = 'ECE'>
      <student> 115 </student>
      <faculty> 30 </faculty>
  </dept>
</college>
```

In the above example instead of 120 we get 240.

Create a new element:

```
newElement = xmlDoc.CreateElement("lab");
xmlDoc.getElementByTagName("dept")[0].
                                appendChild(newElement);
```

In Javascript we write document

In XML we write Doc

To insert value into that:

```
xmlDoc.getElementByTagName("dept")[0].
                        childNode[2].nodeValue = 5;
```

Creating an attribute:

```
xmlDoc.getElementByTagName("dept")[2].
                        setAttribute('Branch', 'Mech');
```

For Deleting / Removing:

```
xmlDoc.getElementByTagName("dept")[0].
                        childNode[1].nodeValue = " ";
```

To remove child:

```
xmlDoc.getElementByTagName("dept")[0].
                        removeChild()
```

* XHTML

XHTML is HTML written as XML.

What is XHTML ?

* XHTML stands for EXtensible HyperText Markup Language.
* XHTML Is almost identical to HTML
* XHTML is supported by all major browsers

XHTML Elements :

* XHTML elements must be properly nested.
* XHTML elements must always be closed.
* XHTML elements must be in lowercase.
* XHTML documents must have one root element.

XHTML Attributes :

* Attribute names must be in lower case
* Attribute values must be quoted
* Attribute minimization is forbidden.

<!DOCTYPE ...> is mandatory :

An XHTML document must have a XHTML DOCTYPE declaration.

* A complete list of all the XHTML Doctypes is found in our HTML Tags Reference.
* The <html>, <head>, <title> and <body> elements must also be present, and the xmlns

attribute in <html> must specify the xml namespace for the document.

* XHTML Elements Must Always Be closed
* Empty Elements Must Also be closed.
* XHTML Elements Must Be in Lower Case.
* XHTML Attribute Names Must Be in Lower Case.
* Attribute Values Must Be Quoted

XML NameSpace :

* XML namespace provides a method to avoid element name conflicts
* To avoid such type of conflicts we use a name prefix.

Example:

Instead of < table > we write <f: table > and for all children of table it should be started/ ended with the same prefix f

```
<f: table>
    <f: tr>
    <f: td> ...  </f: td>
    <f: td> ...  </f: td>
    </f: tr>
</f: table>
```

Note:

xmlns attribute in html specifies the xml namespace for a document. This attribute will be added to first element of your xml.

Eg: <table xmlns: "filename">

Structure of XHTML:

1) DOCTYPE
2) header
3) body

// Create a student table which has 2 columns roll no & name with 3 rows.

Student.xhtml

```
<DOCTYPE html PUBLIC "-// w3c// DTD xHTML
                    1.0 Transitional // EN"
    "http: // www.w3.org / TR / xhtml / DTD /
                                xhtml1.dtd">

<html xmlns = "http:// www.w3.org / 1999 / xhtml">
<head> </head>
<body>
    <table>
      <tr>
        <th> Rno </th>
        <th> Name </th>
      </tr>
```

```
        <tr>
          <td> 501 </td>
          <td> Harika </td>
        </tr>
        <tr>
          <td> 502 </td>
          <td> Sai </td>
        </tr>
        <tr>
          <td> 503 </td>
          <td> Nikhil </td>
        </tr>
      </table>
    </body>
  </html>
```

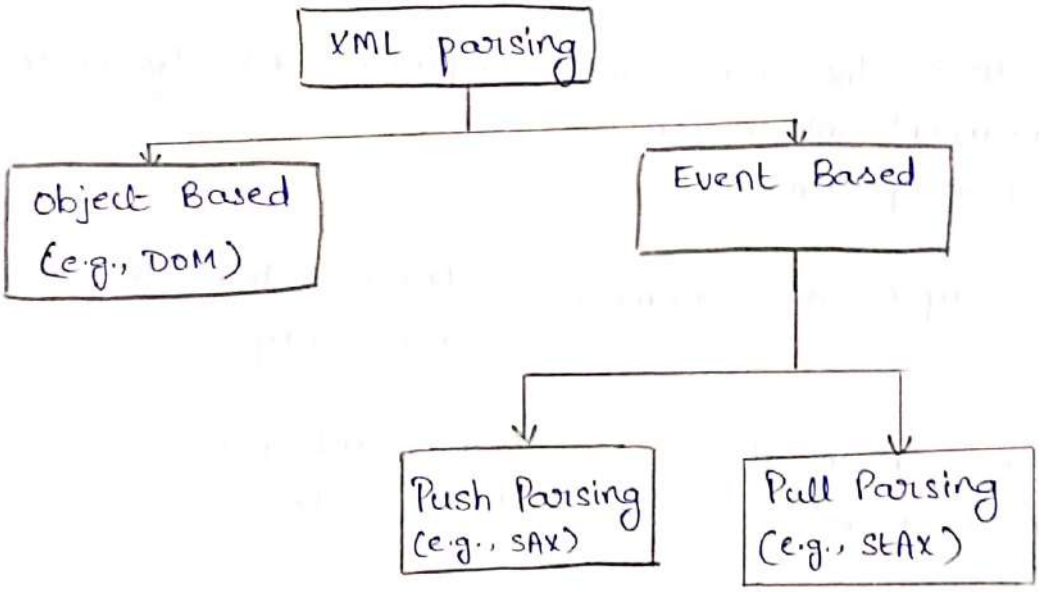
Chapter : 2

* PARSING XML DATA

we can access and parse the XML document in two ways.

- Parsing using DOM (tree based)
- Parsing using SAX (Event based)

* Parsing the XML doc. using DOM methods and properties are called as tree based approach whereas using SAX (simple Api for xml) methods and properties are called as event based approach



DOM and SAX Parsers

DOM	SAX
1) Tree data structure	1) Event based model.
2) Random access	2) Serial access
3) High memory usage	3) Low memory usage
4) Used to process multiple lines (document is loaded in memory)	4) Used to process the document only once.
5) Used to edit the document	5) Used to process parts of the document.
6) Stores the entire xml document into memory before processing	6) parses node by node.
7) occupies more memory	7) Doesn't store the XML in memory
8) we can insert or delete nodes	8) we can't insert or delete nodes.
9) Traverse in any direction	9) Top to bottom traversing
10) Document Object model (DOM) API	10) SAX is Simple API for XML
11) import javax.xml.parsers.*; import javax.xml.parsers.DocumentBuilder; import javax.xml.parsers.DocumentBuilderFactory;	11) packages required to import import javax.xml.parsers.*; import org.xml.sax.*;
12) DOM is slow rather than SAX	12) SAX generally run a little faster than DOM

* Document Object Model is for defining the standard for accessing and manipulating XML documents. XML DOM is used for

- Loading the xml document
- Accessing the xml document
- Deleting the elements of xml document
- Changing the elements of xml document.

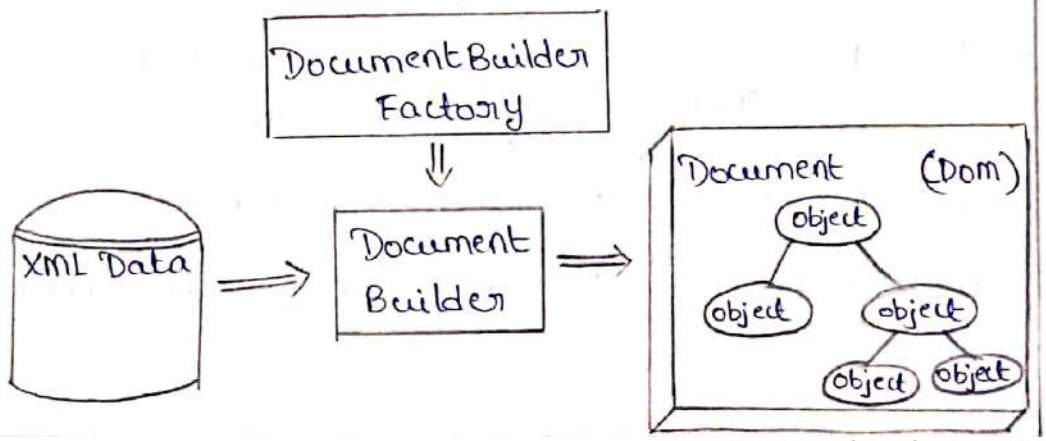
* According to the DOM, everything in an XML document is a node. It considers

- The entire document is a document node
- Every XML element is an element node.
- The text in the XML elements is text nodes.
- Every attribute is an attribute node.
- Comments are comment nodes.

DOM based XML Parsing:

DOM parser parses the entire XML document and loads it into memory; then models it in a "TREE" structure for easy traversal or manipulation.

In short, it turns a XML file into DOM or Tree structure, and you have to traverse a node by node to get what you want.



* In this approach, to access XML document, the document object model implementation is defined in the following packages:

- javax.xml.parsers
- org.w3c.dom

* The following DOM java classes are necessary to process the XML document:

- DocumentBuilderFactory class creates the instance of DocumentBuilder.
- DocumentBuilder produces a document (a DOM) that conforms to DOM specification.

* The following methods and properties are necessary to process the XML document:

Property	Meaning.
nodeName	Finding the name of the node.
nodeValue	Obtaining value of the node.
parentNode	To get parent node.
childNodes	Obtain child nodes.
attributes	For getting the attributes values

Method	Meaning
getElementByTagName (name)	To access the element by specifying its name
appendChild (node)	To insert a child node
removeChild (node)	To remove existing child node.

Java Program to Create XML File

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import java.util.Scanner;
import javax.xml.transform.stream.*;
import java.io.*;
public class CreateXML
{
  public static void main(String[] args) throws
                                    Exception
  {
    DocumentBuilderFactory factory = Document
                    BuilderFactory.newInstance();
    DocumentBuilder builder = factory.
                    newDocumentBuilder();
    Document doc = builder.newDocument();
    Element rootele = doc.createElement
                    ("student-details");
    Element studentele = doc.createElement
                    ("student");
    Element idele = doc.createElement ("studentid");
    Element nameele = doc.createElement ("name");
    Element markesele = doc.createElement
                    ("marks");
    Text t1 = doc.createTextNode ("501");
    Text t2 = doc.createTextNode ("naveen");
    Text t3 = doc.createTextNode ("90");
```

```
Text
idele.appendChild(t1);
nameele.appendChild(t2);
marksele.appendChild(t3);

studentele.appendChild(idele);
studentele.appendChild(nameele);
studentele.appendChild(marksele);

rootele.appendChild(studentele);
doc.appendChild(rootele);

Transformer t = TransformerFactory.
        newInstance().newTransformer();

t.transform(new DOMSource(doc), new
        StreamResult(new FileOutputStream
                ("student.xml")));
}
}
```

Above code will generate an xml file with a name student.xml

<u>Student.xml</u>

```
<?xml version="1.0" encoding="UTF-8"
                standalone="no"?>
<student-details>
  <student>
    <studentid>501</studentid>
    <name>naveen</name>
    <marks>90</marks>
  </student>
</student-details>
```

Using SAX Parser :

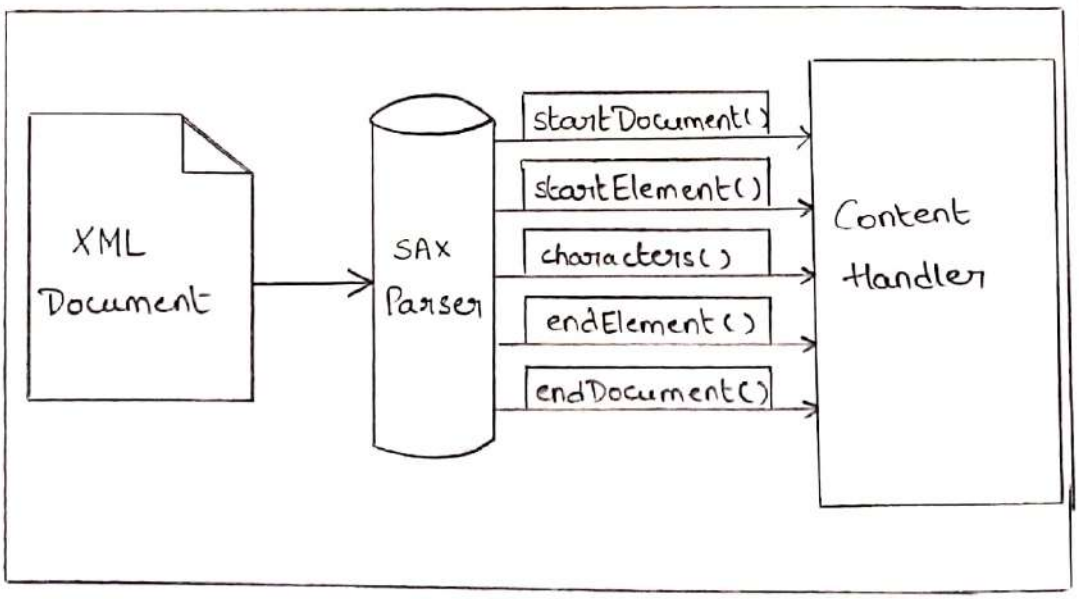
SAX Parser is different from the DOM Parser, where SAX parser doesn't load the complete XML into the memory, instead it parses the XML line by line triggering different events as and when it encounters different elements like: opening tag, closing tag, character data and comments and so on. This is the reason why SAX Parser is called an event based parser.

* Along with the XML source file, we also register a handler which extends the Default Handler class. The DefaultHandler class provides different callbacks out of which we would be interested in:

startElement() – triggers this event when the start of the tag is encountered.

endElement() – triggers this event when the end of the tag is encountered.

characters() – triggers this event when it encounters some text data.



Let's create a demo program to read xml file with SAX parser to understand fully.

student.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<student-details>
  <student>
    <studentid> 501 </studentid>
    <name> Ramu </name>
    <address> ECIL </address>
    <gender> Male </gender>
  </student>
  <student>
    <studentid> 502 </studentid>
    <name> Mahi </name>
    <address> BHEL </address>
    <gender> Male </gender>
  </student>
</student-details>
```

Java program to read data from xml (student.xml) file

```
import java.io.*;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class SAXParserDemo extends DefaultHandler
{
    public void startDocument()
    {
        System.out.println("begin parsing document");
    }
    public void startElement(String url,
            String localname, String qName,
                        Attributes att)
    {
        System.out.print("<"+qName+">");
    }
    public void characters(char[] ch, int start,
                        int length)
    {
        for(int i=start; i<(start+length); i++)
        {
            System.out.print(ch[i]);
        }
    }
```

```
public void endElement (String uri, String
                localname, String qName )
{
    System.out.print ("</"+ qName + ">");
}
public static void main (String [] arg)
                throws Exception
{
    SAXParser p = SAXParserFactory.
        newInstance().newSAXParser();

public void endDocument()
{
    System.out.println ("End parsing
                        document");
}
public static void main (String [] arg)
                throws Exception
{
    SAXParser p = SAXParserFactory.
        newInstance().newSAXParser();

    p.parse (new FileInputStream ("student.
                xml"), new SAXParserDemo());

}

}
```

UNIT - III

INTRODUCTION TO SERVLETS

Servlet technology is used to create web application (resides at server side and generates dynamic web page).

* Servlet technology is robust and scalable because of java language. Before servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language. But there are many disadvantages for this technology.

* There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

What is a Servlet ?

Servlet is described in many ways, depending on the context.

* Servlet is a technology i.e., used to create web application.

* Servlet is an API that provides many interfaces and classes including documents.

* Servlet is an interface that must be implemented for creating any servlet.

* Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any types of requests

* Common Gateway Interface (CGI):

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.

Disadvantages of CGI:

There are many problems in CGI technology:

1) If number of clients increases, it takes more time for sending response.

2) For each request, it starts a process and web server is limited to start processes.

3) It uses platform dependent language. e.g., C, C++, perl.

Advantage of Servlet:

There are many advantages of servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the processes such as they share a common memory area, light weight, cost of communication between the threads are low. The basic benefits of servlet are as follows:

a) Better performance:

because it creates a thread for each request not process.

b) Portability:

because it gives java language.

c) Robust:

Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.

d) Secure:

because it uses java language.

* Life-Cycle Of a Servlet:

Three methods are central to the life cycle of a servlet. These are init(), service() and destroy(). They are implemented by every servlet and are invoked at specific times by the server. Let us consider a typical user scenario to understand when these methods are called.

* First, assume that a user enters a Uniform Resource Locator (URL) to a Web browser. The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server.

* Second, this HTTP request is received by the web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server.

* Third, the server invokes the init() method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may configure itself.

* Fourth, the server invokes the service() method of the servlet. This method is called to process the HTTP request. You will see that it is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client.

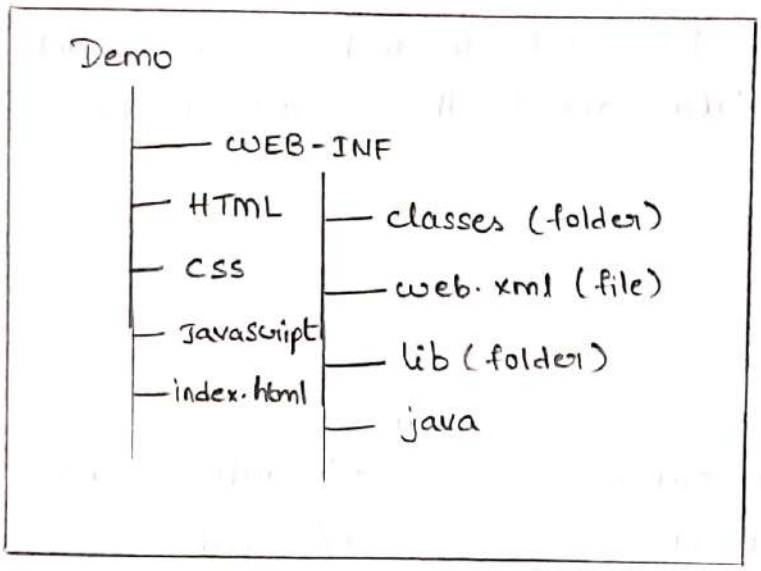
* The servlet remains in the server's address space and is available to process any other HTTP requests received from clients. The service() method is called for each HTTP request.

* Finally, the server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the destroy() method to relinquish any resources such as file handles that are allocated for the servlet. Important data may be saved to a persistant store. The memory allocated for the servlet and its objects can then be garbage collected.

Steps to create Servlet Program:

1) create a directory structure
2) create a servlet
3) compile the servlet
4) create a deployment descriptor
5) start the server and deploy the project.
6) Access the servlet.

1) create a directory structure:



structure of servlet

2) create a servlet:

servlet can be created in 3 different ways.

a) By implementing servlet interface.
b) By inheriting the GenericServlet class
c) By inheriting the HttpServlet class.

3) Compile the servlet :

Servlet can be compiled by using jar files. jar files are required to be loaded.

servlet-api.jar

↓

we can set the classpath or it can be copied manually into the server.

4) create a deployment descriptor :

Deployment descriptor is an XML file with a name web.xml.

* From this xml file web container gets the information about the servlet to be invoked.

web.xml

```
<web-app>
  <servlet>
    <servlet-name> filename </servlet-name>
    <servlet-class> classname </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> filename </servlet-name>
    <url-pattern> replica </url-pattern>
  </servlet-mapping>
</web-app>
```

5) Start the servlet and deploy the project:

There are two ways to deploy the project

a) hard deployment

b) soft deployment

a) Soft deployment:

Copy the demo folder into server manually.

In htdocs (i.e., server) copy & paste the complete folder of the project.

In appache web-apps is the server.

c://xampp/Tomcat/web-apps

↑

copy complete folder.

b) Soft deployment:

For soft deployment first create a war file.

By compressing all files (i.e., zipping) a war file is generated.

* A war file is created with Jar commands.

localhost:8080/manager/html.

D:/>Demo/Jar cvf Demo.war;

6) Access the servlet:

http://localhost/Demo

if filename is not specified then it directly access index.html.

* The Servlet API :

Two packages contain the classes and interfaces that are required to build servlets. They are :

a) javax.servlet

b) javax.servlet.http

a) javax.servlet package :

Interface	Description
Servlet	Declares lifecycle methods for a servlet.
ServletConfig	Allows servlets to get initialization parameters
ServletContext	Enables servlets to log events and access information about their environment
ServletRequest	Used to read data from client request.
ServletResponse	Used to write data to a client response.
SingleThreadModel	Indicates that the servlet is thread safe.

class	Description
Generic Servlet	Implements the Servlet and ServletConfig interfaces.
ServletInputStream	provides an input stream for reading requests from a client.
ServletOutputStream	provides an output stream for writing responses to a client.
ServletException	indicates a servlet error occured.
UnavailableException	indicates a servlet is unavailable.

The Servlet Interface

All servlets must implement the Servlet Interface. It declares the init(), service() and destroy() methods that are called by the server during the life cycle of a servlet. The methods defined by servlet are shown in table below.

Method	Description
void destroy()	called when the servlet is unloaded.
ServletConfig getServletConfig()	returns a ServletConfig object that contains any initialization parameters
String getServletInfo()	returns a string describing the servlet.

void init(ServletConfig sc) throws ServletException	called when the servlet is initialized. Initialization parameters for the servlet can be obtained from sc. An UnavailableException should be thrown if a servlet cannot be initialized.
void service(ServletRequest req, ServletResponse res) throws ServletException, IOException	called to process a request from a client. The request from the client can be read from req. The response to the client can be written to res. An exception is generated if a servlet or IO problem occurs.

The ServletConfig Interface

The ServletConfig interface is implemented by the server. It allows the servlet to obtain configuration data when it is loaded. The methods declared by this interface are summarized here:

Method	Description
ServletContext getServletContext()	returns the context for this servlet.
String getInitParameter (String param)	returns the value of initialization parameter named Param
Enumeration getInitParameterNames()	returns an enumeration of all initialization parameter names.
String getServletName()	returns the name of the invoking servlet.

The ServletContext Interface

The ServletContext interface is implemented by the server. It enables the servlets to obtain information about their environment. Several of its methods are summarized in table below:

Method	Description
Object getAttribute (String attr)	returns the value of the server attribute named attr
String getMimeType (String file)	returns the MIME type of file.
String getRealPath (String vpath)	returns the real path that corresponds to the virtual path vpath.
String getServerInfo()	returns information about the server.
void log(String s)	writes s to the servlet log
void log(String s, Throwable e)	write s and the stack trace for e to the servlet log
void setAttribute (String attr, Object val)	sets the attribute specified by attr to the value passed in val

The ServletRequest Interface

The ServletRequest interface is implemented by the server. It enables a servlet to obtain information about the client request. Several of its methods are summarized in table below.

Method	Description
Object getAttribute (String attr)	returns the value of the attribute named attr.
String getCharacterEncoding ()	returns the character encoding of the request
int getContentLength()	returns the size of the request. The value -1 is returned if the size is unavailable.
String getContentType()	returns the type of the request. A null value is returned if the type cannot be determined.
ServletInputStream getInputStream() throws IOException	returns the ServletInputStream that can be used to read binary data from the request. An IllegalStateException is thrown if getReader() has already been invoked for this request.
String getParameter (String pname)	returns the value of the parameter named pname.
Enumeration getParameterNames ()	returns an enumeration of the parameter names for this request.
String[] getParameterValues (String name)	returns an array containing values associated with the parameter specified by name
String getProtocol()	returns a description of the protocol.

BufferedReader getReader() throws IOException	returns a buffered reader that can be used to read text from the request. An IllegalStateException is thrown if getInputStream() has already been invoked for this request.
String getRemoteAddr()	returns the string equivalent of the client IPaddress.
String getRemoteHost()	returns the string equivalent of the client host name.
String getScheme()	returns the transmission scheme of the URL used for the request
String getServerName()	returns the name of the server.
int getServerPort()	returns the port number.

The ServletResponse Interface

The ServletResponse interface is implemented by the server. It enables a servlet to formulate a response for a client. Several of its methods are summarized in table below.

Method	Description
String getCharacterEncoding()	returns the character encoding for the response
ServletOutputStream getOutputStream() throws IOException	returns a ServletOutputStream that can be used to write binary data to the response. An IllegalStateException is thrown if getWriter() has already been invoked for this request.

PrintWriter getWriter() throws IOException	returns a PrintWriter that can be used to write character data to the response. An IllegalStateException is thrown if getOutputStream() has already been invoked for this request.
void setContentLength (int size)	sets the content length for the response to size.
void setContentType (String type)	sets the content type for the response to type

The SingleThreadModel Interface

This interface is used to indicate that only a single thread will execute the service() method of a servlet at a given time. It defines no constants and declares no methods. If a servlet implements this interface, the server has two options. First, it can create several instances of the servlet. When a client request arrives, it is sent to an available instance of the servlet. Second, it can synchronize access to the servlet.

The GenericServlet Class

The GenericServlet class provides implementations of the basic life cycle methods for a servlet and is typically subclassed by servlet developers. GenericServlet implements the Servlet and ServletConfig interfaces. In addition, a method to append a string to the server log file is available. The signatures of this method are shown here:

void log(String s)
void log(String s, Throwable e)

Here, s is the string to be appended to the log, and e is an exception that occured.

The ServletInputStream Class

The ServletInputStream class extends InputStream. It is implemented by server and provides an input stream that a servlet developer can use to read the data from the client request. It defines the default constructor. In addition, a method is provided to read bytes from the stream. Its signature is shown here:

int readLine(byte[] buffer, int offset, int size)
throws IOException

* Here, array buffer is the array into which size bytes are placed starting at offset. The method returns the actual number of bytes or -1 if the end-of-the stream condition is encountered.

The ServletOutputStream class

The ServletOutputStream class extends OutputStream. It is implemented by the server and provides an output stream that a servlet developer can be used to write data to a client response. A default constructor is defined. It also defines print() and println() methods, which output data to the stream

* Handling Http Request & Responses

The HttpServlet class provides specialized methods that handle the various types of Http requests. A servlet developer typically overrides one of these methods. These methods are doDelete(), doGet(), doHead(), doOptions(), doPost(), doPut() and doTrace(). A complete description of the different types of HTTP requests is beyond the scope of this book. However, the GET and POST requests are commonly used when handling from input. Therefore, this section presents examples of these cases.

* Handling HTTP GET Requests Here we will develop a servlet that handles an HTTP GET request. The servlet is invoked when a form on a Web page is submitted. The example contains two files. A Web page is defined in ColorGet.htm and a servlet is defined in ColorGetServlet.java. The HTML source code for ColorGet.htm. is shown in following listing. It defines a form that contains a select element and a submit button. Notice that the action parameter of the form tag specifies a URL. The URL identifies a servlet to process the HTTP GET request.

