Department of Information Technology
BABU BANARASI DAS
Institute of Technology & Management, Lucknow
Recognized by AICTE, Govt.of India & Affiliated to Dr. A.P.J Abdul Kalam Technical University,Lucknow,AKTU College Code=054

BBDITM

LAB MANUAL

OF

WEB DESIGNING WORKSHOP

BCS353

B.TECH, 2nd Year, Semester -III

Dr. A.P.J. Abdul Kalam Tech. University
Uttar Pradesh

2026-27

INFORMATION TECHNOLOGY

Faculty Name:

Mr.Manish Singh
(Assistant Professor)

Approved by:

(Department Head/Coordinator)

MANUAL CONTENTS

This manual is intended for the 2nd year students of Information Technology in the subject of Web Designing Workshop BCS353. This manual typically contains practical sessions related to web designing covering various aspects related to the subject of enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions.

Vision of the Institute

To become a leading institute of providing professionally competent and socially responsive technocrats with high moral values.

Mission of the Institute

M1: To create an eco-system for the dissemination of technical knowledge, to achieve academic excellence.

M2: To develop technocrats with creative skills and leadership qualities, to solve local and global challenges.

M3: To impart human values and ethics in students, to make them socially and Eco-friendly responsible.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION OF DEPARTMENT

To transform the students to be skilled IT professionals, Innovative leader and environmentally receptive citizens.

MISSION OF DEPARTMENT

M1. To develop and deliver quality academic program in emerging & innovative field of engineering to empower the students to meet industry standards

M2. To provide students with the, ethical and professional tools to become productive & become continuous Learner

M3. To inculcate professional behavior, positive attitude, and communication skills

Program Educational Objectives (PEOs)

PEO I: To develop the ability among students to understand the concept of core Information Technology subjects that will facilitate understanding of new technology.

PEO II: To embed a strong foundation in the engineering fundamentals to solve, analyze and design real time engineering products

PEO III: To give exposures to emerging edge technologies, adequate training and opportunities to work as team on multidisciplinary projects with effective communication skills and leadership qualities

PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Use and apply current technical concepts and practices in the core Information Technologies of human Machine Learning, Information Management, Programming, Cloud Computing.

PSO2: Design, develop and test software systems for world-wide network of computers to provide solutions to real world problems

PROGRAM OUTCOMES

Engineering Graduates will be able to:

1. **Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems..

2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

3. **Design/Development of Solutions**: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5).

4. **Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

5. **Engineering Tool Usag**e: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

6. **The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7)

7. **Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

8. **Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams..

9. **Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

10. **Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments. .

11. **Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

.

Course Evaluation Scheme

Sr No	Subject Code	Subject Name	Periods			Evaluation Scheme				Total	Credit
			L	T	P	Sessional Assessment			PE		
						CT	TA	PS			
1	BCS353	Web Designing Workshop	0	0	2	-	50	50	50	100	1

Course Objectives:

The teacher will explain:

1	Understand Web Development Fundamentals: Gain a solid foundation in HTML, CSS, and JavaScript, including their roles in building and styling web pages.
2	Develop Responsive Web Designs: Learn to create responsive and user-friendly web layouts that function well across various devices and screen sizes.
3	Utilize Development Tools: Familiarize with essential development tools and environments, including text editors, debugging tools, and web browsers for testing and optimization.
4	Build and Deploy a Web Project: Complete a hands-on project that encompasses the entire web development process, from initial design to deployment on a live server.
5	Enhance Problem-Solving Skills: Develop critical thinking and problem-solving abilities by tackling real-world coding challenges and debugging issues that arise during the web development process.

Pre- requisite:

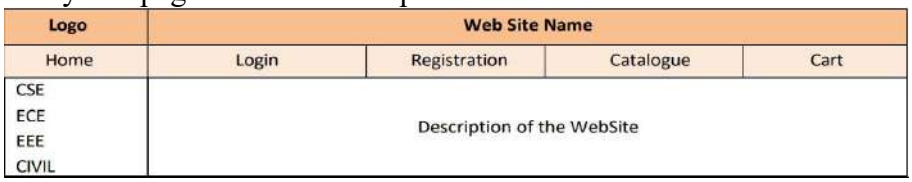
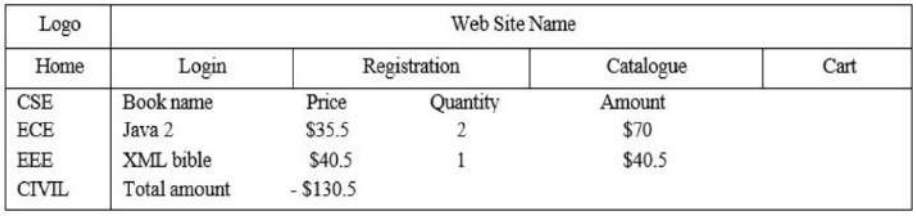
1. Basic Computer Skills: Familiarity with using a computer, including file management, internet browsing, and basic software installation.
2. Understanding of Internet Concepts: A general knowledge of how the internet works, including web browsers, websites, and basic networking concepts.
3. Basic Text Editing Skills: Experience with using text editors or word processors, as students will need to write and edit code.
4. Mathematical Fundamentals: A basic understanding of mathematics, particularly logic and problem-solving skills, which can aid in programming and coding tasks.