```
<html>
  <body>
    <center>
    <form name = "Form1"
        action = "http://localhost:8080/examples/
                                servlet/ColorGetServlet">
    <B> Color </B>
    <select name = "Color" size = "1">
    <option value = "Red"> Red </option>
    <option value = "Green"> Green </option>
    <option value = "Blue"> Blue </option>
    </select>
    <br><br>
    <input type = submit value = "Submit">
    </form>
  </body>
</html>
```

* The source code for ColorGetServlet.java is shown in the following listing. The doGet() method is overridden to process any HTTP GET requests that are sent to this servlet. It uses the getParameter() method of HttpServletRequest to obtain this selection that was made by the user. A response is then formulated.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorGetServlet extends HttpServlet
{
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                 throws ServletException, IOException
   {
      String color = request.getParameter("color");
      response.setContentType("text/html");
      PrintWriter pw = response.getWriter();
      pw.println("<B>The selected color is: ");
      pw.println(color);
      pw.close();
   }
}
```

Compile the servlet and perform these steps to test this example:

1. Start Tomcat, if it is not already running.
2. Display the Web page in a browser.
3. Select a color.
4. Submit the Web page.

* After completing these steps, the browser will display the response that is dynamically generated by the servlet. One other point: Parameters for an HTTP GET request are included as part of the URL that is sent to the Web server. Assume that the user selects the red option and submits the form. The URL sent from the browser to the server is
http://localhost:8080/examples/servlet/ColorGetServlet?

Color = Red. The characters to the right of the question mark are known as the query string.

Handling HTTP POST Requests.

* Here we will develop a servlet that handles an HTTP POST request. The servlet is invoked when a form on a Web page is submitted. The example contains two files. A web page is defined in ColorPost.htm ~~and a servlet is define~~ is shown in the following listing. It is identical to ColorGet.htm except that the method parameter for the form tag explicitly specifies that the POST method should be used, and the action parameter for the form tag specifies a different servlet.

```
<html>
  <body>
   <center>
   <form name = "Form1" method = "post"
          action = "http://localhost:8080/examples/
                                servlet/ColorPostServlet">
    <B> Color:</B>
    <select name = "color" size = "1">
    <option value = "Red"> Red </option>
    <option value = "Green"> Green </option>
    <option value = "Blue"> Blue </option>
    </select>
    <br> <br>
    <input type = submit value = "Submit">
   </form>
  </body>
</html>
```

The source code for ColorPostServlet.java is shown in the following listing. The doPost() method is overridden to process any HTTP POST request to obtain the selection that was made by the user. A response is then formulated.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorPostServlet extends HttpServlet
{
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                  throws ServletException, IOException
    {
        String color = request.getParameter("color");
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<B>The selected color is:");
        pw.println(color);
        pw.close();
    }
}
```

Compile the servlet and perform the same steps as described in the previous section to test it.

Note:

Parameters for an HTTP POST request are not included as part of the URL that is sent to the web server. In this example, the URL sent from the browser to the server is:

http://localhost:8080/examples/servlet/ColorGetServlet

The parameter names and values are sent in the body of the HTTP request.

* Using Cookies

Now, let's develop a servlet that illustrates how to use cookies. The servlet is invoked when a form on a Web page is submitted. The example contains three files as summarized here:

File Description AddCookie.htm Allows a user to specify a value for the cookie named MyCookie.

* AddCookieServlet.java Processes the submission of AddCookie.htm

* GetCookiesServlet.java Displays cookie values.

* The HTML source code for AddCookie.htm is shown in the following listing.

* This page contains a text field in which a value can be entered. There is also a submit button on the page. When this button is pressed, the value in the text field is sent to AddCookie Servlet via an HTTP POST request.

```
<html>
  <body>
    <center>
    <form name = "Form1" method = "post"
          action = "http://localhost:8080/examples/
                                servlet/AddCookieServlet">
    <B> Enter the value for MyCookie: </B>
    <input type=textbox name="data" size =25
                                value = " ">
    <input type= submit value="submit">
    </form>
```

```
</body>
</html>
```

The source code for AddCookieServlet.java is shown in the following listing. It gets the value of the parameter named "data". It then creates a Cookie object that has the name "MyCookie" and contains the value of the "data" parameter. The cookie is then added to the header of the HTTP response via the addCookie() method. A feedback message is then written to the browser.

```
import java.io.*
import javax.servlet.*;
import javax.servlet.http.*;
public class AddCookieServlet extends HttpServlet
{
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
            throws ServletException, IOException
    {
        // get parameter from HTTP request
        String data = request.getParameter("data");
        // create cookie
        Cookie cookie = new Cookie("MyCookie", data);
        // add cookie to HTTP response
        response.addCookie(cookie);
        // write output to browser
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<B> MyCookie has been set to");
```

```
        pw.println(data);
        pw.close();
    }
}
```

* The source code for GetCookiesServlet.java is shown in the following listing. It invokes the getCookies() method to read any cookies that are included in the HTTP GET request. The names and values of these cookies are then written to the HTTP response. Observe that the getName() and getValue() methods are called to obtain this information.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetCookiesServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException
    {
        // Get cookies from header of HTTP request
        Cookie[] cookies = request.getCookies();
        // Display these cookies.
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        pw.println("<B>");
        for (int i=0; i< cookies.length; i++)
        {
            String name = cookies[i].getName();
            String value = cookies[i].getValue();
            pw.println("name = "+name+" value = "+
                                        value);
        }
```

```
        pw.close();
    }
}
```

Compile the servlet and perform these steps:

1. Start Tomcat, if it is not already running.
2. Display AddCookie.htm in a browser.
3. Enter the value for MyCookie
4. Submit the web page.

After completing these steps you will observe that a feedback message is displayed by the browser. Next, request the following URL via the browser

http://localhost:8080/examples/servlet/GetCookieServlet

observe that the name and value of the cookie are displayed in the browser.

* In this example, an expiration date is not explicitly assigned to the cookie via the setMaxAge() method of Cookie. Therefore, the cookie expires when the browser session ends. You can experiment by using setMaxAge() and observe that the cookie is then saved to the disk on the client machine.

* Session Tracking

HTTP is a stateless protocol. Each request is independent of the previous one. However, in some applications, it is necessary to save state information so that information can be collected from several interactions between a browser and a server. Sessions provide such a mechanism.

* A session can be created via the getSession() method of HttpServletRequest. An HttpSession object is returned. This object can store a set of bindings that associate names with objects. The setAttribute(), getAttribute(), getAttributeNames() and removeAttribute() methods of HttpSession manage these bindings. It is important to note that session state is shared among all the servlets that are associated with a particular client.

* The following servlet illustrates how to use session state. The getSession() method gets the current session. A new session is created if one does not already exist. The getAttribute() method is called to obtain the object that is bound to the name "date". That object is a Date object that encapsulates the date and time when this page was last accessed. A Date object encapsulating the current date and time is then created. The setAttribute() method is called to bind the name "date" to this object.

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateServlet extends HttpServlet
{
    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException.
    {
        // Get the HttpSession object
        HttpSession hs = request.getSession (true);
        // Get Writer
        response.setContentType ("text/html");
        PrintWriter pw = response.getWriter ();
        pw.print ("<B>");
        // Display date/time of last access
        Date date = (Date) hs.getAttribute ("date");
        if (date != null)
        {
            pw.print ("Last access:" + date + "<br>");
        }
        // Display current date/time
        date = new Date ();
        hs.getAttribute ("date", date);
        pw.println ("Current date : " + date);
    }
}
```

When you first request this servlet, the browser displays one line with the current date and time information. On subsequent invocations, two lines are displayed. The first line shows the date and time when the servlet was last accessed. The second line shows the current date and time.

* Introduction to JDBC :-

→ In today's scenario, many enterprise level applications need to intract with databases for storing information.

→ For this purpose, we used an API (Application programming Interface) i.e ODBC (open Database connectivity).

→ The ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e platform dependent and unsecured).

→ That is why java has defined its own API, called JDBC (Java Database connectivity). that uses JDBC drivers (written in Java language).

→ The JDBC drivers are more compatible with Java Applns to provide database communication.

→ JDBC is a Java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

→ JDBC supports a wide level of portability and JDBC is simple and easy to use.

→ In JDBC API, a programmer needs a specific driver to connect to specific database.

RDBMS	Driver
Oracle	oracle.jdbc.driver.OracleDriver
MySQL	com.mysql.jdbc.Driver
SyBase	com.sybase.jdbc.SybDriver
SQLServer	com.microsoft.jdbc.SqlServer
DB2	com.ibm.db2.jdbc.net.DB2Driver

* List of some popular Drivers.*

* JDBC Architecture :-

The main function of the JDBC is to provide a standard abstraction for java applications to communication with databases.

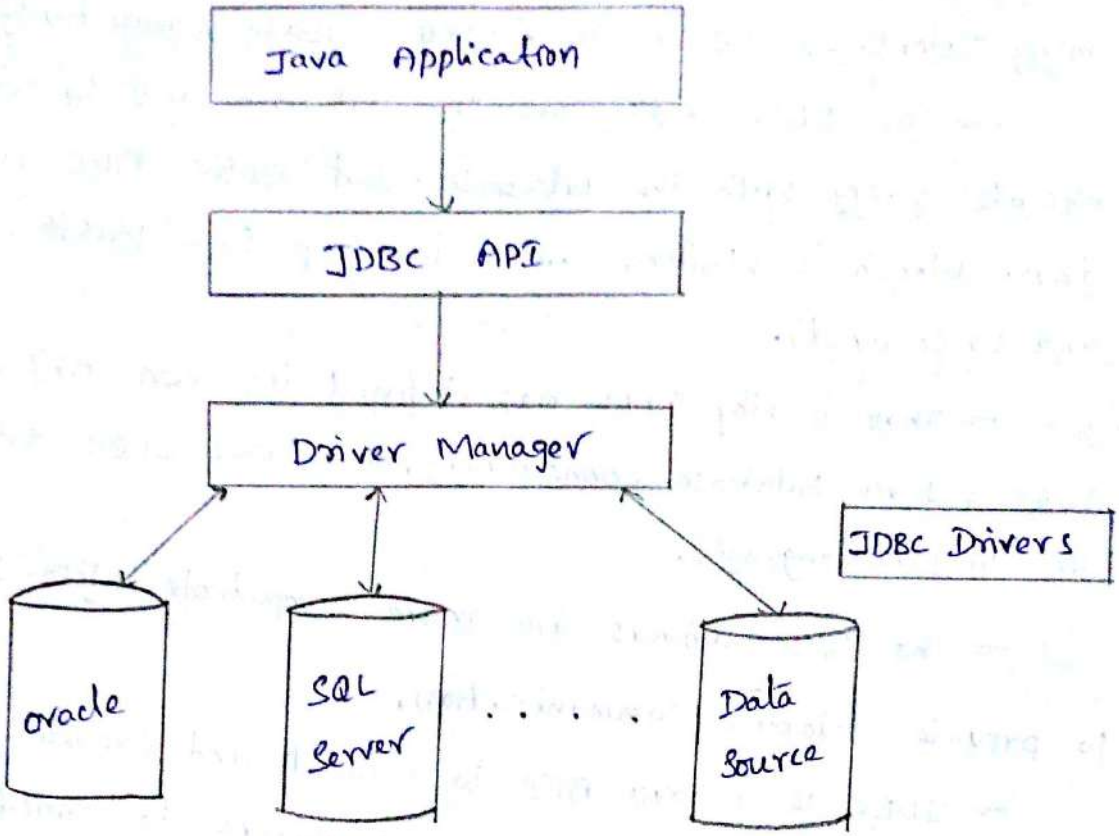


Fig:- The JDBC Architecture

As shown in figure, The Java application that wants to communicate with a database has to be programmed using JDBC API.

The JDBC Driver is required to process the SQL requests and generate the results.

The JDBC driver has to be plays an important role in the JDBC architecture. The Driver Manager uses some specific drivers to effectively connect with specific databases.

→ JDBC Driver is a software component that enables Java application to intract with the database.

There are 4 types of JDBC drivers, those are

→ Type -1 Driver (JDBC - ODBC bridge driver)
→ Type - 2 Driver (partial JDBC driver)
→ Type - 3 Driver (pure java driver for middleware)
→ Type - 4 Driver (pure java driver with direct database connection)

* Type-1 Driver (JDBC - ODBC bridge driver) :-

The type -1 driver acts as a bridge between JDBC and other database connectivity mechanisms such as ODBC.

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC- ODBC bridge driver converts JDBC method calls into the ODBC method calls.

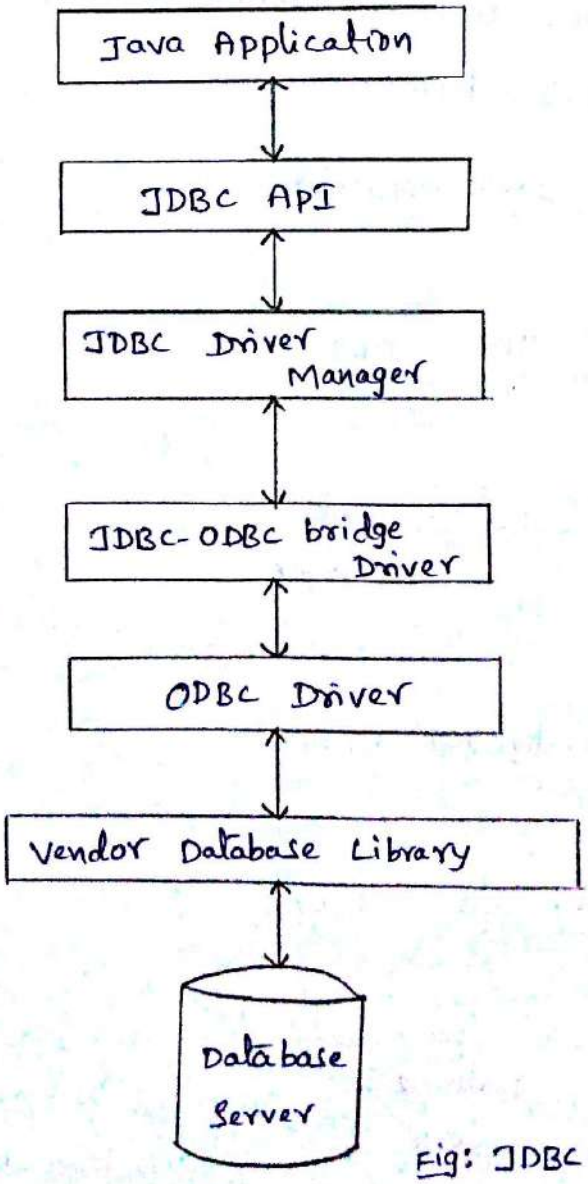


Fig: JDBC - ODBC Bridge Driver

Advantages :

* Easy to use.
* Can be easily connected to any database.

Disadvantages :

* performance degraded because large number of trans-lations (i.e JDBC calls to ODBC calls).
* The ODBC driver needs to be installed on the client machine.

* Type-2 Driver (partial JDBC driver) :-

The type-2 driver uses the client-side libraries of the database. So This driver is also called as Native-API driver.

This driver converts JDBC method calls into native calls of the database API. It is not written entirely in java, so it is called as partial JDBC driver.

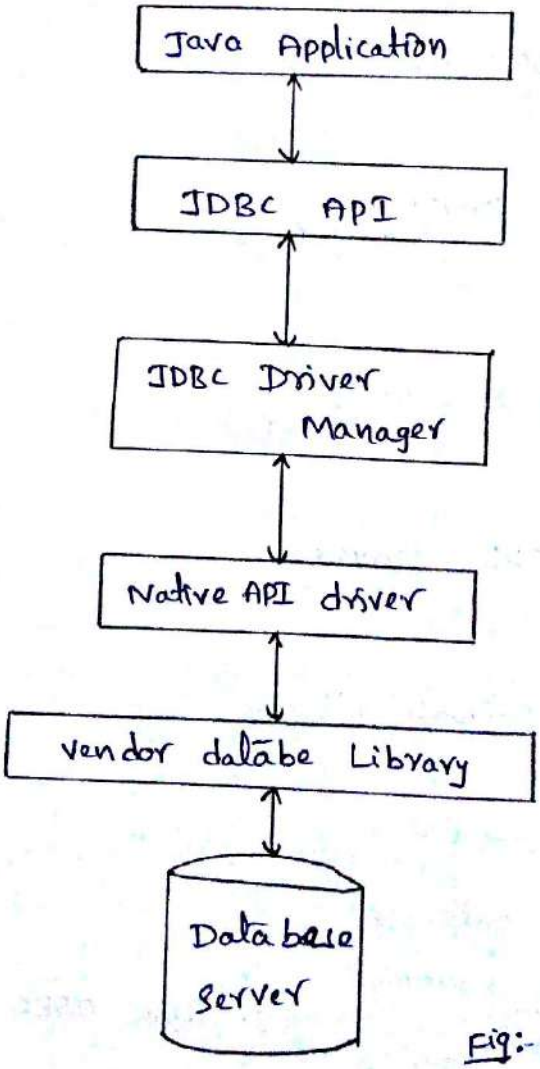


Fig:- Native API driver

Advantages:

* performance upgraded than JDBC-ODBC bridge driver.
* suitable to use with server-side applications.

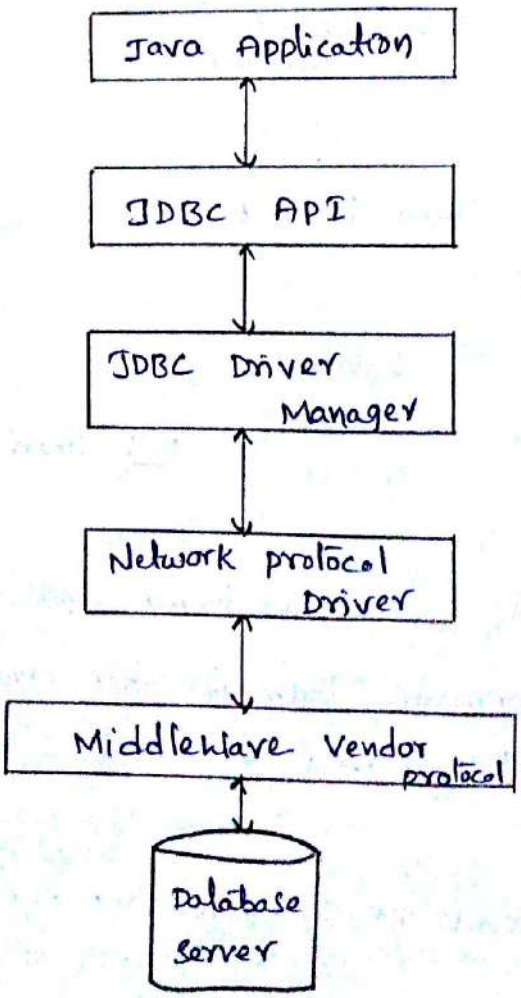
Disadvantages:

* This Native driver needs to installed on the each client machine
* The vendor client library needs to be installed on client machine
* It may increase the cost of the application if the application needs to run on different platforms.

* Type-3 Driver (pure Java driver for middleware) :-

The type-3 driver is completely implemented in Java, hence it is a pure Java JDBC driver.

The type-3 driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. so it is called as Network protocol driver.



Advantages:

* No client side library is required on client side.
* pure Java drivers and auto downloadable.

Disadvantages:

* Network support is required on client machine.
* This driver is costly compared to other drivers.

* Type - 4 Driver (pure Java driver with direct database connection):-

The type-4 driver is a pure Java driver, which converts JDBC calls directly into the vendor-specific database protocol. that is why it is known as thin driver.

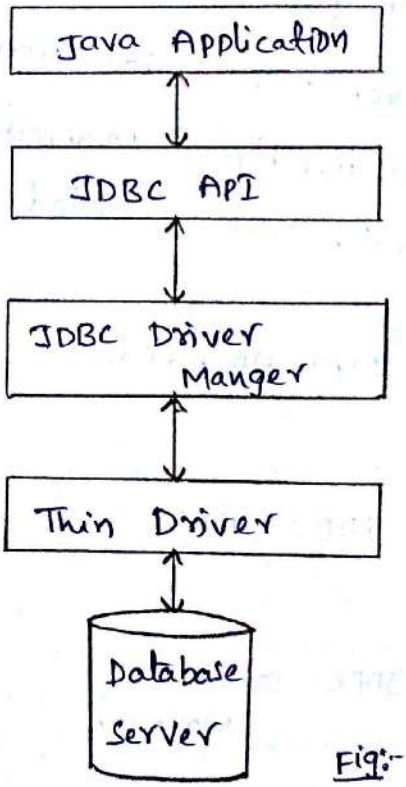


Fig:- Thin Driver.

Advantages:

* This driver is pure Java driver and auto downloadable.
* Better performance than all other drivers
* No software is required at client side or server side.

Disadvantages:

* Drivers depends on the Database.

* Database programming using JDBC :-

JDBC APIs are used by a Java application to communicate with a database.

In otherwords, we use JDBC connectivity code in Java application to communicate with a database.

There are 5 steps to connect any Java application with the database in java using JDBC. They are as follows:

Step 1 : Register the driver class

Step 2 : Creating connection

Step 3 : Creating statement

Step 4 : Executing SQL statements

Step 5 : closing connection.

* Step 1 :- (Register the driver class)

In this step, we register the driver class with driver Manager by using forName() method of class class.

Syntax: Class.forName(Driver class Name)

Example: Class.forName("oracle.jdbc.driver.OracleDriver");

* Step 2 :- (Creating connection)

In this step, we can create a connection with database Server by using getConnection() method of DriverManager class.

Syntax: getConnection(string url, string name, string pwd)

Example:

```
Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe",
                "System", "admin");
```

* Step 3 :- (Creating statement)

After the connection made, we need to create the statement object to execute the SQL statements.

The createStatement() method of Connection interface is used to create statement. This statement object is responsible to execute SQL statements with the database.

Syntax: createStatement()
Example:
Statement stmt = con.Statement();

* Step 4:- (Executing SQL statements)

After the statement object is created, it can be used to execute the SQL statements by using executeUpdate() (or) executeQuery() method of statement interface.

The executeQuery() method is only used to execute SELECT statements.

The executeUpdate() method is used to execute all SQL statements except SELECT statements.

```
Syntax:  executeQuery(String query)
         executeUpdate(String query)

Example: // using executeQuery()
         String query = "Select * from emp";
         Resultset rs = stmt.executeQuery(query);
         // using executeUpdate()
         String query = "insert into emp values(504,'Madhu',29);
          stmt.executeUpdate(query);
```

* Step 5:- (closing the connection)

After executing all the SQL statements and obtaing the results, we need to close the connection and release the session.

The close() method of Connection interface is used to close the connection.

Syntax:- close()
Example:- con.close();

* **Example:- (connectivity with oracle database)**

For connecting java application with the oracle database, we need to know following information to perform database connectivity with oracle.

In this example we are using oracle 10g as the database, so we need to know following information for the oracle database.

* **Driver class:** The driver class for oracle database is "oracle.jdbc.driver.OracleDriver".
* **Connection URL:** The connection URL for the oracle 10G database is "jdbc:oracle:thin:@localhost:1521:xe".

Where jdbc is the API, oracle is the database, thin is the driver, localhost is the servername on which oracle is running, 1521 is the port number and XE is the oracle service name.

* **username:** The default username for the oracle database is "system".
* **password:** password is given by the user at the time of installing the oracle database.

→ To connect java application with the oracle database ojdbc14.jar file is required to be loaded.

→ There are two ways to load the ojdbc14.jar file, we need to follow any one of two ways.

1. paste the ojdbc14.jar file in "java/jre/lib/ext" folder
2. set classpath

Firstly, search the ojdbc14.jar file then go to "java/jre/lib/ext" folder and paste the jar file here.

(or)

Set class path: To set classpath, goto environment variable then click on new tab. In variable name write classpath and in variable value paste the path to ojdbc14.jar by appending ojdbc14.jar;.; as

"c:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar;.;".

* Example:

Let's first create a table and insert two or more records in oracle database.

```
SQL> create table emp(id number(10), name varchar2(40),
                                age number(3));

SQL> insert into emp values(501, 'Madhu', 30);
SQL> insert into emp values(502, 'Hari', 32);
SQL> insert into emp values(503, 'Satti', 33);
```

* program: connect java application with oracle database for selecting or retriving data.

SelectData.java

```
import java.sql.*;
import java.util.*;
class SelectData
{
public static void main(String args[])
{
try
{ // Step 1: load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");
 // Step 2: create the connection object
Connection con = DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:xe", "system", "admin");
// Step 3: create the statement object
Statement stmt = con.CreateStatement();
// Step 4: execute query
ResultSet rs = stmt.executeQuery("select * from emp");
while (rs.next())
{
System.out.println(rs.getInt(1) + "    " + rs.getString(2) + "    "
                    + rs.getString(3));
}
//Step 5: close the connection object
con.close();
} catch (Exception e) { System.out.println(e); }
}
}
```

output:

```
D:\> javac SelectData.java
D:\> java Select-Data
501     Madhu     30
502     Hari      32
503     Satti     33
```

* program: connect Java application with oracle database for inserting data.

InsertData.java

```
import java.sql.*;
import java.util.*;
class InsertData
{
public static void main (String args[])
{
 try
 {
 Class.forName ("oracle.jdbc.driver.OracleDriver");
 Connection con = DriverManager.getConnection ("jdbc : oracle : thin
                    :@localhost : 1521 : xe", "system", "admin");

 Statement stmt = con. create Statement();
 stmt. executeUpdate ("insert into emp values (504, 'Ganesh', 28)");
 System.out.println ("Inserted ...");
 con. close();
 }
 catch (Exception e)
  {
   System.out.println (e);
  }
}
}
```

output:

```
D:\> javac InsertData.java
D:\> java InsertData
Inserted ...
```

* **program:** java application with oracle database for update data.

Updatedata.java

```
import java.sql.*;
import java.util.*;
class Updatedata
{
public static void main(String args[])
{
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:
        thin:@localhost:1521:xe", "system", "admin");
Statement stmt = con.createStatement();
stmt.executeUpdate("update emp set age=38 where id=503");
System.out.println("updated....");
con.close();
}
catch(Exception e)
{
System.out.println("Exception is:" +e);
}
}
}
```

output:

```
D:/> javac Updatedata.java
D:/> java Updatedata
Updated....
```

* **DriverManager class :-**

The DriverManager class acts as an interface between user and drivers. It keeps track of the drivers that are avaliable and handles establishing a connection between a database and the appropri-ate driver.

UNIT - IV

INTRODUCTION TO JSP

* The Anatomy Of a JSP Page

A JSP page is simply a regular web page with JSP elements for generating the parts of the page that differ for each request, as shown in figure below.

```
<%@ page language="java" contentType=
                "text/html" %>
                                    JSP element
<html>
 <body> bgcolor="white">  — template text

 <jsp:useBean
   id="userInfo"
        class="com.ora.jsp.beans.userinfo.
                        UserInfoBean">
 <jsp:setProperty name="userInfo" property=
                        "*"/>
 </jsp:useBean>
                                    JSP element

The following information was saved:
<ul>                                — template text
  <li>User Name :

  <jsp:getProperty name="userInfo"
                property="userName"/>  — JSP element

  <li> Email Address :              — template text

  <jsp:getProperty name="userInfo"
                property="emailAddr"/>  — JSP element
```

```
  </ul>
 </body>
</html>
```

— template text

Everything in the page that is not a JSP element is called template text. Template text can really be any text : HTML, WML, XML or even plain text. Since HTML is by far the most common web page language in use today, most of the descriptions and examples in this book are HTML - based, but keep in mind that JSP has no dependency on HTML ; it can be used with any markup language. Template text is always passed straight through to the browser. When a JSP page request is processed, the template text and the dynamic content generated by the JSP elements are merged, and the result is sent as the response to the browser.

* JSP Processing :

JSP pages can be processed using JSP container only. Following are the steps that need to be followed while processing the request for JSP page –

1) Client makes a request for required JSP page to the server. The server must have JSP container so that JSP request can be processed. For instance : Let the client makes a request for xyz.jsp page.

2) On receiving this request the JSP container searches and then reads the desired JSP page. Then this JSP page is straight away converted to corresponding servlet. Basically any JSP page is a combination of template text and JSP element. Every template text is translated into corresponding println statement.

For instance :

```
<html>
<head>
<title>Demo</title>
...
```

→

```
out.println("<html>");
out.println("<head>");
out.println("<title>Demo
                    </title>");
...
```

Every JSP element is converted into corresponding Java code. This phase is called translation phase. The output of translation phase is a servlet.

For example : our xyz.jsp gets converted into xyzServlet.java

3) This servlet is compiled to generate the servlet class file. This phase is called request processing phase.

4) The JSP container thus executes the servlet class file.

5) A requested page is then returned to the client as a response.

* Declarations

The JSP page that we write is turned into class definition. So when we declare a variable or method in JSP inside Declaration Tag. We can declare static member, instance variable and methods inside Declaration Tag.

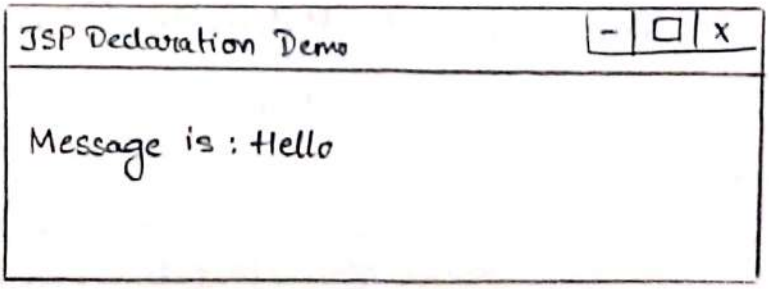
Syntax of Declaration Tag:

```
<%! Declaration code %>
```

Example:

```
<html>
  <head>
    <title> JSP Declaration Demo </title>
  </head>
  <%!
     String msg = "Hello";
  %>
  <body>
    Message is:
      <% out.println(msg); %>
  </body>
</html>
```

Output:



The above JSP code contains the declaration within <%! %> tag.

* We can declare a function of a method in JSP just similar to variable. Following JSP example illustrates the use of function declaration and definition.

MethodDemo.jsp

```
<%@ page language="java" contentType=
                                "text/html" %>
    <%!
        String msg="Hello";
    %>
    <%! public String MyFunction(String msg)
    {
        return msg;
    }
    %>
<html>
  <head>
    <title>Use of Method</title>
  </head>
  <body>
    <% out.println("Before function call : " +
                                    msg); %>

    <br/>
    After function call : <% = MyFunction("Technical
                                    Publications") %>

  </body>
</html>
```

Output :

Use of Method

Before function call : Hello

After function call : Technical Publications

JSP – Directives

Directives in JSP provide directions and instructions to the container, telling it how to handle certain aspects of the JSP processing.

* A JSP directive affects the overall structure of the servlet class. It usually has the following form –

```
<%@ directive attribute = "value" %>
```

* Directives can have a number of attributes which you can list down as key-value pairs and seperated by commas.

* The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

S.No.	Directive & Description
1.	<%@ page ... %> Defines page-dependent attributes, such as scripting language, error page and buffering requirements.
2.	<%@ include ... %> Includes a file during the translation phase.
3.	<%@ taglib ... %> Declares a tag library, containing custom actions, used in the page

JSP – The page Directive :

The page directive is used to provide instructions to the container. These instructions pertain to the current JSP page. You may code page directives anywhere in your JSP page. By convention, page directives are coded at the top of the JSP page.

Syntax :

```
<%@ page attribute = "value" %>
```

Attributes :

S.No.	Attribute & purpose.
1.	buffer specifies a buffering model for the output stream.
2.	autoFlush controls the behavior of the servlet output buffer.
3.	contentType Defines the character encoding scheme.
4.	extends specifies a superclass that the generated servlet must extend
5.	language Defines the programming language used in the JSP page.
6.	session specifies whether or not the JSP page participates in HTTP sessions

The Include Directive :

The 'include' directive is used to include a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code the 'include' directives anywhere in your JSP page.

* The general usage form of this directive is as follows -

```
<%@ include file = "relative url">
```

* The filename in the include directive is actually a relative URL. If you just specify a filename with no associated path, the JSP compiler assumes that the file is in the same directory as your JSP.

* You can write the XML equivalent of the above syntax as follows -

```
<jsp:directive.include file = "relative url" />
```

The taglib Directive :

The JavaServer Pages API allow you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.

* The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides

means for identifying the custom tags in your JSP page.

* The taglib directive follows the syntax given below -

```
<%@ taglib uri="uri" prefix="prefixOfTag">
```

* Here, the uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions

```
<jsp:directive.taglib uri="uri" prefix =
                        "prefixOfTag" />
```

Expressions

The expression tag is used to represent the expression in JSP page.

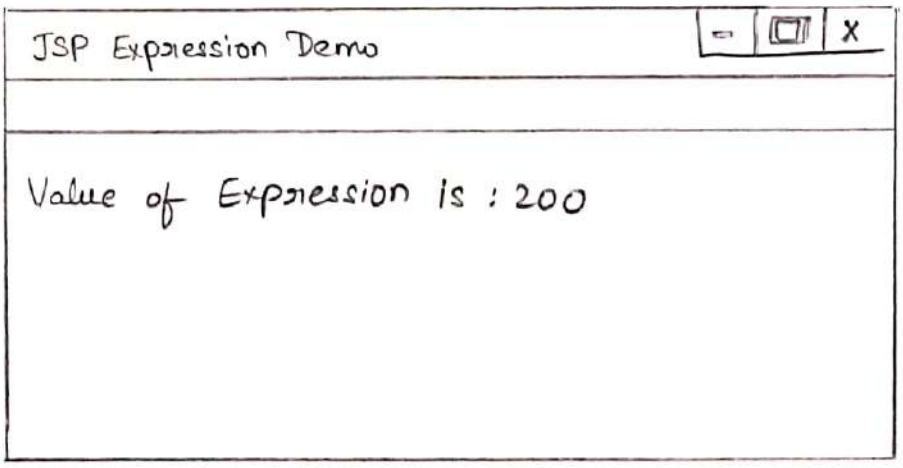
Syntax of writing expression:

```
<% = Java Expression %>
```

Example:

```
<html>
  <head>
    <title> JSP Expression Demo </title>
  </head>
  <body>
    Value of Expression is :
    <% = (10*20) %>
  </body>
</html>
```

Output:



* Code Snippets

The code that appears between the <%. and %.> delimiters is called a scriptlet. Scriptlets are nothing but java code enclosed within <%. and %.> tags

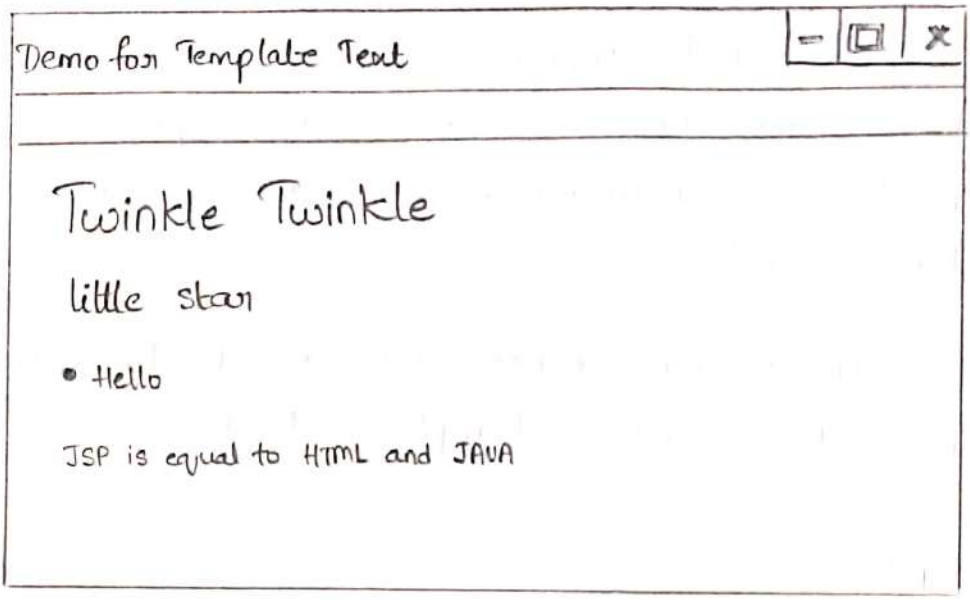
* Every Thing other than a JSP statement in the JSP is called template text.

Example :

TemplateText.jsp

```
<%.@ page language="java" contentType =
                        "text/html" %.>
<html>
  <head>
    <title>Demo for Template Text</title>
  </head>
  <body bgcolor="gray">
    <h1> Twinkle Twinkle </h1>
    <h2> little star </h2>
    <li> hello </li>
    <p>
      <%. out.println("JSP is equal to HTML
                        and JAVA"); %.>
    </p>
  </body>
</html>
```

Output :



* JSP – Implicit Objects:

The Implicit Objects are the Java objects that the JSP container makes available to the developers in each page and the developer can call them directly without being explicitly declared.

* Following table lists out the nine Implicit objects that JSP supports –

• request	This is the HttpServletRequest object associated with the request
• response	This is the HttpServletResponse object associated with the response to the client
• out	This is the PrintWriter object used to send output to the client
• session	This is the HttpSession object associated with the request
• application	This is the ServletContext object associated with the application context
• config	This is the ServletConfig object associated with the page.
• page Context	This encapsulates use of server-specific features like higher performance JspWriters
• page	This is simply a synonym for this, & is used to call the methods defined by the translated servlet class

• Exception	The Exception object allows the exception data to be accessed by designated JSP.

* Using Beans in JSP Pages:

A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications.

* Following are the unique characteristics that distinguish a JavaBean from other Java classes –

- It provides a default, no-argument constructor.
- It should be serializable and that which can implement the Serializable interface.
- It may have a number of properties which can be read or written.
- It may have a number of "getter" and "setter" methods for the properties.

JavaBeans Properties:

A JavaBean property is a named attribute that can be accessed by the user of the object. The attribute can be of any Java data type, including the classes that you define.

* A JavaBean property may be read, write, read only or write only. JavaBean properties are accessed through two methods in the JavaBean's implementation class –

S.No.	Method & Description
1.	<u>getPropertyName( )</u> For example, if property name is firstName, your method name would be getFirstName() to read that property. This method is called accessor.
2.	<u>setPropertyName ( )</u> For example, if property name is firstName, your method name would be setFirstName() to write that property. This method is called mutator.

* A read-only attribute will have only a <u>getPropertyName( )</u> method, and a write-only attribute will have only a <u>setPropertyName( )</u> method.

// Example

```
public class StudentBean implements Serializable
{
        String Rno;
        String Name;
        public void setRno(String rno)
        {
            this.Rno = rno;
        }
```

```
    public void getRno( String )
    {
        return Rno;
    }
    public void setName (String name)
    {
        this.Name = name;
    }
    public void getName( )
    {
        return Name;
    }

}
```

* There are various scopes using which the bean can be used in JSP page.

1) Page scope:

The bean object gets disappeared as soon the current page gets discarded. The default scope for a bean in jsp page is a page scope

2) Request scope:

The bean object remains in existence as long as the request object is present.

3) Session scope:

A session can be defined as a specific period of time, the user spends browsing the site.

4) Application scope:

During application scope the bean will get stored to ServletContext. Hence particular bean is available to all the servlets in the same web application.

* Application scope is the broadest scope provided by JSP and it should be used only when it is necessary.

Example

xyz.html

```
<html>
  <body>
    <form action = "abc.jsp" method = "post">
    <input type = "text" name = "text1" id = "text1">
    <input type = "password" name = "password"
                                id = "text2">
    <input type = "button" name = "submit"
                              value = "submit">
    </form>
  </body>
<html>
```

abc.jsp

```
<html>
 <body>
  <jsp:useBean id = "login" class = "ValidateBean" />
  <jsp: setProperty name = "login" property = "user"/>
  <jsp: setProperty name = "login" property = "pass" />
  You entered username as : <jsp: getProperty
                  name = "login" property = "user" />
  You entered password as: <jsp: getProperty
                  name = "login" property = "pass" />
  You are a <% login.validate ("naveen", "cse")
                  %> user <br>
```

```
</body>
</html>
```

ValidateBean.java

```
class ValidateBean implements Serializable
{
    String Name;
    String Pass;
    public void setName(String name)
    {
        this.Name = name;
    }
    public void getName()
    {
        return Name;
    }
    public void setPass(String pass)
    {
        this.Pass = pass;
    }
    public void getPass()
    {
        return Pass;
    }
    public String Validate(String s1, String s2)
    {
        if (s1.equals(name)&& s2.equals(pass))
            return valid;
        else
            return invalid;
    }
}
```

* Using Cookies

Cookies are the small text files that are stored in the client's computer.

* These are basically used to keep track of the users who browse the web. The information stored in the cookie is generally name, age, id, city and so on.

* The server script sends a set of cookies to the browser. The browser stores this information on the local machine and makes use of this information next time when the browser is browsing the web.

* Cookies are usually set in HTTP header.

* Various methods used in handling the cookies are -

1) Create Cookie
2) Read Cookie
3) Delete Cookie

1) Create cookie:

step 1 : In JSP the cookie is created using the constructor named Cookie. It requires two parameters — name and value.

Example —

```
Cookie cookie = new Cookie("name", "value");
```

Step 2 : Then we can set the validity period for the cookie using the method setMaxAge. For example to set the cookie alive for 24 hr we will write the code as

```
cookie.setMaxAge(60*60*24);
```

Step 3 : Now our cookie is ready to send over. We can add the cookie in HTTP response header as follows

```
response.addcookie(cookie);
```

2) Read Cookie :

Step 1 : First the cookie is retrieved using getCookies() method.

```
Cookie[] cookies = request.getCookies();
```

Step 2 : Then using getName() and getValue() methods the cookies are read.

3) Delete cookie :

Step 1 : Read the already created cookie and store it in cookie object

```
Cookie cookie = new Cookie("name", "");
```

Step 2: Then set its period of existence as 0 by setMaxAge method. This means that cookie is actually deleted.

```
cookie.setMaxAge(0);
cookie.setValue("");
```

Step 3: Add this cookie back to response header.

```
response.addCookie(cookie);
```

Cookie example

```
<body>
<%
  String str1 = request.getParameter("item");
  String str2 = request.getParameter("qty");
  String str3 = request.getParameter("add");
  String str4 = request.getParameter("list");
  if (str3 != null)
  {
      Cookie c1 = new Cookie (str1, str2);
      response.addCookie (c1);
      response.sendRedirect ("index.html");
  }
  else if (str4 != null)
  {
     cookie clientCookies[] = request.getCookies();
     for (int i=0; i<clientCookies.length; i++)
     {
         out.print ("<B>" + clientCookies[i].
                    getName()+":"+ clientCookies[i].
                    getValue()+"</B><BR>");

     }
  }
%>
</body>
```

* Session Handling in JSP

If we use a request scope and try to access the data over multiple pages, then same data can be shared by multiple pages. But sometimes we need to use same data for multiple requests. For example in Hospital management system, the patient information is entered initially only. That patient may undergo through various tests or operations. It is then not necessary for him to enter the same information over again and again. The same set of information is used by various operations in the hospital management system. In such a case the session scope is used.

* HTTP is a request-response protocol. That means when user wants to access some web page, the web browser makes request to server and server returns that page as a response.

* But at the same time HTTP is also called as a stateless protocol. That means when browser sends a request to the server, server processes it and sends the response to the browser and does not remember anything about the request. So when browser sends the same request to the server, server takes it as a new request process. So, it is required that server should keep track of the user or request made by the user. To solve this problem there are three methods used -

1. Use of Cookies
2. Embedding hidden fields in an HTML form
3. Sending URL string in response body.

* For sending information to and fro between browser and server, usually an ID is used. This ID is basically a <u>session-ID</u>. Thus session-ID is passed between the browser and server while processing the information. This method of keeping track of all the information between server and browser using session-ID is called <u>session tracking</u>.

* Connecting to database in JSP

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

a) Register the driver class
b) Creating connection
c) Creating statement
d) Executing queries
e) closing connection.

a) Register the driver class:

The forName() method of class is used to register the driver class. This method is used to dynamically load the driver class

Syntax:

```
public static void forName(String className)
                    throws ClassNotFoundException
```

Example:

```
Class.forName("com.mysql.jdbc.Driver");
```

b) Create the connection object:

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax:

```
public static Connection getConnection(String
                url, String name, String password)
```

Example:

```
Connection con = DriverManager.getConnection
                (url, user, password);
```

c) Create a Statement Object:

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax:

```
public Statement createStatement() throws
                SQLException
```

Example:

```
Statement stmt = con.createStatement();
```

d) Execute the query:

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of the table.

Syntax:

```
public ResultSet executeQuery(String sql)
                    throws SQLException.
```

Example:

```
ResultSet rs = stmt.executeQuery("select *
                                  from emp");
```

e) close the connection object:

By closing connection, object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax:

```
public void close() throws SQLException.
```

Example:

```
con.close();
```

UNIT-V
CLIENT SIDE SCRIPTING

* Introduction to JavaScript :

JavaScript is a dyamic language that executes within a browser. JavaScript code is embedded within an HTML page using the JavaScript tag. The <script> tag is used to embed JavaScript code. JavaScript code can be embedded in :

- An external file.
- The header of the page
- The body of the page.

* In this example, JavaScript is embedded within the header. As soon as the page is loaded this code is executed.

```
<html>
 <head>
   <title> JavaScript Example </title>
   <script language = " JavaScript 1.2">
    <!--
    document.write ("Hello world");
    //-->
   </script>
 </head>
</html
  <body> The body </body>
</html>
```

* The Document write method displays the text.

Hello world The body

* Notice that the JavaScript code is enclosed in HTML commented tags.

<!--
//-->

These are often used to surround JavaScript code. In older browsers JavaScript was not recognized or handled. To avoid the display of this code in a page, the browser would ignore the contents of the comment. However, in a browser that supports JavaScript the comments tags are ignored and the code is executed.

Uses of JavaScript :

* JavaScript can be used as an alternative to Java applets.

* JavaScript can get embedded in XHTML.

* Using DOM JavaScript can access and modify the properties of CSS (cascading Style sheets) and contents of XHTML document.

* JavaScript can be used to detect the visitor's browsers and can load the page accordingly

* JavaScript can be used to create cookies.

Features of JavaScript :

1) Browser support : For running the JavaScript in the browser there is no need to use some plug-in. Almost all the popular browsers support JavaScripting.

2) It automatically inserts the semicolon at the end of the statement, hence there is no need to write semicolon at the end of the statement in Javascript.

3) Dynamic Typing: It supports, dynamic typing, that means the data type is bound to the value and not to the variable.

4) Run Time Evaluation: Using the 'eval' function the expression can be evaluated at run time.

5) Support for Object: JavaScript is object oriented scripting language. JavaScript has a small number of in-built objects.

6) Function Programming: In JavaScript functions are used. One function can accept another function as a parameter.

* JavaScript Variable

Variable means anything that can vary. JavaScript includes variables which hold the data value and it can be changed anytime

* JavaScript uses reserved keyword "var" to declare a variable. A variable must have a unique name. You can assign a value to a variable using equal to (=) operator when you declare it or before using it.

Syntax:

```
var <variable-name>;
var <variable-name> = <value>;
```

Example: Variable declaration & Initialization

```
var one = 1; // variable stores numeric value
var two = 'two'; // variable stores string value
var three; // declared a variable without assigning a value
```

In the above example, we have declared three variables using var keyword: one, two and three. we have assigned values to variables one and two at the same time when we declared it, whereas variable three is declared but does not hold any value yet, so its value will be 'undefined'.

Declare Variables in Single line:

Multiple variables can also be declared in a single line separated by comma.

Example: Multiple Variables in Single Line

```
var one = 1, two = 'two', three;
```

Declare variable without var keyword:

JavaScript allows variable declaration without var keyword. You must assign a value when you declare a variable without var keyword.

Example: Variable without var keyword

```
one = 1;
```

* Scope of the variables declared without var keyword become global irrespective of where it is declared. Global variables can be accessed from anywhere in the webpage.

Loosely-typed Variables:

C# or Java has strongly typed variables. It means variable must be declared with a particular data type, which tells what type of data the variable will hold.

JavaScript variables are loosely-typed which means it does not require a data type to be declared. You can assign any type of literal values to a variable i.e., string, integer, float, boolean etc..

Example : Loosely Typed Variables

```
var one = 1; // numeric value
one = 'one'; // string value
one = 1.1; // decimal value
one = true ; // Boolean value
one = null; // null value
```

Primitive Types :

JavaScript defines two entities primitives and objects. The primitives are for storing the values whereas the object is for storing the reference to the actual value.

* There are following primitive types used in JavaScript.

1) Number
2) String
3) Boolean
4) Undefined
5) Null

* There are three types of predefined objects in JavaScript.

1) Number
2) String
3) Boolean

These objects are called wrapper objects. These wrapper objects provide properties and methods which can be used by primitive types.

* Scope Of Variables:

Scope in JavaScript defines accessibility of variables, objects and functions.

* There are two types of scope in JavaScript

1) Global scope
2) Local scope.

1) Global Scope:

Variables declared outside of any function become global variables. Global variables can be accessed and modified from any function.

Example: Global scope

```
<script>
    var userName = "Peter";
    function modifyUserName ( )
    {
        userName = "Steve";
    };
    function showUserName ( )
    {
        alert (userName );
    };
    alert (userName ); // display peter
    modifyUserName ( );
    showUserName ( ); //display steve
</script>
```

* In the above example, the variable userName becomes a global variable because it is declared outside of any function. A modifyUserName ( ) function modifies userName as userName is a global variable and can be accessed inside any function. The same way, showUserName ( ) function displays current value of userName variable. Changing value of global variable in any function will reflect throughout the program.

* Note that variables declared inside a function without var keyword also become global variables.

2) Local Scope:

Variables declared inside any function with var keyword are called local variables. Local variables cannot be accessed or modified outside the function declaration.

Example: Local scope

```
<script>
    function createUserName ()
    {
        var userName = "Peter";
    }
    function showUserName()
    {
        alert (userName);
    }
```

```
    createUserName();
    showUserName(); // throws error: userName
                                is not defined.
</script>
```

* In the above example, userName is local to createUserName() function. It cannot be accessed in showUserName() function or any other functions. It will throw an error if you try to access a variable which is not in the local or global scope. Use try catch block for exception handling.

No Block level Scope :

JavaScript doesnot allow block level scope inside { }. For example, variables defined in if block can be accessed outside if block, inside a function.

Example : No block level scope

```
Function NoBlockLevelScope ()
{
    if (1>0)
    {
        var myVar = 22;
    }
    alert (myVar);
}
NoBlockLevelScope ();
```

* Following are the variable scoping rules used in JavaScript.

1) Script level scope :

If a variable is declared with a var and if it is declared outside any function then it has the script level scope. This variable is also called "global variable".

2) Function level scope :

If a variable is declared with a var inside a function then it has at the function level scope. A variable with a function level scope, called "local variable".

3) Auto-declaration :

If a variable is used without the var declaration statement, it will be automatically declared with the script level scope, becoming a global variable. But using this approach of auto declaration global variables is not recommended.

4) Collission :

If a variable is explicitly defined in a function has the same name as the variable defined outside the function, then the variable outside the function cannot be accessible within this function.

* Functions

A function consists of the function keyword followed by the name of the function, a set of open and close parentheses enclosing an optional parameter list and a body enclosed in a set of curly braces.

Syntax:

```
function functionName(parameterList)
{
    // body
}
```

* A function uses a return keyword to return a value from a function.

```
<html>
  <head>
    <title> JavaScript Example </title>
    <script type = "text/javascript">
     function getHeader()
     {
        return "<h1> Main Heading </h1>"
     }
     </script>
  </head>
  <body>
    <script type = "text/javascript">
       document.write (getHeader());
    </script>
  </body>
</html>
```

o/p:

JavaScript Example
Main Heading

* Parameters are seperated by commas in the function declaration.

```
<html>
 <head>
   <title> JavaScript Example </title>
   <script type = " text / javascript ">
   function multiply (num1, num2)
   {
       return num1*num2;
   }
   </script>
 </head>
 <body>
   <script type = "text / javascript">
     document.write (multiply (2,4));
   </script>
 </body>
</html>
```

o/p:

JavaScript Example.
8

Passing an array to the function:

Similar to C or C++ we can pass an entire array as a parameter to the function. This method of array passing is called "call by reference".

Example:

```
<html>
  <head>
    <script type="text/javascript">
    function display(a)
    {
      document.write("The contents of the array
                                are .. "+"<br>");
      i=0;
      for(i in a)
      {
        document.write(a[i]+"<br>");
        i++;
      }
    }
    </script>
  </head>
  <body>
    <script type="text/javascript">
    var ar = new Array(10);
    for(i=0; i<=9; i++)
    {
      ar[i]=i;
    }
    display(ar);
    </script>
  </body>
</html>
```

Output:

Passing array to a function

The contents of the array are..

0
1
2
3
4
5
6
7
8
9

Done

* Events Handlers

Event is an activity that represents a change in the environment. For example, mouse clicks, pressing a particular key on keyboard represent the events. Such events are called "intrinsic events".

* Event Handler is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window

* Events are specified in lowercase letters and these are case - sensitive.

* The process of connecting event handler to an event is called event registration. The event handler registration can be done using two methods -

- Assigning the tag attributes.
- Assigning the handler address to object properties.

Events, Attributes and Tags :

On occurence of events the tag attribute must be assigned with some used defined functionalities. This will help to execute certain action on occurence of particular event.

Commonly used events and tag attributes are enlisted in the following table –

Events	Intrinsic event attribute	Meaning	Associated tags
change	onchange	on occurence of some change	<input> <textarea> <select>
click	onclick	when user clicks the mouse button	<a> <input>
mouseout	onmouseout	when the user moves the mouse away from some element	Form elements such as input, button, text, textarea & soon
mouse over	onmouseover	when the user moves the mouse away over some element	Form elements such as input, button, text, textarea and so on.
load	onload	After getting the document loaded	<body>
reset	onreset	when the reset button is clicked	<form>
submit	onsubmit	when the submit button is clicked	<form>
select	onselect	on selection	<input> <textarea>

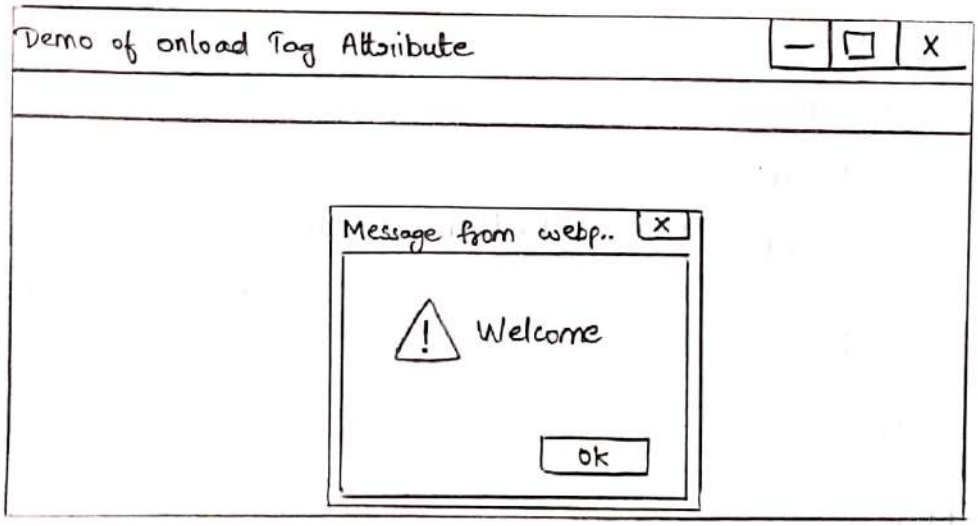
Handling Events from the Body Elements

To understand how events works in JavaScript let us put some form components on the Java Script. The onload event gets activated as soon as the web page gets loaded in the browser's window. Following script along with the output illustrate the onload tag attribute

Onload Demo.html

```
<html>
  <head>
    <script type="text/javascript">
    function my-fun()
    {
        alert("Welcome");
    }
    </script>
  </head>
  <body onload="my-fun">
  </body>
</html>
```

Output



Handling Events from Button Elements :

For handling the event using button element we have used the tag attribute onclick. The idea is that whenever we click the button some event handler must be called. This event handler can be a user defined function in which certain set of instructions get executed.

* Following is a simple JavaScript in which on the button click we have called a function my-fun(). This is a simple function in which we have displayed some message using alert popup box.

OnclickDemo.html

```
<html>
  <head>
    <title>Demo of onclick Tag Attribute </title>
    <script type = "text/javascript">
    function my-fun()
    {
       alert ("Hello I am in my function");
    }
    </script>
    </head>
    <body>
      <form>
        <input type="button" value ="click"
                                onclick = "my-fun()">
      </form>
  </body>
</html>
```

Output

* Document Object Model :

The Document Object Modeling (DOM) is for defining the standard for accessing and manipulating XHTML, XML and other scripting languages.

* Basically, DOM is an Application Programming Interface (API) that defines the interface between XHTML document and application program. That means, suppose application program is written in Java and this Java program wants to access the elements of XHTML web document then it is possible by using a set of API which belongs to the DOM.

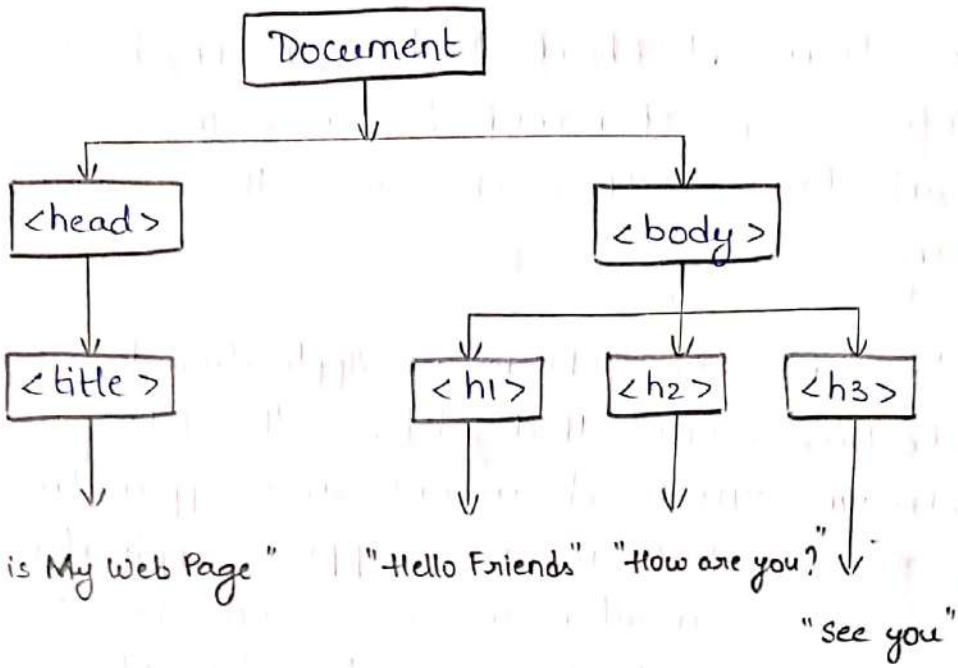
DOM Tree :

The documents in DOM are represented using a tree like structure in which every element is represented as a node. Hence the tree structure is also referred as DOM tree.

Example:

```
<html>
   <head>
      <title> This is My Web Page </title>
   </head>
   <body>
      <h1> Hello Friends </h1>
      <h2> How are you ? </h2>
      <h3> See you </h3>
   </body>
</html>
```

The DOM tree will be



* We can describe some basic terminologies used in DOM tree as follows–

1) Every element in the DOM tree is called <u>node</u>.
2) The topmost single node in the DOM tree is called the <u>root</u>.
3) Every child node must have a parent node.
4) The bottommost nodes that have no children are called <u>leaf nodes</u>.
5) The nodes that have the common parent are called <u>siblings</u>.

DOM Tree Traversal and Modification:

The main intension of objects is to use the collection of all related objects on the web page. There is a special object model collection called '<u>all</u>' which is used to refer all the HTML documents. The order in which these HTML elements come in our program in the same order all

those elements will be displayed.

Example:

```
<html>
 <head>
   <script type="text/javascript">
   var web_page_element = " ";
   function Display()
   {
     for (i=0; i<document.all.length ; i++)
         web_page_element += "<br>"+
                document.all[i].tagName;

     pmsg.innerHTML += web_page_element ;
   }
   </script>
 </head>
 <body onload="Display()">
 <p id="pmsg">Various tags used in this web
                document are : </p>

 </body>
</html>
```

Output :

Object model Demo

Various tags used in this web document are :

HTML
HEAD
SCRIPT
BODY
P

100%

* Form Validation

Form Validation is a technique which is useful in checking the validity of the input submitted by the user. One of the common technique used in form validation is password validation.

* We can use password verification process that comes along the web document. In this process user has to enter the password correctly for two times. If both the passwords are matching then the password is verified. Normally this facility is given when user creates his web account.

* In the following JavaScript we have used two textboxes which are of password type. That means whatever we type in these boxes appear in the form of dots. we will compare the entries in the two text boxes; if those are not same then the alert message will be displayed.

Example: TextDemo.html

```
<html>
 <head>
   <script type = "text/javascript">
   function my-fun()
   {
       var mypwd = document.getElementById
                                  ("pwd");
```

```
var my_re_pwd = document.getElement
                    By Id ("re_pwd");
if (mypwd.value = = " ")
{
    alert (" You have not entered the
                    password ");
    mypwd.focus ();
    return false;
}
if mypwd.value != my_re_pwd.value)
{
    alert ("Password is not verified,
                Re-enter both the passwords");
    mypwd.focus ();
    mypwd.select();
    return false ;
}
else
{
    alert ("Congratulations !!! ");
    return true ;
}
}
</script >
</head >
<body>
<form id = "form1">
<label > Enter your password
<input type = "password" value = " "
                    id = "pwd" />
</label >
< br/> < br/>
```

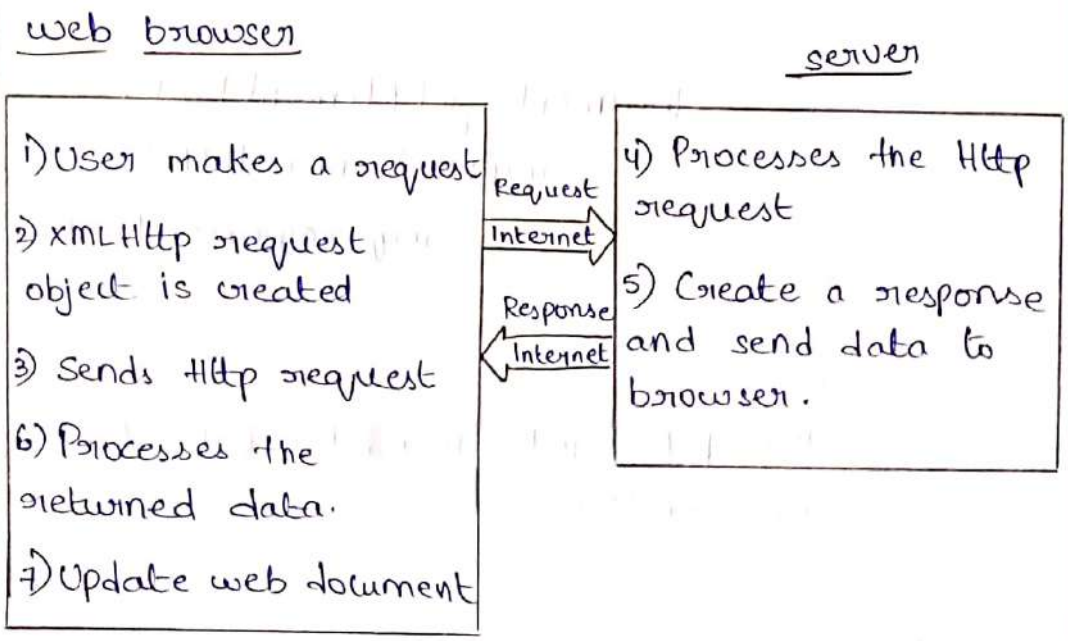
```
<label> Re-Enter the password
<input type = "password" value = ""
        id = "re_pwd" onblur = "my-fun();"/>
</label><br/>
<input type="submit" value = "submit"
        name = "submit" onsubmit =
                "my-fun();"/>
<input type = "reset" value = "Reset"
        name = "reset"/> <br/>
</form>
</body>
</html>
```

Simple AJAX Application

AJAX is a Asynchronous Java Script. It is not a new programming language but it is a kind of web document which adopts certain standards. AJAX allows the developer to exchange the data with the server and updates the part of web document without reloading the webpage.

How AJAX Works:

When user makes a request, the browser creates a object for the HttpRequest and a request is made to the server over an internet. The server processes this request and sends the required data to the browser. At the browser side the returned data is processed using JavaScript and the web document gets updated accordingly.



Let us understand how AJAX works with the help of an example.

```
<html>
 <head>
  <script type = " text / javascript " >
   function Myfun( )
   {
      if ( window . XMLHttpRequest )
      {
          req = new XMLHttpRequest ( ) ;
      }
      else
      {
          req = new ActiveXObject ( " Microsoft .
                                        req " ) ;
      }
      req . onreadystatechange = function( )
      {
          if ( req . readyState = = 4 &&
                          req . status = = 200 )
          {
              document . getElementById
                  ( " myID " ) . innerHTML =
                          req . responseText ;
          }
      }
      req . open ( " GET " , " newdata . txt " , true ) ;
      req . send ( ) ;
   }
  </script>
 </head>
```

```
<body>
<div id="myID">This text can be changed
</div>

<button type="button" onclick="MyFun()">
Change</button>

</body>
</html>
```

In above script, we have written some text which can be replaced by some another text on button click. On button click a function Myfun is invoked. In this function

1. XMLHttpRequest object is used to exchange data with a server. This object allows the user to change / update the parts of the web page without reloading it fully. The modern web browsers such as IE7+, Firefox, Chrome have built in xmLHttpRequest but old web browsers make use of ActiveXObject.

2. when a request to a server is sent, then onreadystatechange event is triggered.

3. • The readyState property holds the status of the xmLHttpRequest. The readystate =4 means request is finished and response is ready. The status=200 means "ok".

3. The request can be sent to the server by using two functions open() and send().

```
req.open("GET", "newdata.txt", true);
```

GET or POST method

Location of file on server

true: means asynchronous

false: means synchronous

* Asynchronous communication allows fast processing of the data.

* The newdata.txt file contains some updating text using which our web page can be updated.

* The send() method sends the request to the server.

AJAX with XML and PHP:

We can use Ajax along with XML and PHP. In the following example we will discuss how, HTML file along with javascript communicates with XML and PHP. It will work as follows –

1. HTML displays the form that contains a drop down list. User can select his/her friend's name from the dropdown list.

2. when user selects some name, a function named showNames will be triggered. This function is defined in javaScript file.

3. This function in javascript file will send the name as a query string to some php file. The name of the PHP file as considered as wil.

4. The PHP file make use of DOM. It will load XML file using DOM. Using DOM object we go through each node of XML file and

5. These contents are then returned to the HTML using the innerHTML. Hence on browser we can get the details of the friend whose name we have selected.

Step:1

Create an HTML document for displaying the form.

AjaxDemo.html

```
<html>
 <head>
  <script SRC = "testing.js"> </script>
 </head>
 <body>
 <form>
 Select a name :
   <select name = "names" onchange = "show
                        Names(this.value)">
   <option value = "chitra"> Chitra </option>
   <option value = "priyanka"> Priyanka
                                </option>
```

```
    <option value = "Raj"> Raj </option>
  </select>
 </form>
 <p>
 <div id = "txtHint"><b> Friend Details :
                                  </b></div>
 </p>
 </body>
</html>
```

Step : 2

The javascript will be as follows. It contains the function showNames.

testing.js

```
var xmlHttp;
function showNames (str)
{
    xmlHttp = GetXmlHttpObject();
    {
        alert (" Browser does not support HTTP
                                Request ");

        return;
    }
    var url = "getInfo.php";
    url = url + " ?q= " + str;
    url = url + "&sid = " + Math.random();
    xmlHttp.onreadystatechange = stateChanged;
    xmlHttp.open ("GET", url, true);
    xmlHttp.send (null);
}
```

```
function stateChanged()
{
    if (xmlHttp.readyState == 4 || xmlHttp.status
                                    == 200)
    {
        document.getElementById("txtHint").
                innerHTML = xmlHttp.responseText;
    }
}
function GetXmlHttpObject()
{
    var xmlHttp = null;
    try
    {
        // firefox, Opera 8.0+, Safari
        xmlHttp = new XMLHttpRequest();
    }
    catch (e)
    {
        // Internet Explorer
        try
        {
            xmlHttp = new ActiveXObject
                        ("Msxml2.XMLHTTP");
        }
        catch (e)
        {
            xmlHttp = new ActiveXObject
                        ("Microsoft.XMLHTTP");
        }
    }
    return xmlHttp;
}
```

Step : 3

The PHP script that normally runs on the server side is as given below. It will make use of DOM to load and handle the XML file.

getInfo.php

```
<?php
  $q = $_GET["q"]
  $xmlDoc = new DOMDocument();
  $xmlDoc -> load("FriendNames.xml");
  $a = $xmlDoc -> getElementsByTagName('name');
  for ($i=0; $i <= $a -> length-1 ; $i++)
  {
    if ($a -> item($i) -> nodeType == 1)
    {
     if ($a -> item($i) -> childNodes -> item(0)
                        -> nodeValue == $q)
     {
        $b = ($a -> item($i) -> parentNode);
     }
    }
  }
  $fr = ($b -> childNodes);
  for ($i=0; $i < $fr -> length ; $i++)
  {
    if ($fr -> item($i) -> nodeType == 1)
    {
      echo ("<b>". $fr -> item($i) -> nodeName."
                                  :"</b>");
      echo ($fr -> item($i) -> childNodes -> item(0)
                                  -> nodeValue);
      echo ("<br/>");
    }
?> }
```

Step: 4

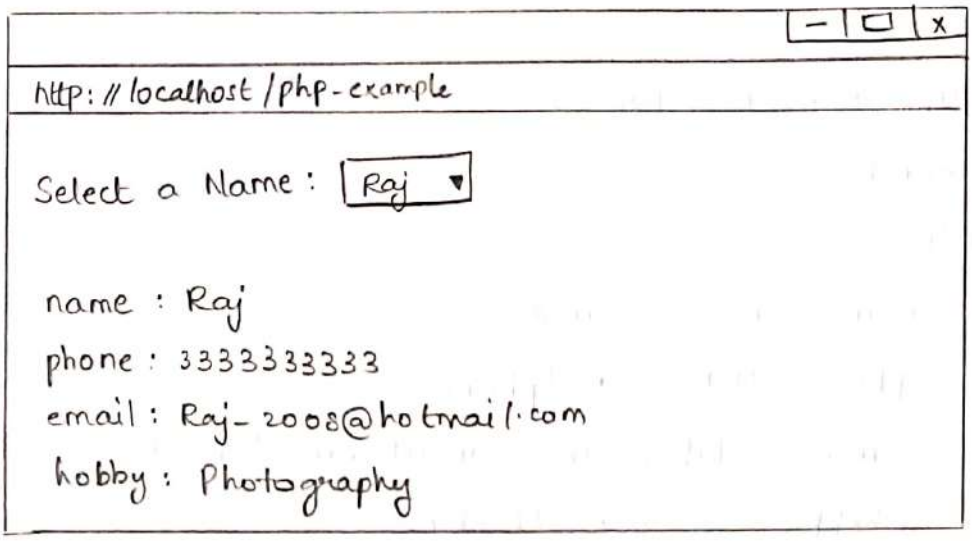
The XML file which is handled by the PHP in above step is

FriendNames.xml

```
<?xml version="1.0"?>
<Friend>
  <Info>
    <name> Chitra </name>
    <phone> 1111111111 </phone>
    <email> Chitra_abc@gmail.com </email>
    <hobby> Singing </hobby>
  </Info>
  <Info>
    <name> Priyanka </name>
    <phone> 2222222222 </phone>
    <email> Pri123@rediffmail.com </email>
    <hobby> Reading </hobby>
  </Info>
  <Info>
    <name> Raj </name>
    <phone> 3333333333 </phone>
    <email> Raj_2008@hotmail.com </email>
    <hobby> Photography </hobby>
  </Info>
</Friend>
```

Step: 5

For getting the output we will open the HTML file (created in step1) in browser window



ABSTRACT

Our project aim is to demonstrate real time stock monitor that uses the popular ESP8266 wi-fi module controlled by ARDUINO.

From today world of automation the field of biomedical is no longer aloof. Application of engineering and technology has proved its significance in the field of biomedical. It not only made doctor more efficient but also helped them in improving total process of medication.

Amid the rising number of COVID-19 cases in India, citizens are scrambling for medical oxygen, hospital beds, antiviral drugs and other supplies. Many have even put out desperate pleas on social media platforms to find COVID-related resources for their loved ones.

The goal was driven by a desire to create a ARDUINO system that connects to the internet and can work as a server/client to perform several functions and eventually serve as a central home hub. The system can be easily modified to fetch any kind of data from the internet and display it as long as there is an API for it.

The rationale behind this project was that there exists almost no library or application of the ARDUINO using the ESP8266. Both the chips individually are highly capable, cheap and can be used for even large-scale manufacture. We wanted to create a prototype ARDUINO system that has internet connectivity and can be easily extended to perform a multitude of things. Using protothreads for this makes this system only much more capable since this threading library is lightweight and makes it easy for anyone to use the ARDUINO. In order to use the keyboard without rewriting most of the configuration fie, we had to use the port expander. The alternative was to use the small board and rewrite the keyboard code. A software tradeoff is that using an intermediary python client slows down the communication (through it is more reliable). It would be slightly faster if we used the ESP to send API calls directly

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Embedded Systems

Embedded systems are designed to do some specific task, rather than be a general- purpose computer for multiple tasks. Some also have real time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

An embedded system is not always a separate block - very often it is physically built-in to the device it is controlling. The software written for embedded systems is often called firmware, and is stored in read-only memory or flash convector chips rather than a disk drive. It often runs with limited computer hardware resources: small or no keyboard, screen, and little memory.

Wireless communication has become an important feature for commercial products and a popular research topic within the last ten years. There are now more mobile phone subscriptions than wired-line subscriptions.

Lately, one area of commercial interest has been low-cost, low-power, and short-distance wireless communication used for \personal wireless networks." Technology advancements are providing smaller and more cost- effective devices for integrating computational processing, wireless communication, and a host of other functionalities.

It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works.

Fig 1.1.1 Real time embedded system examples

These embedded communications devices will be integrated into applications ranging from homeland security to industry automation and monitoring. They will also enable custom tailored engineering solutions, creating a revolutionary way of disseminating and processing information. With new technologies and devices come new business activities, and the need for employees in these technological areas. Engineers who have knowledge of embedded systems and wireless communications will be in high demand. Unfortunately, there are few adorable environments available for development and classroom use, so students often do not learn about these technologies during hands-on lab exercises.

1.2 Aim of project

Our project aim is to demonstrate real time oxygen cylinder and bed availability tracking that uses the ESP8266 wi-fi module controlled by ARDUINO.

1.3 Motivation of project

- Today, internet application development demand is very high.

  Basically, IOT is a network in which all physical objects are connected to the internet through network devices or routers and exchange data.
- IOT allows objects to be controlled remotely across existing network infrastructure.

  In our project we are going to explain how this IOT is used in the stock market for getting the information about the required product in our mobile phone.

1.4 The brief history of IOT

The internet of things (IoT) has only recently become ingrained in our everyday life. It surrounds us everywhere we go: connected cars driving on the street, home automation devices located in the house, smart office sensors embedded in the workplace, and fitness trackers worn on our bodies. Altogether, they create a massive ecosystem of 26.66 billion interconnected things, according to Statista, which hold a remarkable influence over societies and economies worldwide.

But the world hasn't always been this way. Until 1999, the term "internet of things" didn't even exist. So, how exactly did the internet of things evolve so fast and become such a regular buzzword, and what milestones marked internet of things development globally. The best way to answer these questions, let's dive into the roots of this incredible technology.

The concept of connected devices itself dates back to 1832 when the first electromagnetic telegraph was designed. The telegraph enabled direct communication between two machines through the transfer of electrical signals. However, the true IoT history started with the invention of the internet—a very essential component—in the late 1960s, which then developed rapidly over the next decades.

Fig 1.4.1 Examples of IOT

This might be hard to believe, but the first connected device was a Coca-Cola vending machine situated at the Carnegie Melon University and operated by local programmers. They integrated micro-switches into the machine and used an early form of the internet to see if the cooling device was keeping the drinks cold enough and if there were available Coke cans. This invention fostered further studies in the field and the development of interconnected machines all over the world.

1.5 Literature Survey

To understand the actual concept of IOT and related work of this seminar I have gone through various websites including Tata Institute of Fundamental Research, Mumbai. I also referred through the University Grants Commission, Department of Science and Technology (DST) and Science and Engineering Research Bord (SERB). Kachris and I. Tomkos.

According to IEEE survey to get a clear picture what currently is considered as an IoT- Service we surveyed more than, SENSEI, RUNES and OASiS and ongoing EU projects and did a comprehensive search through the ACM and IEEE literature databases.

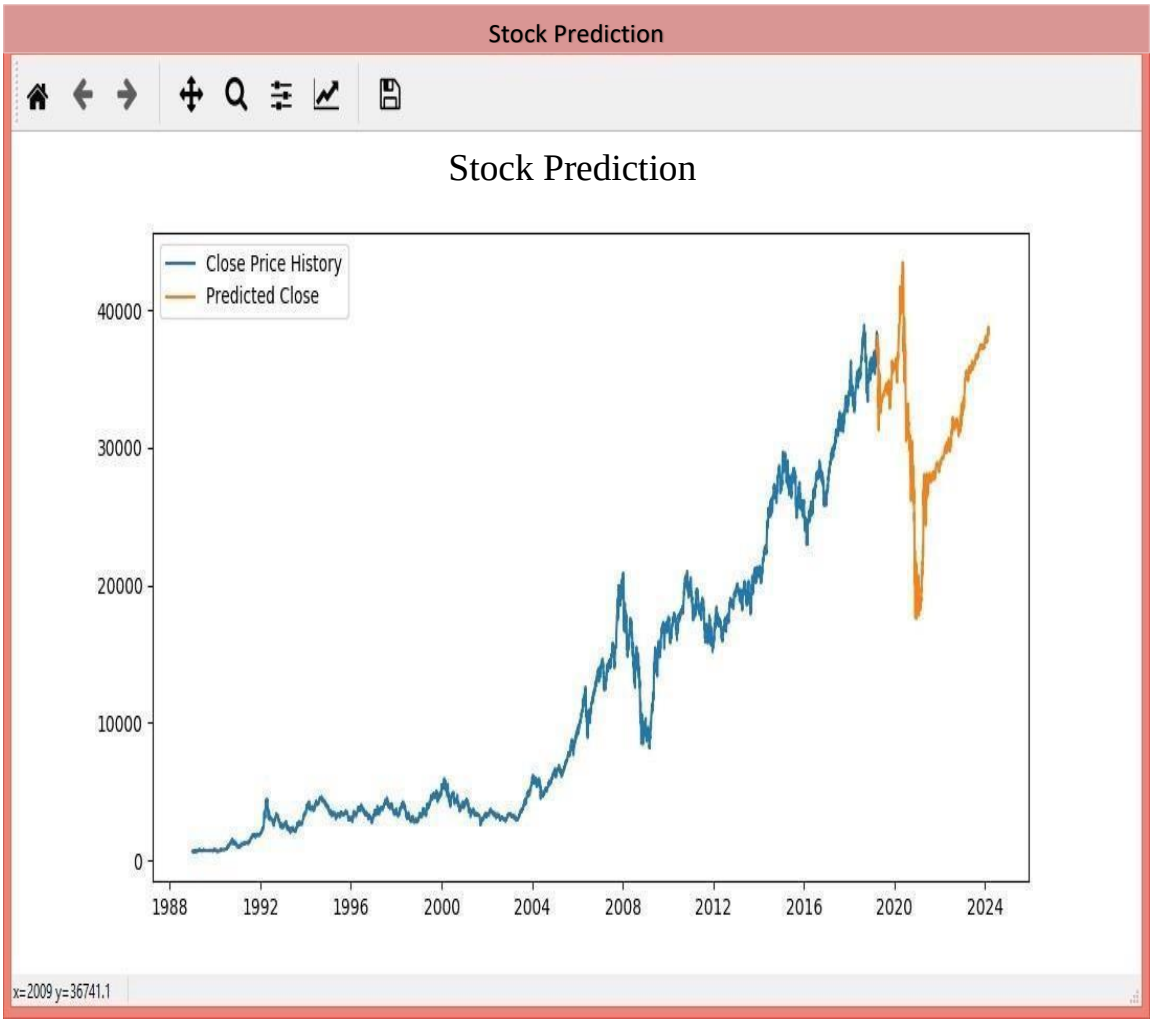


Fig 1.5.1 Oxygen cylinder & Beds monitor using machine learning

The work in by Alsheikh et al. provides a survey of ma- chine learning methods for wireless sensor networks (WSNs) … Compared to some related work in the literature that have … Communications Surveys & Tutorials IEEE COMMUNICATIONS SURVEYS & TUTORIALS.

1.6 Organization of report

The main body of the seminar report is preceded by the detailed contents including list of figures, tables and observations as the follows

Chapter 1: This chapter explains the introduction to embedded systems, aim, motivation and the brief history of IOT and literature survey of the project.

Chapter 2: This chapter describes the ARDUINO and features of the ARDUUINO

Chapter 3: This chapter describes in detail about the hardware components, Internal Sch

Chapter 4: This chapter explains the Software development, Wifi module in IOT technology, Liquid crystal display and switches with their operations.

Chapter 5: This chapter includes the advantages, disadvantages, applications, future scope and the conclusion.

Chapter 6: This chapter contains the bibliography, references and the Appendix of the project.

1.7 Conclusion

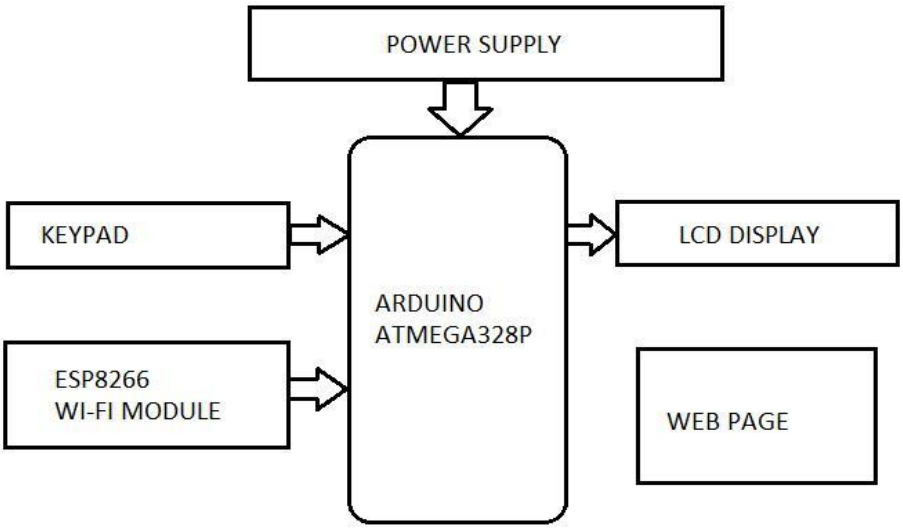
The project “Tracking of Beds and Oxygen cylinders” has been successfully designed and tested. Integrating features of all the hardware components used have developed it. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced IC’s and with the help of growing technology the project has been successfully implemented.

Chapter 2

ARDUINO

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P datasheet. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

2.1 Block Diagram



2.1 : Block Diagram

2.1.1 Hardware

- ARDUINO
- LCD
- ESP 8266
- switches

2.1.2 Software

- Embedded C
- Arduino IDE

2.2 ARDUINO UNO

Description of hardware components and its interfacing

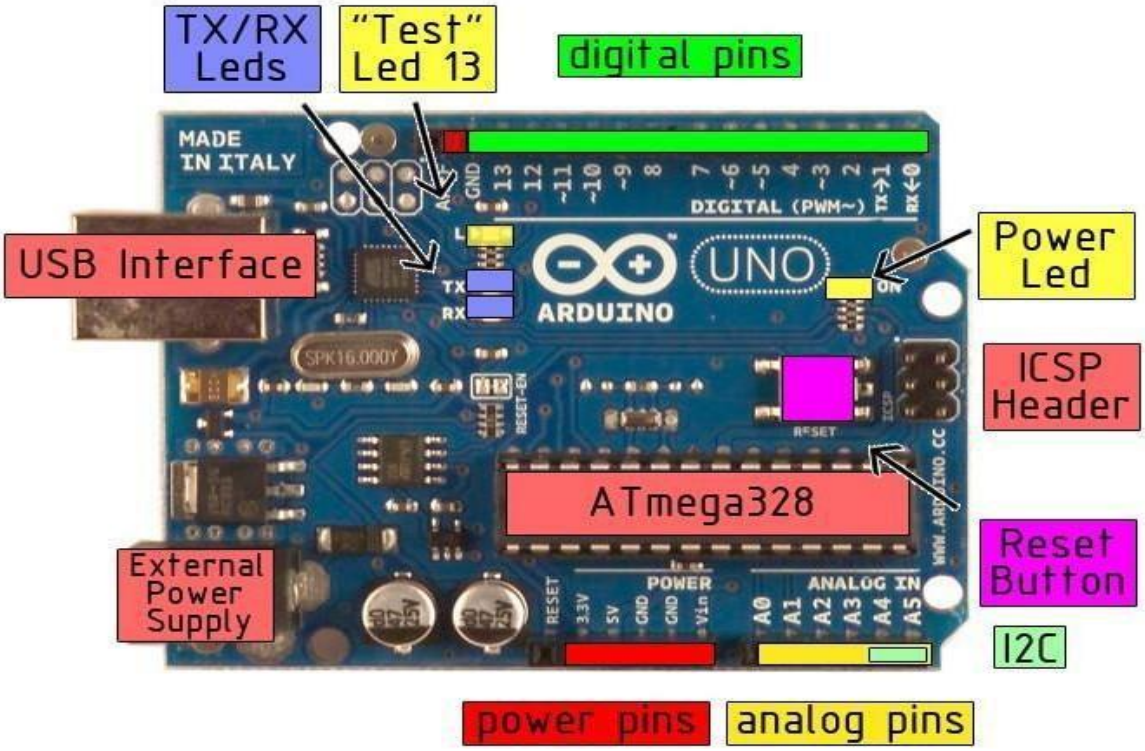


Fig 2.2: Description of Hardware Components

Fig 2.2.1: Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P datasheet. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.
You can find here your board warranty information.

You can find in the Getting Started section all the information you need to configure your board, use the Arduino Software (IDE), and start tinker with coding and electronic The Arduino/Genuino Uno can be programmed with the (Arduino Software (IDE)). Select "Arduino/Genuino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino/Genuino Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the Arduino repository.

The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy).
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an externalprogrammer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

The Arduino/Genuino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

2.1.1 Features:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

2.2.2 Comparisons with other boards

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Chapter 3

Hardware components

3.1 Internal Schematic of AT mega328P

The Atmega328P microcontroller is a 8-bit AVR RISC -based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHZ. A common alternative to the ATmega328 is the "picoPower" ATmega328P. A comprehensive list of all other members of the megaAVR series can be found on the Atmel website

- ATmega328
- ATmega328P and ATmega328P-AUTOMOTIVE
- ATmega328PB and ATmega328PB-AUTOMOTIVE (superset of ATmega328P)
  - has more UART, I2C, and SPI peripherals than ATmega328P

As of 2013 the ATmega328 is commonly used in many projects and autonomous systems where a simple, low-powered, low-cost micro-controller is needed. Perhaps the most common

implementation of this chip is on the popular Arduino development platform, namely the Arduino Uno and Arduino Nano models.

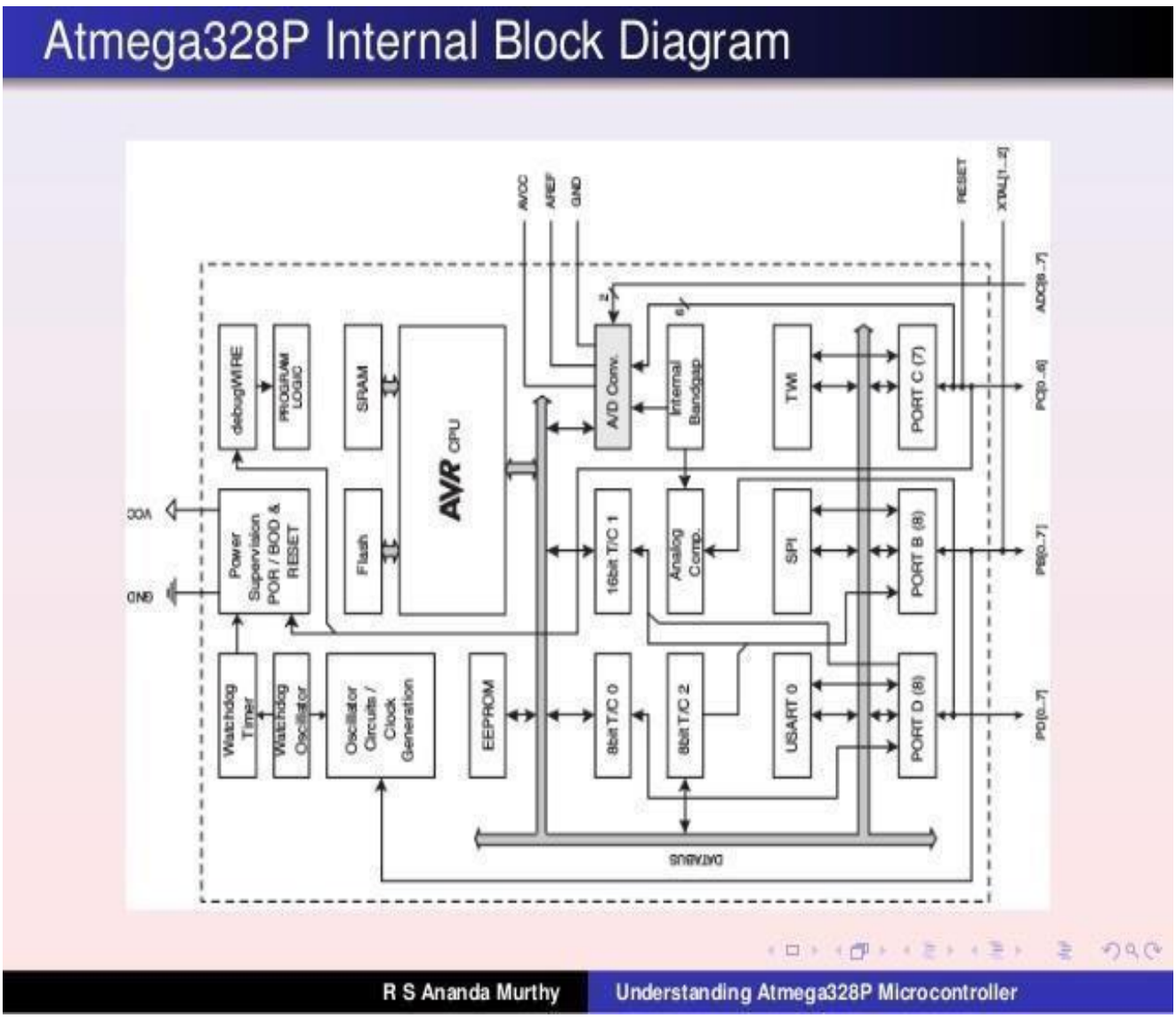


Fig 3.1: Block diagram of ATmega328P Microcontroller

Over the years the Arduino boards have been used to build thousands of projects, from daily objects to compound scientific instruments. An international community of designers, artists, students, programmers, hobbyists, and experts has gotten together around this open source stage, their donations have added up to an unbelievable amount of available knowledge that can be of immense help to beginners and specialists alike.

All boards are entirely open-source, allowing users to build them separately and finally adapt them to their exact needs. Over the years the Arduino boards have been used to build thousands of projects, from daily objects to compound scientific instruments.An international community of designers, artists, students, programmers, hobbyists, and experts has gotten together around this open source stage, their donations have added up

to an unbelievable amount of available knowledge that can be of immense help to beginners and specialists alike.

3.2 Communication:

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a.inf file is required.

The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A Software Serial library allows serial communication on any of the Uno's digital pins.The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

3.3 Automatic (Software) Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor.

When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno.

While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; for details. get a regulated positive supply from the mains supply.

3.4 Power Supply:

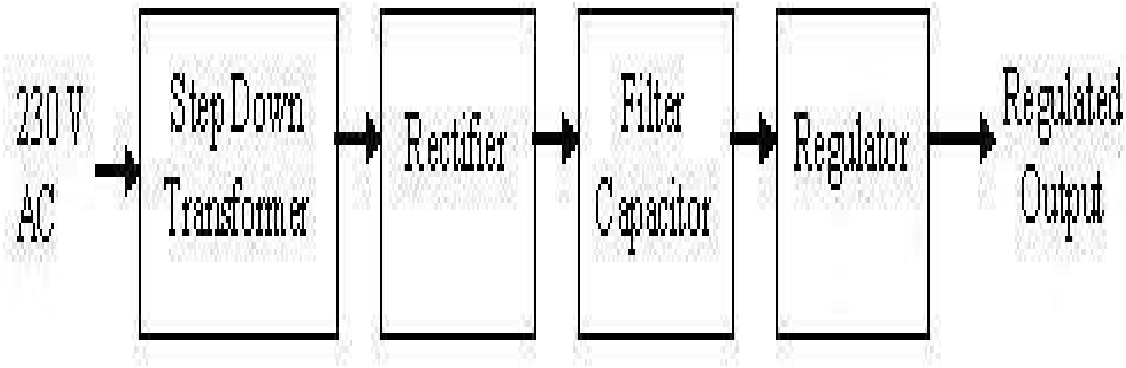


Fig 3.5.1: Basic block diagram of a fixed regulated power supply.

Let us go through each block.

- Transformer

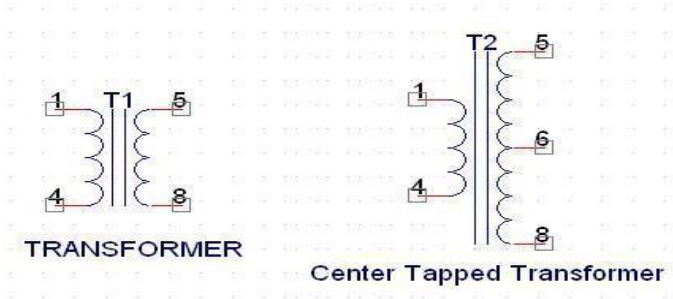


Fig 3.5.2: Transformer

A transformer consists of two coils also called as “WINDINGS” namely PRIMARY & SECONDARY. They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core & this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition

then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

So

$$I_p V_p = I_s V_s$$

The secondary voltage of the transformer depends on the number of turns in the Primary as well as in the secondary.

$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

- Rectifier

A rectifier is a device that converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. as when the anode of the diode is positive with respect to the cathode also called as forward biased condition & blocks current in the reversed biased condition.

Rectifier can be classified as follows

1) Half Wave rectifier

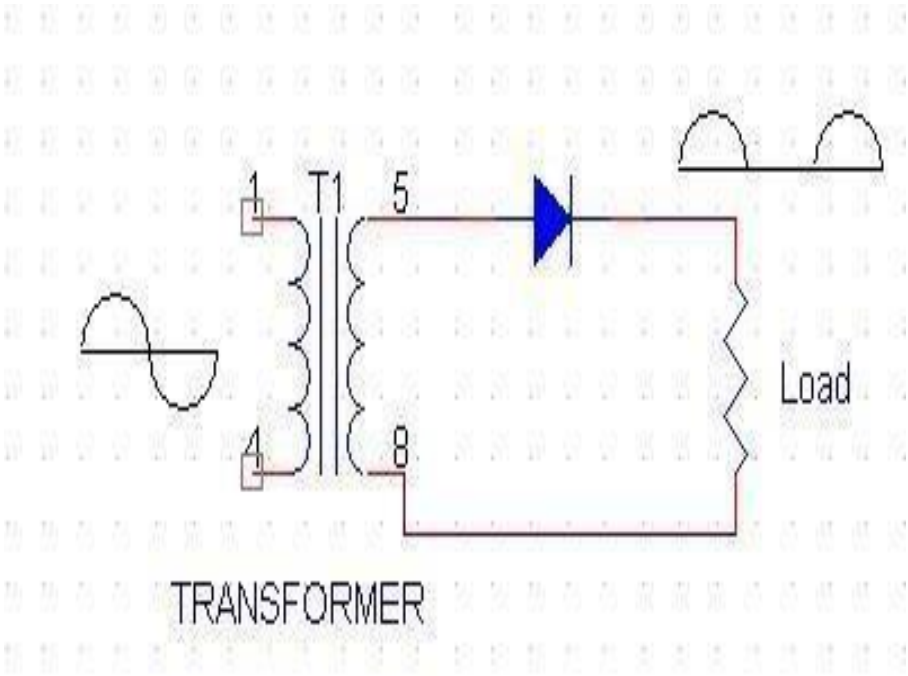


Fig 3.5.3: Half Wave Rectifier

This is the simplest type of rectifier as you can see in the diagram a half wave rectifier consists of only one diode. When an AC signal is applied to it during the positive half cycle the diode is forward biased & current flows through it. But during the negative half cycle diode is reverse biased & no current flows through it. Since only one half of the input reaches the output, it is very inefficient to be used in power supplies.

2) **Full wave rectifier**:

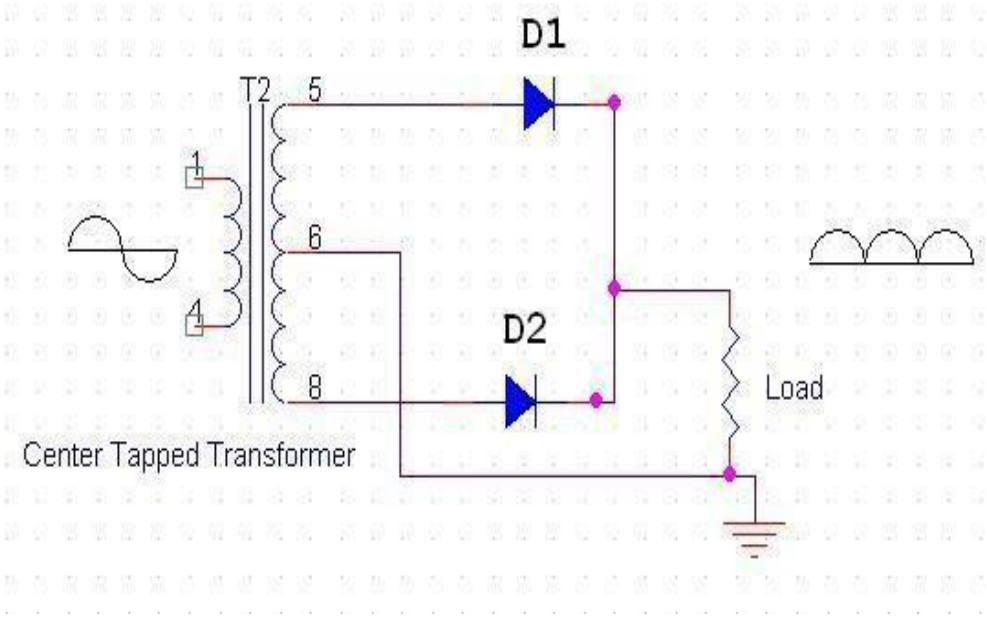


Fig 3.5.4: Full Wave Rectifier

Half wave rectifier is quite simple but it is very inefficient, for greater efficiency we would like to use both the half cycles of the AC signal. This can be achieved by using a center tapped transformer i.e. we would have to double the size of secondary winding & provide connection to the center. So during the positive half cycle diode D1 conducts & D2 is in reverse biased condition. During the negative half cycle diode D2 conducts & D1 is reverse biased. Thus we get both the half cycles One of the disadvantages of Full Wave Rectifier design is the necessity of using a center tapped transformer, thus increasing the size & cost of the circuit. This can be avoided by using the Full Wave Bridge Rectifier.

across the load.

3) **Bridge Rectifier**:

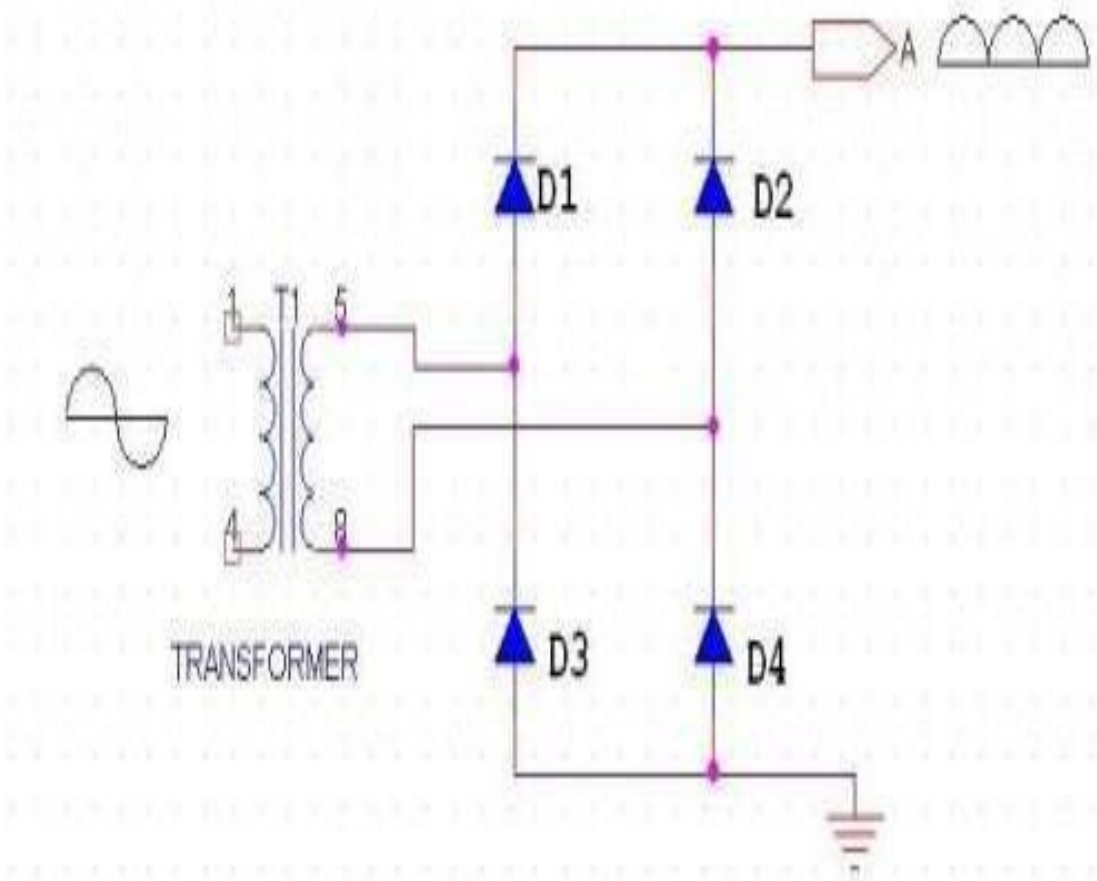


Fig 3.5.5: Bridge Rectifier

As the name suggests it converts the full wave i.e. both the positive & the negative half cycle into DC thus it is much more efficient than Half Wave Rectifier & that too without using a center tapped transformer thus much more cost effective than Full Wave Rectifier.

Full Bridge Wave Rectifier consists of four diodes namely D1, D2, D3 and D4. During the positive half cycle diodes D1 & D4 conduct whereas in the negative half cycle diodes D2 & D3 conduct thus the diodes keep switching the transformer connections so we get positive half cycles in the output.

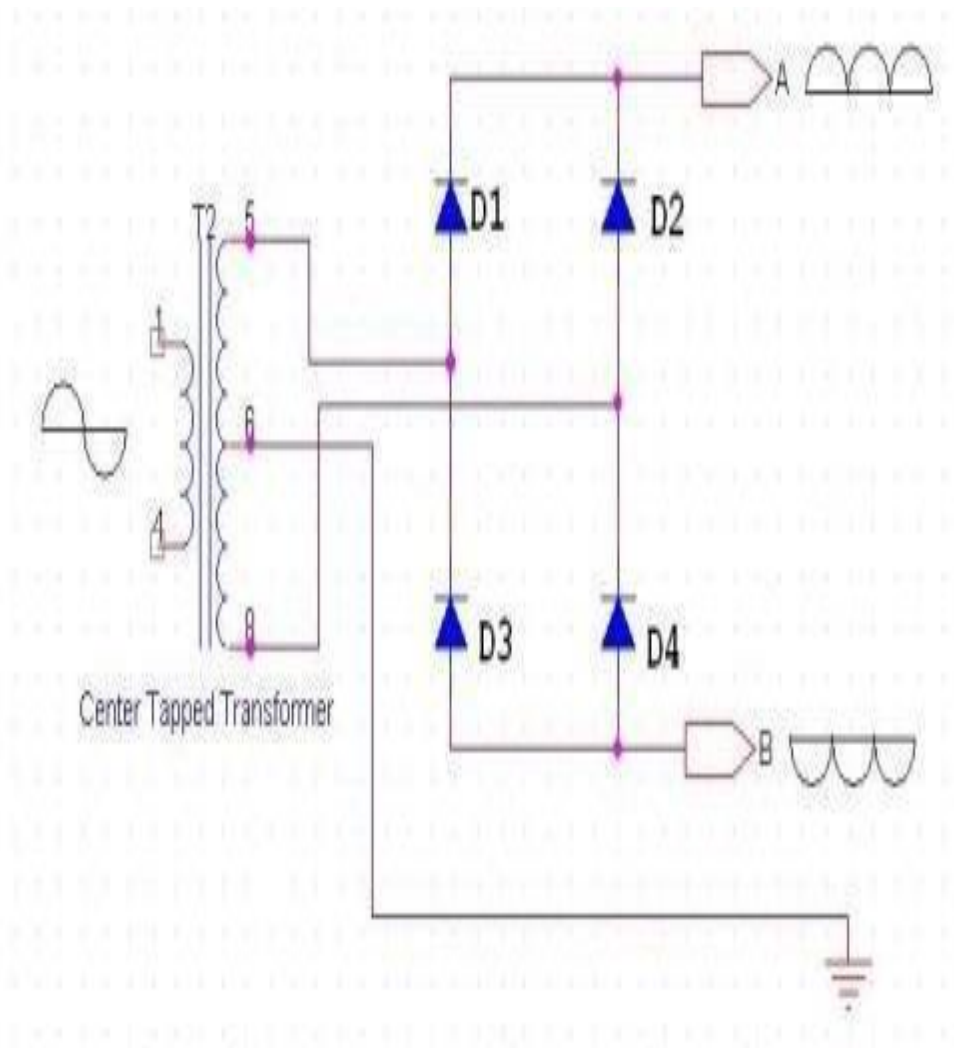


Fig 3.5.6: Bridge Rectifier with Center Trapped Transformer

If we use a center tapped transformer for a bridge rectifier we can get both positive & negative half cycles which can thus be used for generating fixed positive & fixed negative voltages.

Filter capacitor

Even though half wave & full wave rectifier give DC output, none of them provides a constant output voltage. For this we require to smoothen the waveform received from the rectifier. This can be done by using a capacitor at the output of the rectifier this capacitor is also called as "FILTER CAPACITOR" or "SMOOTHING CAPACITOR" or "RESERVOIR CAPACITOR". Even after using this capacitor a small amount of ripple will remain.

We place the Filter Capacitor at the output of the rectifier the capacitor will charge.

peak voltage during each half cycle then will discharge its stored energy slowly through the load while the rectified voltage drops to zero, thus trying to keep the voltage as constant as possible.

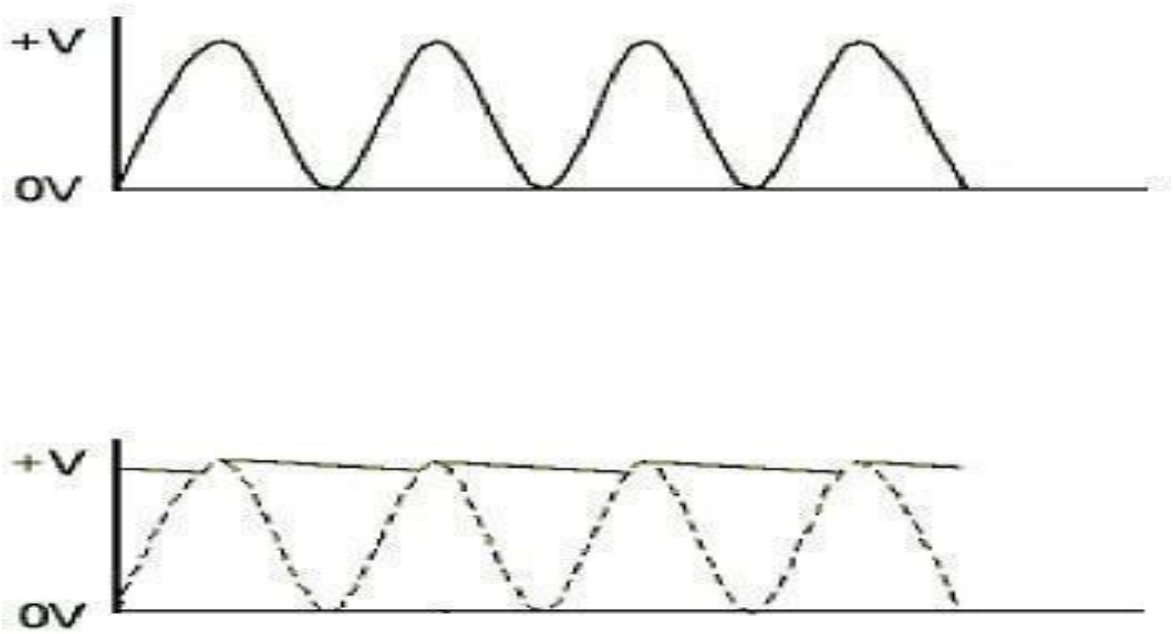


Fig 3.5.7: Output of Filter Capacitor

If we go on increasing the value of the filter capacitor then the Ripple will decrease. But then the costing will increase. The value of the Filter capacitor depends on the current consumed by the circuit, the frequency of the waveform & the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr= accepted ripple voltage.(should not be more than 10% of the voltage) I= current consumed by the circuit in Amperes.

F= frequency of the waveform. A half wave rectifier has only one peak in one cycle so F=25hz

Whereas a full wave rectifier has Two peaks in one cycle so F=100hz.

Voltage Regulator

A voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

If we go on increasing the value of the filter capacitor then the Ripple will decrease. But then the costing will increase. The value of the Filter capacitor depends on the current consumed by the circuit, the frequency of the waveform & the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr= accepted ripple voltage.(should not be more than 10% of the voltage) I= current consumed by the circuit in Amperes.

F= frequency of the waveform. A half wave rectifier has only one peak in one cycle so F=25hz

Whereas a full wave rectifier has Two peaks in one cycle so F=100hz.

Voltage Regulator

A voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types Linear Voltage Regulator:

Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.

1) Switching Regulators:

They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex & generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

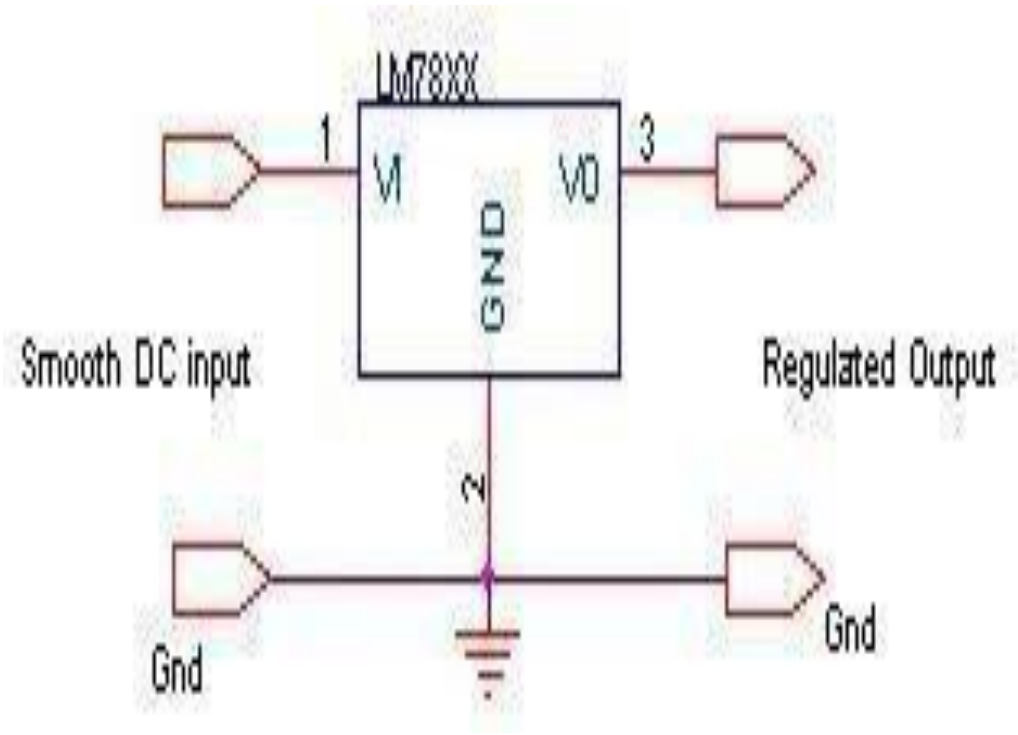


Fig 3.5.8: Switching Regulator

After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V.Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage. If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

Switching regulators convert one voltage to another by temporarily storing energy and then releasing that stored energy to the output at a different voltage. The terms DC to DC converter, switched mode power supply (SMPS), switching regulator, and switching converter all refer to the same thing. These operate by controlling a solid state device, like a transistor or diode, that acts like a switch. The switch interrupts the flow of current to an energy storage component, such as a capacitor or an inductor, in order to transform one voltage to another. There are many types of switching regulator topologies including the three most common ones:

- Buck (Step-Down) Switching Regulator

Characteristics

Line Regulation

Line regulation refers to fluctuations of the output voltage relative to the variation of the input DC voltage. This may be expressed in percentage points or a specific fluctuation in a given input range, such as 12 mV. For power supply ICs, and in particular for linear regulators, in most cases they have the same name specification. In terms of semantics, it is identical. Input voltage conditions for line regulation of a power supply are based on a presumed input voltage range of the power supply. In the case of line regulation, the property to be addressed means static output voltage fluctuations, that is, non-transient fluctuations.

Although newer power supply ICs provide excellent its line regulation performance, in terms of circuitry as a power supply, we need to look beyond IC capabilities, but also we must study the capability of input capacitor to be used to ensure sufficient line regulation.

Load Regulation

Load Regulation refers to fluctuations in the output voltage relative to the variation in load current. Similar to line Regulation, the load regulation is expressed in terms of percentage points and fluctuations between a given set of load variations. As in the case of line regulation, load

regulation specifications apply to the IC itself. However, when the IC is viewed as a power supply, we need to focus on the fact that voltage levels differ between power supply outlet and load input as the voltage declines, due to the resistive components of the output wires. At the outlet for power output, when the load current fluctuates, changes occur in a manner dependent upon the load regulation of the power circuit itself. At the load inlet, however, there is an additional decrease in voltage due to the resistance component of the interconnect. For this
reason, many situations can arise where the voltages at the power supply pins for the load requiring large currents decline unexpectedly. A more detailed discussion on this topic will be presented in the section on “Evaluating the Switching Regulator”.

One of the load fluctuations is a transient fluctuation. As in the case of line regulation, however, load regulation is not a property on transient phenomena. To address load transients, we invoke a separate concept of transient response.

Efficiency

Efficiency is defined as the ratio (%) of the output power to the input power. In simple terms, efficiency is a value that can be arrived at by measuring the power (current x voltage) pulled in at the input end and the power extracted from the output end.

While the importance of efficiency is obvious, remember that minimizing losses directly translates to reducing heat generation. Heat generation represents a critical evaluation item because not only it limits the amount of output power that can be utilized, but it also requires the space and devices for heat dissipation and cooling, and can even be a factor that reduces the reliability of power supply circuits and of add-on circuits.

Input/Output Ripple Voltage

Ripple Voltage, which refers to pulsation, occurs on both the input and output ends. On the output end, since the device of interest is a switching regulator, there always exists a ripple voltage stemming from switching operations. Although the term Switching Noise may also be used to describe Ripple Voltage, the former generally encompasses both harmonics and spikes.

In terms of ripples, the ripple voltage, which is the height of a pulse, and the frequency, need to be evaluated. In cases where a low power supply voltage, such as 1V or less, is used, as in the case of an FPGA, situations may arise where the required power supply voltage accuracy cannot be satisfied due to the ripple voltage. In addition, ripples, including harmonics and spikes, tend to reduce the system S/N.

Although output ripples can be reduced by means of an output filter, in situations where the frequency fluctuates, such as in PFM, methods for reducing the output ripple requires a careful analysis.

Input ripples arise when the switching transistor pulls in a large current by switching operations. Because spikes can occur by the switching (on/off) of the current and by the parasitic inductance of the input, elimination of spikes requires a careful circuit layout design. In concrete terms, the input capacitor should be connected right next to the input pins for the IC to eliminate parasitic inductance.

Transient Response

The transient response characteristic describes the rate of response from the time the output load current changes suddenly until the output voltage returns to the set value. Critical factors affecting the transient response characteristic include the response performance of the IC itself, in addition to the output capacitor and the equivalent serial resistance (ESR).

In the current-mode power supply IC, the transient response characteristic can be optimized by adjusting the phase characteristics. Also, hysteresis (ripple) control provides highly favorable transient response characteristics.

Allowable Dissipation

Allowable dissipation refers to the extent of direct loss that can be tolerated by the devices (ICs and transistors) used in a power supply circuit. Specifically, it means the quantity of allowable power loss that can be calculated from Tjmax (the maximum junction temperature rating) and the package thermal resistance. In the case of power elements (switching transistors), the term refers

to the allowable loss, and for built-in power devices, the term refers to the allowable loss inherent in the IC itself. In terms of circuits, because newer power devices are surface-mounted on a circuit board, in most cases the PCB can be used as a heat sink (it goes without saying that in the case of large-power circuits a separate heat sink is provided); consequently, pattern layout is an important consideration. At any rate, since thermal dissipation and allowable dissipation must be evaluated carefully, sound heat calculations are an important step.

3.5 Proposed Hardware:

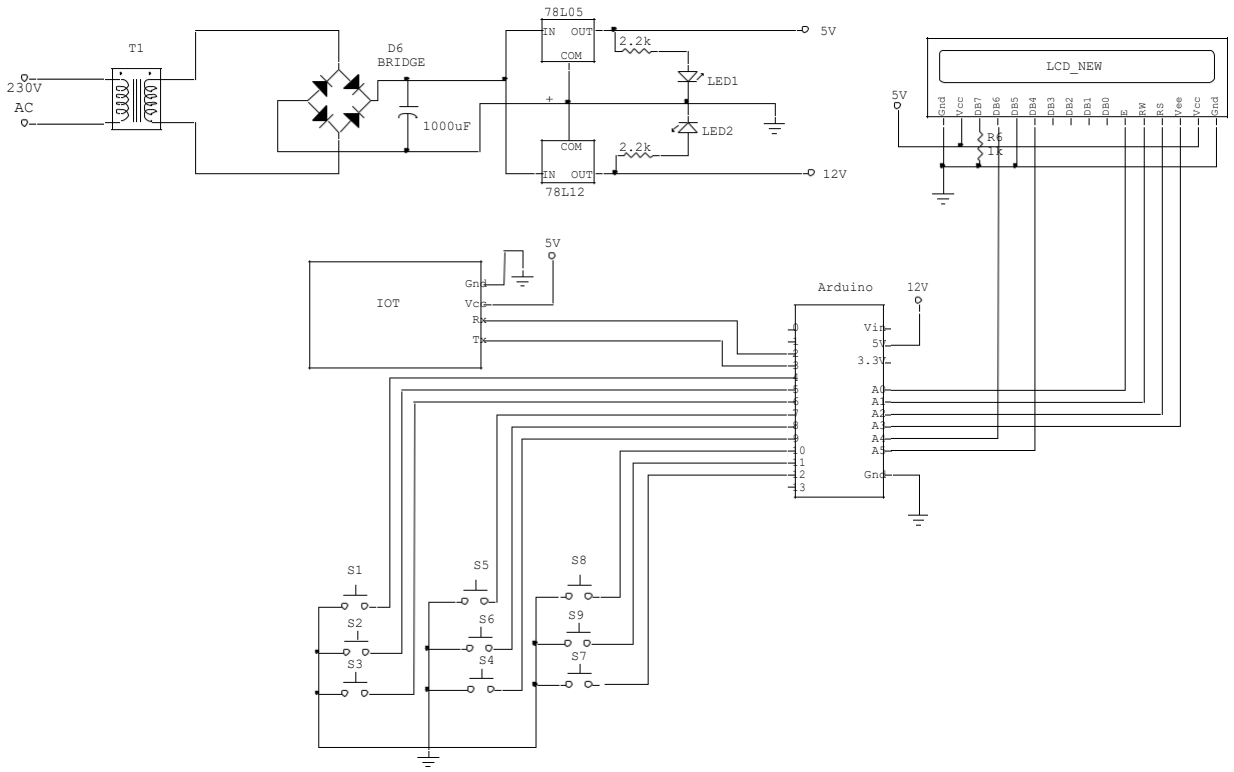


Fig 3.5.1: Proposed Hardware

7805 is an integrated three-terminal positive fixed linear voltage regulator. It supports an input voltage of 10 volts to 35 volts and output voltage of 5 volts. It has a current rating of 1 amp although lower current models are available. Its output voltage is fixed at 5.0V. The 7805 also has a built-in current limiter as a safety feature. 7805 is manufactured by many companies, including National Semiconductors and Fairchild Semiconductors.

The 7805 will automatically reduce output current if it gets too hot. The last two digits represent the voltage; for instance, the 7812 is a 12-volt regulator. The 78xx series of regulators is designed

to work in complement with the 79xx series of negative voltage regulators in systems that provide both positive and negative regulated voltages, since the 78xx series can't regulate negative voltages in such a system.

The 7805 & 78 is one of the most common and well-known of the 78xx series regulators, as it's small component count and medium-power regulated 5V make it useful for powering devices.

SPECIFICATIONS	**IC 7805**
V_{out}	5V
V_{ein} - V_{out} Difference	5V - 20V
Operation Ambient Temp	0 - 125°C
Output I_{max}	1A

Table 3.5.2: Specifications of IC7805

Fig 3.5.3 MAX 232

Fig 3.6.4: TTL/CMOS Serial Logic Waveform

The diagram above shows the expected waveform from the UART when using thecommon 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1

The diagram above shows the expected waveform from the UART when using the common 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1). A transmission starts with a start bit which is (Logic 0). Then each bit is sent down the line, one at a time. The LSB (Least Significant Bit) is sent first. A Stop Bit (Logic 1) is then appended to the signal to make up the transmission. The data sent using this method, is said to be framed. That is the data is framed between a Start and Stop Bit.

- RS-232 Voltage levels

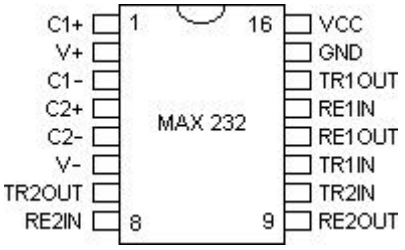
  - +3 to +25 volts to signify a "Space" (Logic 0)
  - -3 to -25 volts for a "Mark" (logic 1).
  - Any voltage in between these regions (i.e. between +3 and -3 Volts) is undefined. The data byte is always transmitted least-significant-bit first.

The bits are transmitted at specific time intervals determined by the baud rate of the serial signal. This is the signal present on the RS-232 Port of your computer, shown below.

Fig 3.5.5: RS-232 Logic Waveform

- RS-232 Level Converter: Standard serial interfacing of microcontroller (TTL) with PC or any RS232C Standard device , requires TTL to RS232 Level converter . A MAX232 is used for this purpose. It provides 2-channel RS232C port and requires external 10uF capacitors. The driver requires a single supply of +5V.

The standard specifies a maximum open-circuit voltage of 25 volts: signal levels of ±5 V, ±10 V, ±12 V, and ±15 V are all commonly seen depending on the voltages available to the line driver circuit. Some RS-232 driver chips have inbuilt circuitry to produce the required voltages from a 3 or 5 volt supply.



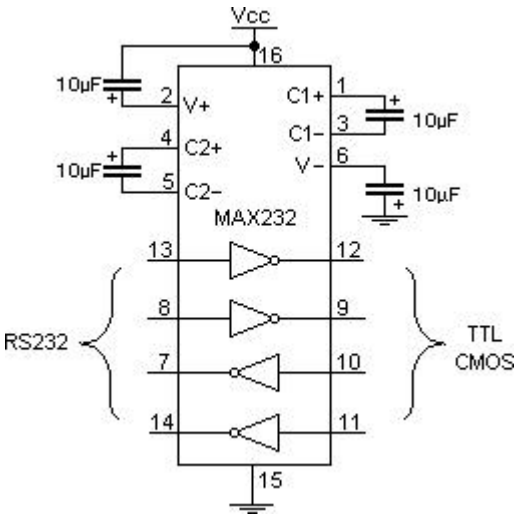


Fig 3.5.6: MAX 232 Pin description

3.6 Serial communication

When a processor communicates with the outside world, it provides data in byte sized chunks. Computers transfer data in two ways: parallel and serial. In parallel data transfers, often more lines are used to transfer data to a device and 8 bit data path is expensive. The serial communication transfer uses only a single data line instead of the 8 bit data line of parallel communication which makes the data transfer not only cheaper but also makes it possible for two computers located in two different cities to communicate over telephone.

Serial data communication uses two methods, asynchronous and synchronous. The synchronous method transfers data at a time while the asynchronous transfers a single byte at a time. There are some special IC chips made by many manufacturers for data communications. These chips are commonly referred to as UART (universal asynchronous receiver-transmitter) and USART (universal synchronous asynchronous receiver transmitter). The AT89C51 chip has a built in UART.

In asynchronous method, each character is placed between start and stop bits. This is called framing. In data framing of asynchronous communications, the data, such as ASCII characters, are packed in between a start and stop bit. We have a total of 10 bits for a character: 8 bits for the ASCII code and 1 bit each for the start and stop bits. The rate of serial data transfer communication is stated in bps or it can be called as baud rate. To allow the compatibility among data communication equipment made by various manufacturers, and interfacing standard called RS232 was set by the Electronics industries Association in 1960.

Today RS232 is the most widely used I/O interfacing standard. This standard is used in PCs and numerous types of equipment. However, since the standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible.

In RS232, a 1 bit is represented by -3 to -25V, while a 0 bit is represented +3 to +25 V, making -3 to +3 undefined. For this reason, to connect any RS232 to a microcontroller system we must use voltage converters such as MAX232 to connect the TTL logic levels to RS232 voltage levels and vice versa. MAX232 ICs are commonly referred to as line drivers.

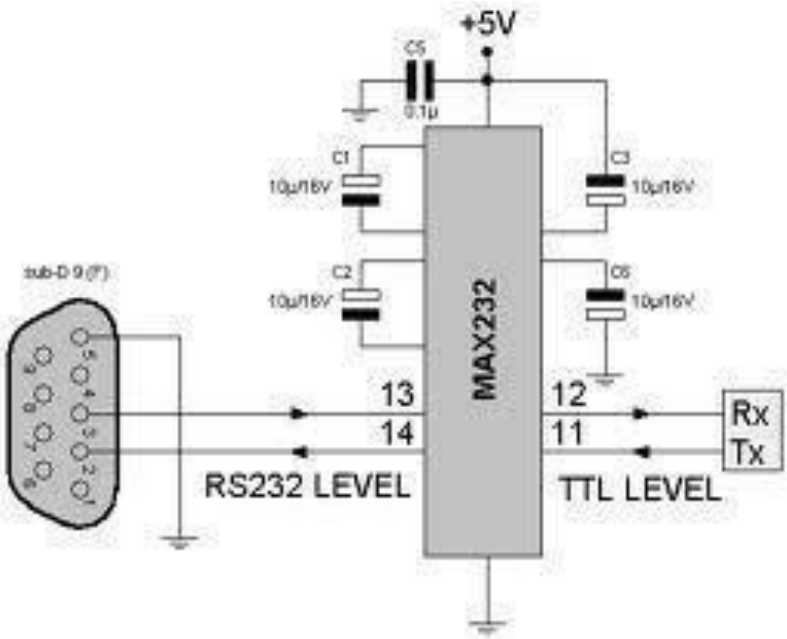


Fig 3.6.1 serial communication circuit

The RS232 cables are generally referred to as DB-9 connector. In labeling, DB-9P refers to the plug connector (male) and DB-9S is for the socket connector (female). The simplest connection between a PC and microcontroller requires a minimum of three pin,

TXD, RXD, and ground. Many of the pins of the RS232 connector are used for handshaking signals. They are bypassed since they are not supported by the UART chip.

Fig 3.6.2: DB9 Connector

IBM PC/ compatible computers based on x86(8086, 80286, 386, 486 and Pentium) microprocessors normally have two COM ports. Both COM ports have RS232 type connectors. Many PCs use one each of the DB-25 and DB-9 RS232 connectors.

The COM ports are designated as COM1 and COM2. We can connect the serial port to the COM 2 port of a PC for serial communication experiments. We use a DB9 connector in our arrangement.

3.7 Wi-fi module

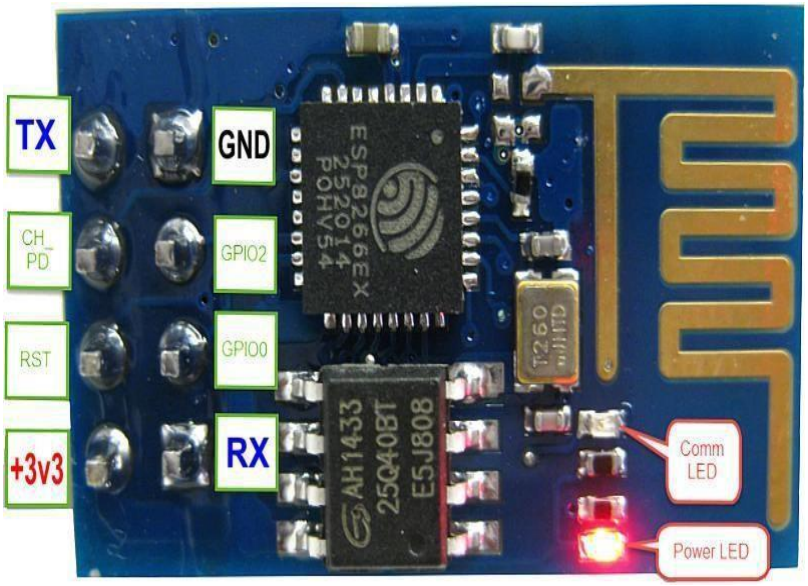


Fig 3.7.1 Wi-fi Module

3.8 Wi-fi module Features

- 802.11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
  - +19.5dBm output power in 802.11b mode
- Integrated temperature sensor
- Supports antenna diversity
- Power down leakage current of < 10uA
- Integrated low power 32-bit CPU could be used as application processor

- SDIO 2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4µs guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (Micro Controller Unit) capability produced by Shanghai-based Chinese manufacturer, The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, AI-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

However, at the time there was almost no English- language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module which suggests that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The **ESP8285** is an ESP8266 with 1 MB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi. The successor to these module(s) is ESP32.This is the series of ESP8266-based modules made by Express if.

The reason for the popularity of many of these boards over the earlier ESP-xx modules is the inclusion of an on-board USB-to-UART bridge (like the Silicon Labs' CP2102 or the WCH CH340G) and a Micro-USB connector, coupled with a 3.3-volt regulator to provide both power to the board and connectivity to the host (software development) computer – commonly referred to as the console, making it an easy development platform.

With earlier ESP-xx modules, these two items (the USB-to-serial adapter and the regulator) had to be purchased separately and be wired into the ESP-xx circuit. Modern ESP8266 boards like the NodeMCU are easier to work with and offer more GPIO pins.

Most of the boards listed here are based on the ESP-12E module, but new modules are being introduced seemingly every few months.components on the module which suggests that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The **ESP8285** is an ESP8266 with 1 MB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi. The successor to these module(s) is ESP32.This is the series of ESP8266-based modules made by Express if.

“Active pins" include the GPIO and ADC pins with which you can attach external devices to the ESP8266 MCU. The "Pitch" is the space between pins on the ESP8266 module, which is important to know if you are going to breadboard the device.

The "Form factor" also describes the module packaging as "2 x 9 DIL", meaning two rows of 9 pins arranged "Dual In Line", like the pins of DIP ICs. Many ESP-xx modules include a small on-board LED which can be programmed to blink and thereby indicate activity.

There are several antenna options for ESP-xx boards including a trace antenna, an on- board ceramic antenna, and an external connector which allows you to attach an external Wi-Fi antenna. Since Wi-Fi communications generates a lot of RFI (Radio Frequency Interference), governmental bodies like the FCC like shielded electronics to minimize interference with other devices. Some of the ESP-xx modules come housed within a metal box with an FCC seal of approval stamped on it. First and second world markets will likely demand FCC approval and shielded Wi-Fi devices.

3.9 AI-Thinker modules

Fig 3.9.1: ESP-01 module

These are the first series of modules made with the ESP8266 by the third-party manufacturer AI-Thinker and remain the most widely available. They are collectively referred to as "ESP-xx modules".

To form a workable development system they require additional components, especially a serial TTL-to-USB adapter (sometimes called aUSB-to-UART bridge) and an external 3.3 Volt power supply. Novice ESP-8266 developers are encouraged to consider larger ESP8266 Wi-Fi development boards like the Node MCU which includes the USB-to-UART bridge and a Micro-USB connector coupled with a 3.3 Volt power regulator already built into the board.

When project development is complete, you may not need these components and can consider using these cheaper ESP-xx modules as a lower power, smaller footprint option for yourproduction runs.

ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

When ESP8266 hosts the application, and when it is the only application processor in the device, it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and to minimize the memory requirements.

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any microcontroller-based design with simple connectivity through UART interface or the CPU AHB bridge interface.

The popularity of many of these "other boards" over the earlier ESP-xx modules is the inclusion of an on-board USB-to-UART bridge (like the Silicon Labs' CP2102 or the WCH CH340G) and a Micro-USB connector coupled with a 3.3 Volt regulator to provide both power to the board and connectivity to the host (software development) computer commonly referred to as the console.

With earlier ESP-xx modules, these two items (the USB-to-Serial adaptor and a 3.3 Volt regulator) had to be purchased separately and be wired into the ESP-xx circuit. Modern ESP8266 boards like the Node MCU boards are a lot less painful and offer more GPIO pins to play with. Most of these "other boards" are based on the ESP-12E module, but new modules are being introduced seemingly every few months.

3.10 Liquid Crystal Display

To display interactive messages we are using LCD Module. We examine an intelligent LCD display of two lines,16 characters per line that is interfaced to the controllers. The protocol (handshaking) for the display is as shown.

Whereas D0 to D7th bit is the Data lines, RS, RW and EN pins are the control pins and remaining pins are +5V, -5V and GND to provide supply. Where RS is the Register Select, RW is the Read Write and EN is the Enable pin.

The display contains two internal byte-wide registers, one for commands (RS=0) and the second for characters to be displayed (RS=1). It also contains a user-programmed RAM area (the character RAM) that can be programmed to generate any desired character that can be formed using a dot matrix.

To distinguish between these two data areas, the hex command byte 80 will be used to signify that the display RAM address 00h will be chosen. Port 1 is used to furnish the command or data type, and ports 3.2 to 3.4 furnish register select and read/write levels.

The display takes varying amounts of time to accomplish the functions as listed. LCD bit 7 is monitored for logic high (busy) to ensure the display is overwritten.Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose.

The display takes varying amounts of time to accomplish the functions as listed. LCD bit 7 is monitored for logic high (busy) to ensure the display is overwritten.

Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose. The most common type of LCD controller is HITACHI 44780 which provides a simple interface between the controller & an LCD. These LCD's are very simple to interface with the controller as well as are cost effective.

Fig 3.10.1: 4x20 Line Alphanumeric LCD Display

The most commonly used ALPHANUMERIC displays are 1x16 (Single Line & 16 characters), 2x16 (Double Line & 16 character per line) & 4x20 (four lines & Twenty characters per line). The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required. How do we decide which mode to use? It’s simple if you have sufficient data lines you can go for 8 bit mode & if there is a time constrain i.e. display should be faster then we

have to use 8-bit mode because basically 4-bit mode takes twice as more time as compared to 8-bit mode.

Pin	Symbol	Function
1	Vss	Ground
2	Vdd	Supply Voltage
3	Vo	Contrast Setting
4	RS	Register Select
5	R/W	Read/Write Select
6	En	Chip Enable Signal
7-14	DB0-DB7	Data Lines
15	A/Vee	Gnd for the backlight
16	K	Vcc for backlight

Table 3.10.2: Pin description of LCD

When RS is low (0), the data is to be treated as a command. When RS is high (1), the data being sent is considered as text data which should be displayed on the screen.

When R/W is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively reading from the LCD. Most of the times there is no need to read from the LCD so this line can directly be connected to ground thus saving one controller line. The ENABLE pin is used to latch the data present on the data pins. A HIGH - LOW signal is

required to latch the data. The LCD interprets and executes our command at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed.

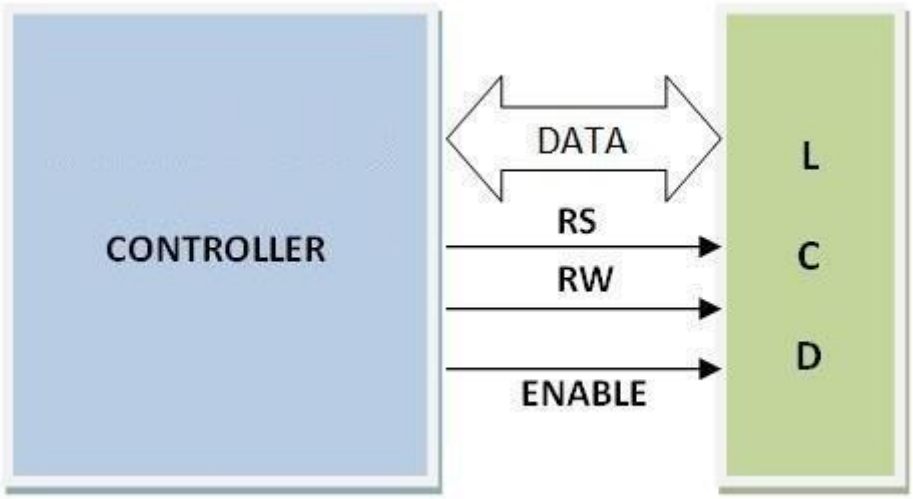


Fig: 3.10.3 latching the data

3.11 FEATURES of LCD 20*4

- These are some features of 20 x 4 LCD modules that are described here with the detailed.
- The most important feature of this module is that it can display 80 characters at a time
- The cursor of this module has 5x8 (40) dots.
- On this module already assembled the controller of RW1063.
- This module operates on the plus five volts input supply and can also work on the plus three volts.
- The plus three volts pinout can also be used for the negative supply.
- The duty cycle of this module is one by sixteen (1/16).

- The light-emitting diode of this module can get supply from the pinout one, pinout two, pinout fifteen, pinout sixteen, or pinout A and K.

Parameters	Symbol	Conditions
Input Voltage	It denoted as VDD	The value of VDD is plus five volts.
Supply Current	It denoted as IDD	Its value is ten milliamperes.
LC Driving Voltage for Normal Temperature Version Module	Its symbol is VDD to V0.	Its value is 5.2 volts
LED Forward Voltage	It is denoted as VF.	Its value is 4.3V
LED Forward Current	It denoted as IF.	Its value is 4.6V.
EL Supply Current	This pinout denoted as EL	VEL = 110 VAC, and four hundred frequency

Advantages

- These are some advantages of this module that are described with the detailed.
- It is less expensive, lightweight as compared to the cathode ray tube display.
- It uses less power according to the brightness resolution.

Switches interfaces input/output devices are critical of an embedded system. It allows to human to input binary information into the computer. Typically we define the asserted state, or logic true when the switch is pressed. Contact switches can also be used in machines to get mechanical contact.

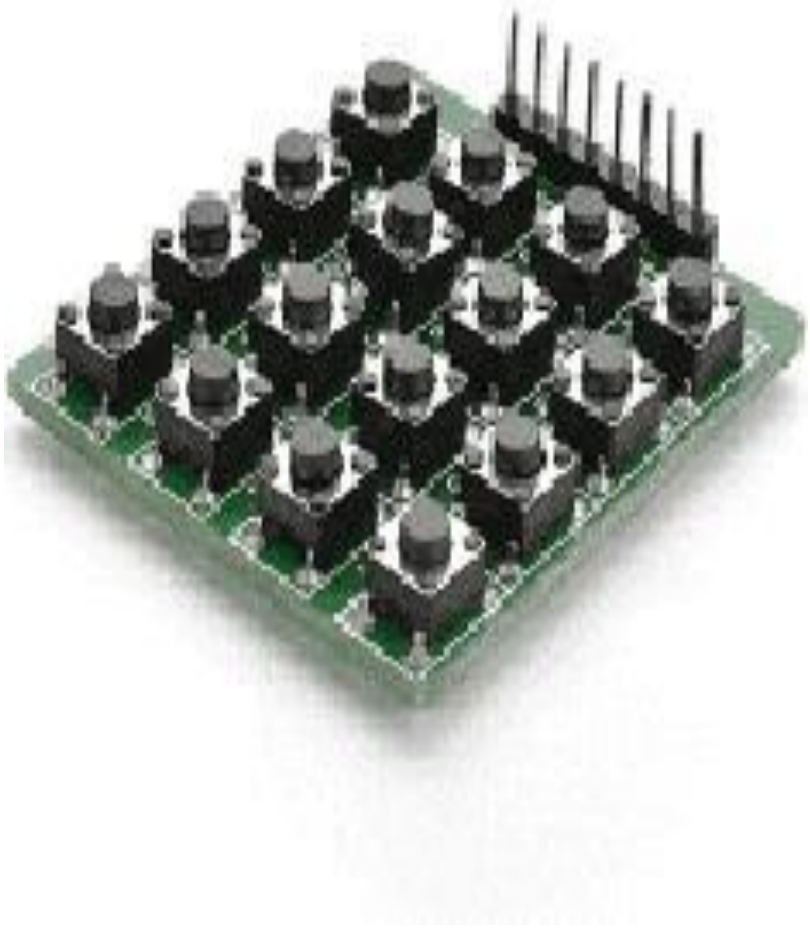


Fig 3.11.1 Switch

Chapter 4

Software Development

4.1 Software installation:

Fig 4.1.1: Software installation

4.2 Software Requirements:

- A computer (Windows, Mac, or Linux)
- An Arduino-compatible microcontroller (anything from this guide should work)
- A USB A-to-B cable, or another appropriate way to connect your Arduino- compatible microcontroller to your computer .

Fig 4.2.1: An A-to-B USB Cable

4.3 An Arduino:

If you're ready to get started, click on the link in the column on the left that matches up with your operating system, or you can jump to your operating system here.

- Windows
- Mac

Linux

Windows:

This page will show you how to install and test the Arduino software with a Windows operating system (Windows 8, Windows 7, Vista, and XP).

4.4 Windows 8, 7, Vista, and XP:

Go to the Arduino download page and download the latest version of the Arduino software for Windows.

When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.

- Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI connector if you're using an Arduino pro). You should see the an LED labelled 'ON' light up. (this diagram shows the placement of the power LED on the UNO).
- If you're running Windows 8, you'll need to disable driver signing, so go see the Windows 8 section. If you're running Windows 7, Vista, or XP, you'll need to install some drivers, so head to the Windows 7, Vista, and XP section down be

4.5 Windows 8:

Windows 8 comes with a nice little security 'feature' that 'protects' you from unsigned driver installation. Some older versions of Arduino Uno come with unsigned drivers, so in order to use your Uno, you'll have to tell Windows to disable driver signing. This issue has been addressed in newer releases of the Arduino IDE, but if you run into issues, you can try this fix first.

For a nice, step-by-step tutorial with pictures click here, otherwise the steps are outlined below.

To temporarily disable driver signing:

- From the Metro Start Screen, open Settings (move your mouse to the bottom- right-corner of the screen and wait for the pop-out bar to appear, then click the Gear icon)
- Click 'More PC Settings'
- Click 'General'
- Scroll down, and click 'Restart now' under 'Advanced startup'.
- Wait a bit.
- Click 'Troubleshoot'.
- Click 'Advanced Options'
- Click 'Windows Startup Settings'

When your computer restarts, select 'Disable driver signature enforcement' from the list.

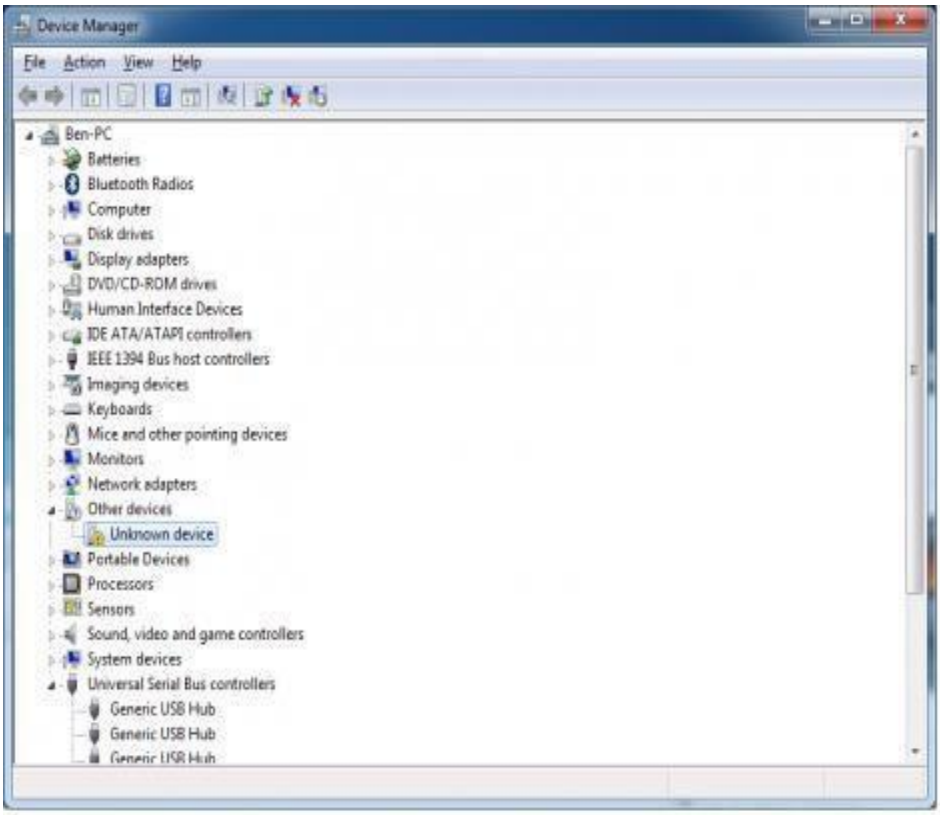
To permanently disable driver signing (recommended, but has some minor security implications):

- Go to the metro start screen
  Type in "cmd"
- Right click "Command Prompt" and select "Run as Administrator" from the buttons on the bottom of your screen
- Type/paste in the following commands: bcded-set loadoptions DISABLE_INTEGRITY_CHECKS bcdedit -set TESTSIGNING ON
- Reboot!

4.6 Windows 7, Vista, and XP:

Installing the Drivers for the Arduino Uno (from Arduino.cc)

- Plug in your board and wait for Windows to begin it's driver installation process
- After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel
- While in the Control Panel, navigate to System and Security. Next, click on System
- Once the System window is up, open the Device Manager
  - Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under 'Other Devices' for 'Unknown Device'



  - Right click on the "Arduino UNO (COMxx)" or "Unknown Device" port and choose the "Update Driver Software" option
  - Next, choose the "Browse my computer for Driver software" option

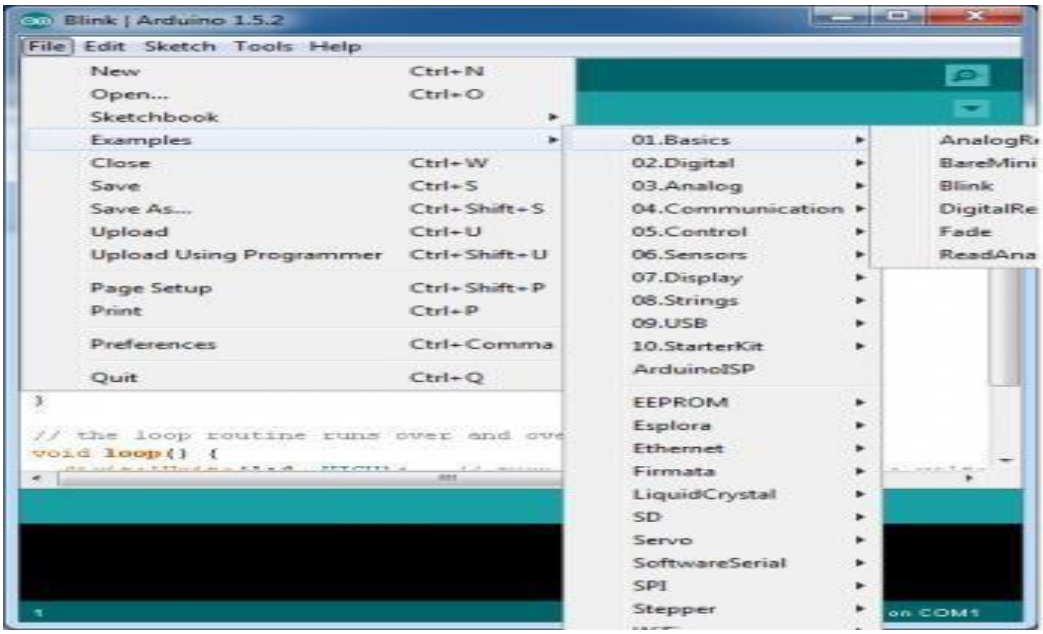
- Finally, navigate to and select the Uno's driver file, named "ArduinoUNO.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you cannot see the .inf file, it is probably just hidden. You can select the 'drivers' folder with the 'search sub-folders' option selected instead.
- Windows will finish up the driver installation from there

For earlier versions of the Arduino boards (e.g.Arduino Duemilanove, Nano, or Diecimila) check out this page for specific directions.

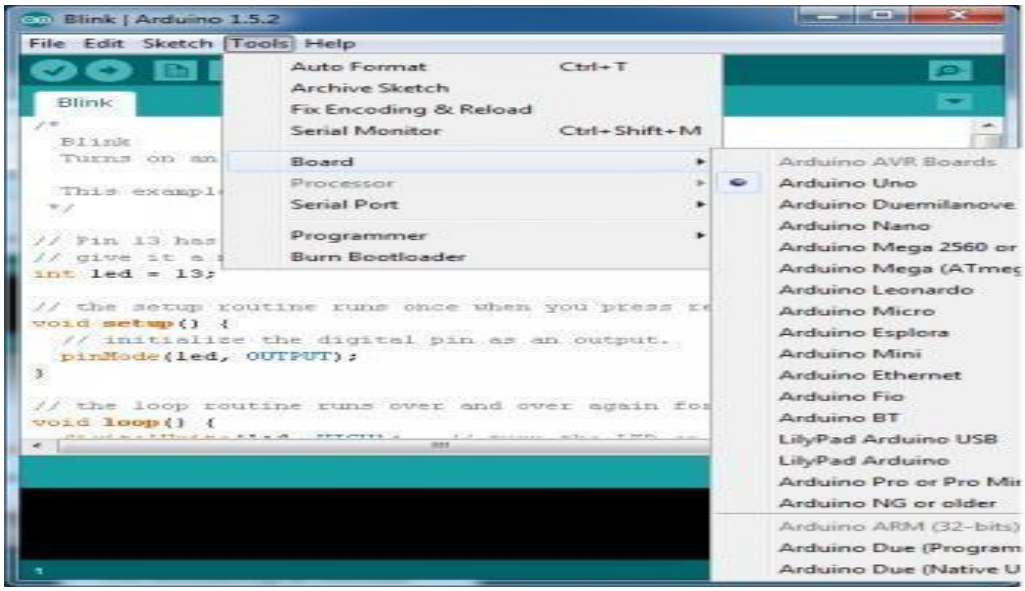
4.7 Launch and Break:

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board.

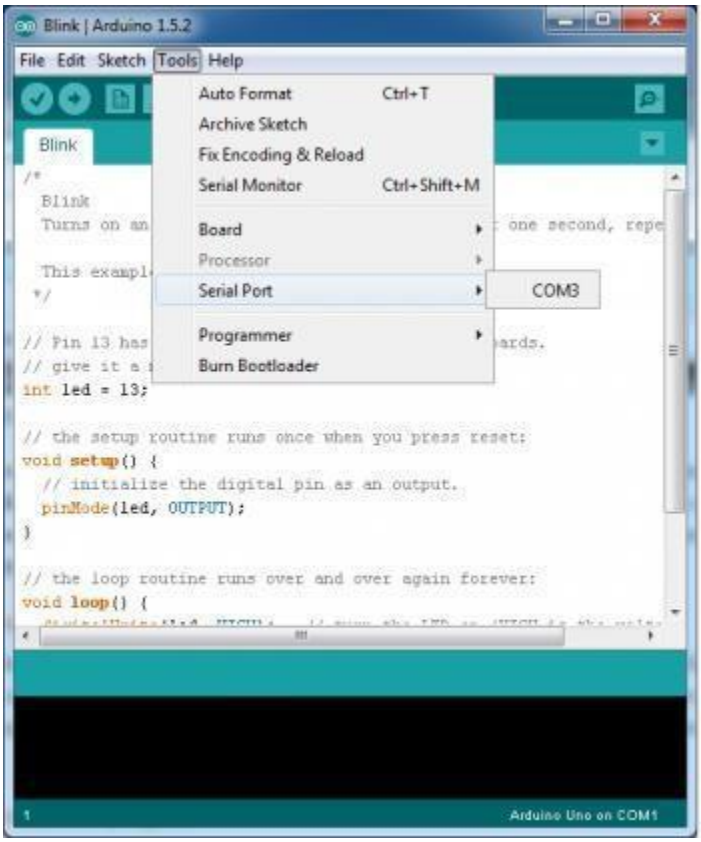
- Launch the Arduino application
- If you disconnected your board, plug it back in
- Open the Blink example sketch by going to: File > Examples > 1.Basics > Blink



Select the type of Arduino board you're using: Tools > Board > your board type



- Select the serial/COM port that your Arduino is attached to: Tools > Port > COMxx



- If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino.
- With your Arduino board connected, and the Blink sketch open, press the 'Upload' button.

- After a second, you should see some LEDs flashing on your Arduino, followed by the message ‘Done Uploading’ in the status bar of the Blink sketch.
- If everything worked, the onboard LED on your Arduino should now be blinking! You just programmed your first Arduino.

Troubleshooting

This guide from Arduino has some more details and troubleshooting tips if you get stuck.

Chapter 5

Advantages, Disadvantages, Applications, Result

5.1 Advantages:

- Predictive maintenance, rather than waiting for a machine to fail.
- Reduces the cost and complexity of operation sustain in business.
- Accessing information is easy, you can control a device that is miles apart in real time.

Fig 5.1.1 Advantages of IOT

5.2 Disadvantages:

- Lack of security on privacy
- Yields unemployment
- Today's lifestyle is technology driven, we depend on technology for the tiniest of tasks.

Fig 5.2.1 Disadvantages of IOT

5.3 Applications:

- It is used in the library to search for the required book and the count of it from the library's web page. It can be accessed by anyone from their home itself.

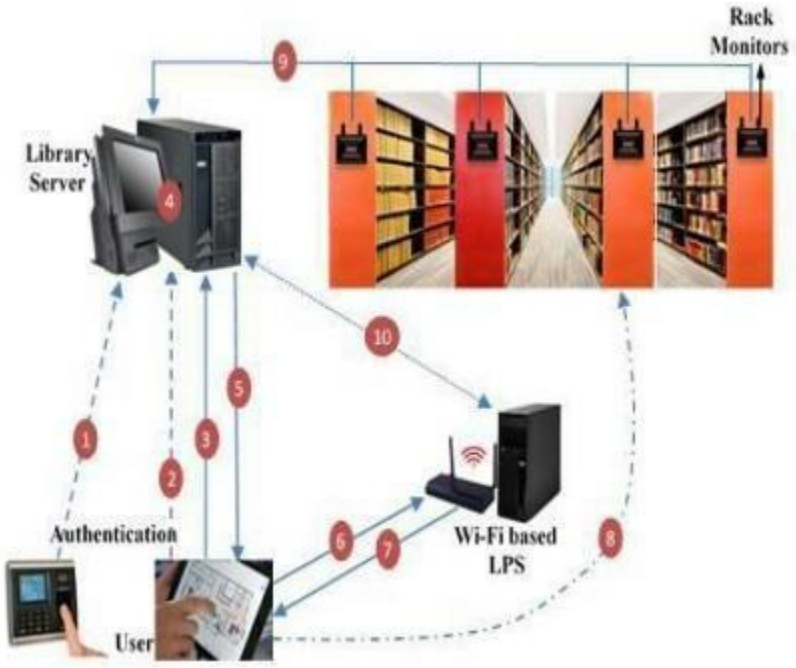


Fig 5.3.1 Application of IOT

- Similarly, we can also use this for any stock market applications such as gold shop too. Here we can verify the availability of the quantity of the gold required.
- We can apply this in the shopping malls for the required product or any accessories, in the medical shop and also in the super markets for the desired items.
- In this way, it is applied in various sectors of the stock market.

5.4 Result:

Web page gives the information about the required items list from the stock market. We can also check the count of the required gold, silver and bronze material through the LCD display as follows:

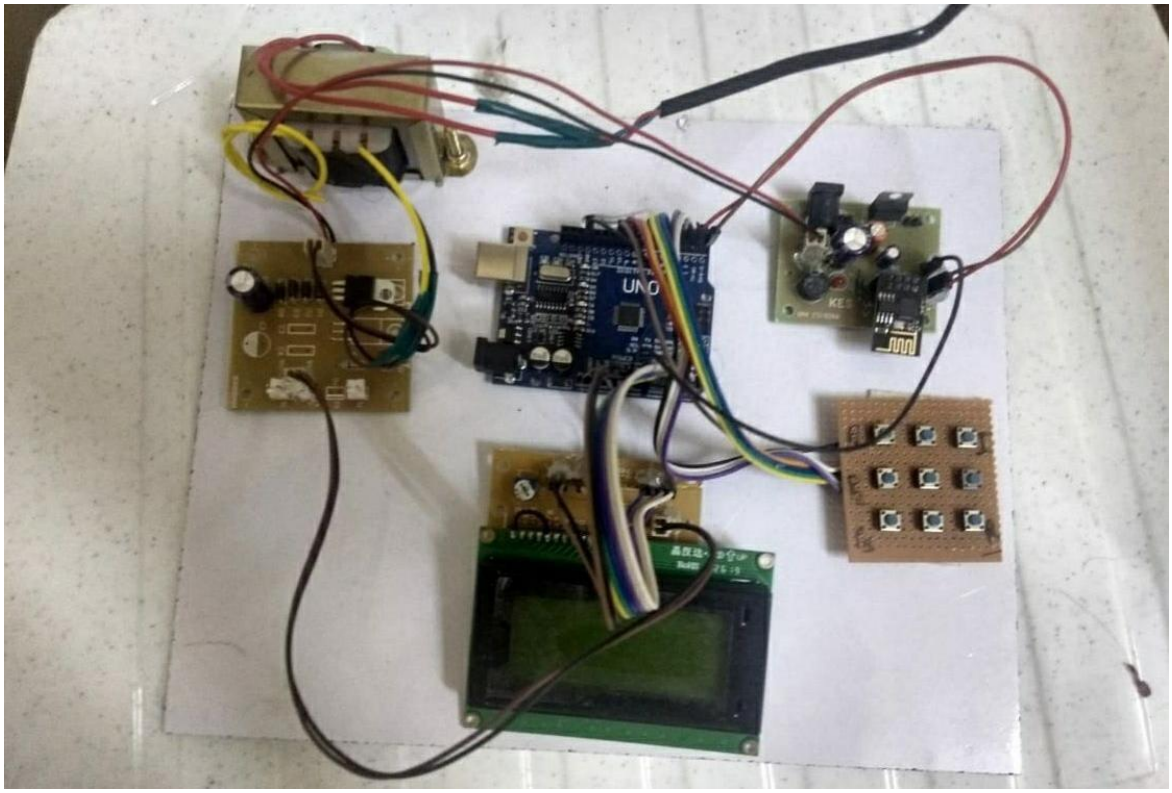


Fig 5.5.1 Final hardware components connection

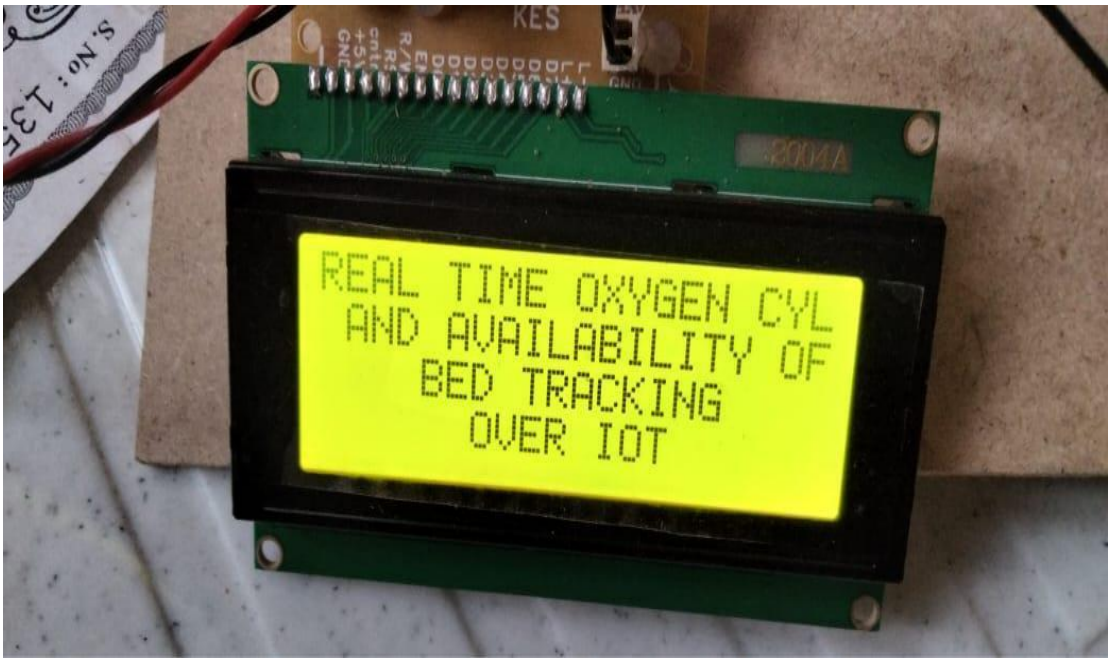


Fig : REAL TIME OXYGEN CYLINDER AND AVAILABILITY OF BEDS TRACKING OVER IOT

Fig 5.5.2 From the above picture it is clear that the availability of beds and oxygen cylinder Tablets are shown total availability as beds are 1000 , oxygen cylinders are 2000 And tablets are 5000.

Fig 5.5.2 From the above picture it is clear that the availability of beds and oxygen cylinder Tablets are shown total availability has been decreased in oxygen cylinders as 2 where taken from 2000 it displays remaining 1998 oxygen cylinders.

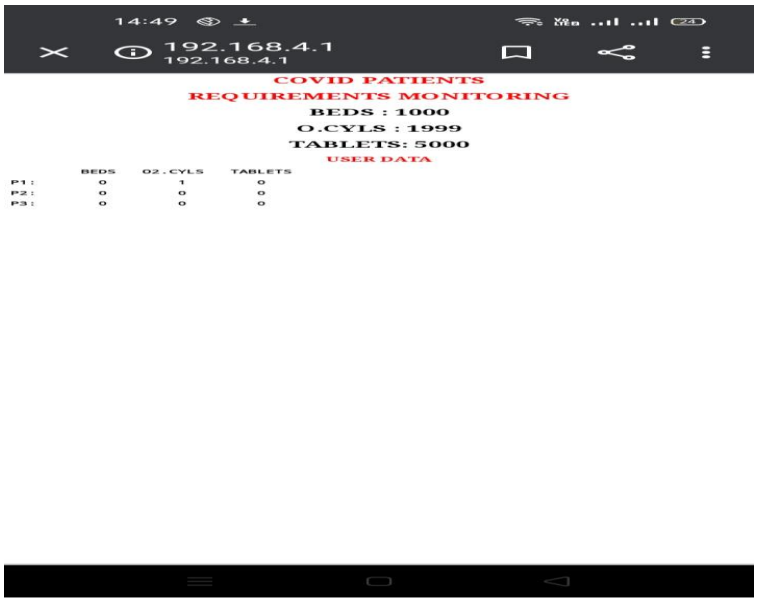


Figure displays the availability after the oxygen cylinders were taken it shows the remaining availability as there is a decrease in oxygen cylinders

Chapter 6

6.1 Conclusion:

Computers and smartphones aren't the only devices connecting to the internet. Everyday objects such as light bulbs, TVs, major appliances, and even doorbells are increasingly featuring internet connectivity. The Internet of Things (IoT) comprises all these devices and objects, all communicating with each other and with data centers over the internet.

Investing in IoT is tricky because so many companies are involved in its various aspects, including businesses that make or provide.

6.2 Future Scope:

Technology contributing to the future of IoT in healthcare is the introduction of 5G networks which provide 100 times faster speeds for connectivity than traditional 4G networks. IoT devices rely on connectivity to communicate and transfer data between patient and care provider. Faster cellular data transfer provides IoT flexibility in terms of the volumes of data it can exchange and at a much faster rate. With these improvements, new healthcare IoT uses include devices that assist patients with their medication adherence at home; sleep monitoring devices that can track heart rate, oxygen levels and movements for high-risk patients; remote temperature monitoring tools; and continuous glucose monitoring sensors that connect to mobile devices and alert patients and clinicians to changing blood sugar levels.

This new pandemic experience combined with the progress and recent advancements will increase the adoption of IoT and encourage those who might have otherwise ignored the technology in the past to get on board.

Bibilography, References

6.3 Bibilography:

1. WWW.MITEL.DATABOOK.COM
2. WWW.ATMEL.DATABOOK.COM
3. WWW.FRANKLIN.COM
4. WWW.KEIL.COM en.wikipedia.org/wiki/ZigBee

   www.zigbee.orG

   www.nxp.com/documents/user_manual/UM1

   http://www.futurlec.com/GPS.shtml013

   HTTP://EN.WIKIPEDIA.ORG/WIKI/GLOBAL_POSITIONING_SYSTEM9.PDF

   http://electronics.howstuffworks.com/gadgets/travel/gps.htm

   http://en.wikipedia.org/wiki/GSM http://burnsidetelecom.com/whitepapers/gsm.pdf

   http://www.itu.int/osg/spu/ni/3G/casestudies/GSM-FINAL.pdf

6.4 References:

1. ARM-systemonchip-architecture by Steve furber.
2. ARM-user manual UM10114.
3. ARM System Developers Guide by Andrew N.SLOSS
4. "Power Electronics" by M D Singh and K B Khanchandan
5. "Linear Integrated Circuits" by D Roy Choudary & Shail Jain
6. "Electrical Machines" by S K Bhattacharya
7. "Electrical Machines II" by B L Thereja
8. www.8051freeprojectsinfo.com

6.5 Appendix:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(14, 15, 16, 17, 18,
19); int SelectBook( int x); int
St[3][3]={{0,0,0},
          {0,0,0},
          {0,0,0}};

const int S1 = 2;
const int S2 = 3;
const int S3 = 4;


const int B2 = 5;
const int B3 = 6;
const int B4 = 7;

const int I = 8;
const int S = 9;
const int R = 10;

int beds=1000;
int ocyl=2000;
```

```
int tabs=5000;
int z;
int y;



void setup()
{
Serial.begin(9600); // connect serial
lcd.begin(20, 4);
pinMode(S1, INPUT);
pinMode(S2, INPUT);
pinMode(S3, INPUT);


pinMode(B2, INPUT);
pinMode(B3, INPUT);
pinMode(B4, INPUT);

pinMode(I, INPUT);
pinMode(S, INPUT);
pinMode(R, INPUT);

lcd.print("REAL TIME OXYGEN CYLS");
lcd.setCursor(0, 1);
```

```
lcd.print(" AND AVAILABILITY OF
"); lcd.setCursor(0, 2);
lcd.print("    BED TRACKING     ");
lcd.setCursor(0, 3);
lcd.print("     OVER IOT      ");
delay(5000);

lcd.clear();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(" BEDS O.CYL TABS");
      lcd.setCursor(0, 1);
      lcd.print("P1:");
      lcd.setCursor(4, 1);
      lcd.print(St[0][0]);
      lcd.setCursor(8, 1);
      lcd.print(St[0][1]);
      lcd.setCursor(13, 1);
      lcd.print(St[0][2]);
      lcd.setCursor(0, 2);
      lcd.print("P2:");
      lcd.setCursor(4, 2);
      lcd.print(St[1][0]);
      lcd.setCursor(8, 2);
      lcd.print(St[1][1]);
```

```
        lcd.setCursor(13, 2);
        lcd.print(St[1][2]);
        lcd.setCursor(0, 3);
        lcd.print("P3:");
        lcd.setCursor(4, 3);
        lcd.print(St[2][0]);
        lcd.setCursor(8, 3);
        lcd.print(St[2][1]);
        lcd.setCursor(13, 3);
        lcd.print(St[2][2]);
        delay(5000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("AVAILABLE BEDS/OC/TA");
lcd.setCursor(0, 1);
lcd.print(" BEDS: ");
lcd.print(GOLD);
lcd.setCursor(0, 2);
lcd.print(" O.CYLS: ");
lcd.print(SILVER);
lcd.setCursor(0, 3);
lcd.print("TABLETS: ");
lcd.print(BRONZE);
//lcd.clear();
}
```

```
void loop()
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("AVAILABLE BEDS/OC/TABS ");
lcd.setCursor(0, 1);
lcd.print("   BEDS: ");
lcd.print(beds);
lcd.setCursor(0, 2);
lcd.print(" O.CYLS: ");
lcd.print(ocyl);
lcd.setCursor(0, 3);
lcd.print("TABLETS: ");
lcd.print(tabs);

while((digitalRead(S1)==HIGH)&&(digitalRead(S2)==HIGH)&
&(digitalRead(S3)==HIGH))
{
  Serial.print("<h1 style=\"color:red;text-
align:center\">COVID PATIENTS </h1><h1
style=\"color:red;text-align:center\">REQUIREMENTS
MONITORING</h1>");
   Serial.print("<h1 style=\"text-align:center;\">BEDS   :
");Serial.print(beds);Serial.print("<h1>");
   Serial.print("<h1 style=\"text-align:center;\">O.CYLS :
```

```
");Serial.print(ocyl);Serial.print("<h1>");
   Serial.print("<h1 style=\"text-align:center\">TABLETS:
");Serial.print(tabs);Serial.print("<h1>");
    Serial.print("<h2 style=\"color:red;text-align:center;\">
USER DATA</h2>");
   Serial.print("<h2>       BEDS   O2.CYLS
TABLETS</h2>");

    Serial.print("<h2>P1:      ");
    Serial.print(St[0][0]);
    Serial.print("       ");
    Serial.print(St[0][1]);
    Serial.print("       ");
    Serial.print(St[0][2]);
    Serial.print("       ");
    Serial.print("<h2>P2:      ");
    Serial.print(St[1][0]);
    Serial.print("       ");
    Serial.print(St[1][1]);
    Serial.print("       ");
    Serial.print(St[1][2]);
    Serial.print("<h2>P3:      ");
    Serial.print(St[2][0]);
    Serial.print("       ");
    Serial.print(St[2][1]);
```

```
    Serial.print("       ");
    Serial.print(St[2][2]);
}

if(digitalRead(S1)==LOW)
{
  z=0;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" P1 SELECTED    ");
  lcd.setCursor(0, 1);
  lcd.print("SELECT ANY REQUIRED");
  Select(0);
}

if(digitalRead(S2)==LOW)
{
  z=1;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" P2 SELECTED    ");
  lcd.setCursor(0, 1);
  lcd.print("SELECT ANY REQUIRED");
  Select(1);
}
```

```
if(digitalRead(S3)==LOW)
{
  z=2;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" P3 SELECTED ");
  lcd.setCursor(0, 1);
  lcd.print("SELECT ANY REQUIRED");
  Select(2);
}
}
int Select( int x)
{
  if(digitalRead(B2)==0)
  {
    y=0;
  lcd.setCursor(0, 2);
  lcd.print(" BED SELECTED ");
    if(digitalRead(I)==LOW)
    {
        if(beds==0)
        {
        lcd.setCursor(0, 3);
        lcd.print("!!!NOT AVAILABLE!!! ");
        }
```

```
      if(beds>0)
      {
      beds=beds-1;
      lcd.setCursor(0, 3);
      lcd.print("   BED ALLOTTED   ");
      St[x][y]=St[x][y]+1;
      }
  }
      if(digitalRead(R)==LOW)
      {
      beds=beds+1;
      lcd.setCursor(0, 3);
      if( St[x][y]>10)
      {
       St[x][y]=St[x][y]-1;
     lcd.print(" BED CANCELLED    ");
      }
      else
      lcd.print("!!! NO STOCK !!!");
      }
}
if(digitalRead(B3)==0)
{
  y=1;
  lcd.setCursor(0, 2);
```

```
lcd.print("O.CYL SELECTED ");
if(digitalRead(I)==LOW)
{
  if(ocyl==0)
  {
  lcd.setCursor(0, 3);
  lcd.print("!!!NOT AVAILABLE!!!");
  }
  if(ocyl>0)
  {
  ocyl=ocyl-1;
  lcd.setCursor(0, 3);
  lcd.print("O.CYL ALLOTED");
  St[x][y]=St[x][y]+1;
  }
}
  if(digitalRead(R)==LOW)
  {
  ocyl=ocyl+1;
  lcd.setCursor(0, 3);

  if( St[x][y]>1)
  {
   St[x][y]=St[x][y]-1;
   lcd.print("   O.CYL CANCELLED ");
```

```
      }
      else
      lcd.print("!!! NO STOCK !!!");
    }
}
if(digitalRead(B4)==0)
{
  y=2;
  lcd.setCursor(0, 2);
  lcd.print("TABLETS SELECTED");
    if(digitalRead(I)==LOW)
    {
      if(tabs==0)
      {
      lcd.setCursor(0, 3);
      lcd.print("!!!NOT AVAILABLE!!!");
      }
      if(tabs>0)
      {
      tabs=tabs-10;
      lcd.setCursor(0, 3);
      lcd.print("TABLETS ALLOTED ");
      St[x][y]=St[x][y]+10;
      }
    }
```

```
  if(digitalRead(R)==LOW)
  {
  tabss=tabs+10;
  lcd.setCursor(0, 3);
  if( St[x][y]>10)
  {
   St[x][y]=St[x][y]-10;
  lcd.print("TABLETS
  CANCELLED"); }
  else
  lcd.print("!!! NO STOCK !!!");

  }
}
delay(2000);
 lcd.clear();
 lcd.setCursor(0, 0);
  lcd.print("   BEDS O.CYL TABS");
  lcd.setCursor(0, 1);
  lcd.print("P1:");
  lcd.setCursor(4, 1);
  lcd.print(St[0][0]);

  lcd.setCursor(8, 1);
  lcd.print(St[0][1]);
```

```
    lcd.setCursor(13, 1);
    lcd.print(St[0][2]);

    lcd.setCursor(0, 2);
    lcd.print("P2:");
    lcd.setCursor(4, 2);
    lcd.print(St[1][0]);
   lcd.setCursor(8, 2);
    lcd.print(St[1][1]);

    lcd.setCursor(13, 2);
    lcd.print(St[1][2]);
    lcd.setCursor(0, 3);
    lcd.print("P3:");
    lcd.setCursor(4, 3);
    lcd.print(St[2][0]);
    lcd.setCursor(8, 3);
    lcd.print(St[2][1]);
    lcd.setCursor(13, 3);
    lcd.print(St[2][2]);
    delay(3000);
    lcd.clear();
    return 0;
}
```