Course Outcome of Practical

Course Outcome (CO) At the end of course, the student will be able to:		**Bloom's Level (L)**
CO 1	Understand and apply HTML to create structured web pages with elements, attributes, forms, and media	L3
CO 2	Analyze and implement CSS styles to enhance the visual presentation of web pages, including colors, margins, and padding. Design and implement responsive web layouts using CSS and HTML.	L4
CO 3	Evaluate and utilize JavaScript to manipulate web page content and validate forms	L5
CO 4	Demonstrate the use of JavaScript functions, objects, and events in web development	L3
CO 5	Create interactive web applications by integrating HTML, CSS, Bootstrap, and JavaScript.	L6

CO-PO-PSO Mapping

COURSE OUTCOME	COs	PROGRAM OUTCOMES(POs) & PROGRAM SPECIFIC OUTCOME(PSO)											
		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
	CO1	3	2	3	2	3	–	–	2	2	2	3	2	2
	CO2	3	2	3	2	3	–	–	2	2	2	3	2	2
	CO3	3	3	3	2	3	–	–	2	2	2	3	2	2
	CO4	3	2	3	3	3	–	–	2	2	2	3	2	2
	CO5	3	3	3	3	3	–	–	3	3	3	3	2	2
CO (Average)		3	2	3	2	3	–	–	–	2	2	2	2	2

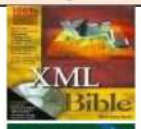
S. No	List of Programs as per AKTU
1	Design the following static webpages required for an online bookstore website. HOMEPAGE: The static home page must contain three frames. Top frame: Logo and the college name and links to Homepage, Login page, Registration page, Catalogue page and Cart page (the description of these pages will be given below). For example: When you click the link "CSE" the catalogue for CSE Books should be displayed in the Right frame. Right frame: The pages to the links in the left frame must be loaded here. Initially this page contains description of the web site. 
2	LOGINPAGE: This page looks like below: 
3	CATOLOGUE PAGE: The catalogue page should contain the details of all the books available in the website in a table. The details should contain the following: 1. Snap shot of Cover Page. 2. Author Name. 3. Publisher. 4. Price. 5. Add to cart button. 
4	CARTPAGE: The cart page contains the details about the books which are added to the cart. The cart page should look like this: 
5	REGISTRATION PAGE : Create a" registration form" with the following fields 1)Name (Text field) 2)Password (password field) 3) E-mailid(text field) 4) Phone Number(text field)

	5) Sex(radio button) 6) Date of birth(3 select boxes) 7) Languages known(checkboxes–English, Telugu, Hindi, Tamil) 8) Address(text area)
6	Js VALIDATION: Write JavaScript to validate the following fields of the above registration page. 1. Name (Name should contains alphabets and the length should not be less than 6 characters). 2. Password (Password should not be less than 6 characters length).
7	Js VALIDATION: 3. E-mailid (should not contain any invalid and must follow the standard pattern(name@domain.com) 4. Phone Number(Phone number should contain 10 digits only).
8	CSS: Design a web page using CSS(Cascading Style Sheets) which includes the following: 1) Use different font, styles: In the style definition you define how each selector should work(font, color etc.). Then, in the body of your pages, you refer to these selectors to activate the styles. 2) Set a background image for both the page and single elements on the page.
9	Design with help of CSS 1) Control the repetition of the image with the background-repeat property. 2) Define styles for links as A:link A:visited A:active A:hover
10	Consider a small topic of your choice on which you can develop static Webpages and try to implement all topics of html, CSS and Js within the topic. Choose any one topic. 1. Your Own Portfolio 2. To-Do List 3. Survey Form 4. A Tribute Page 5. A Questionnaire

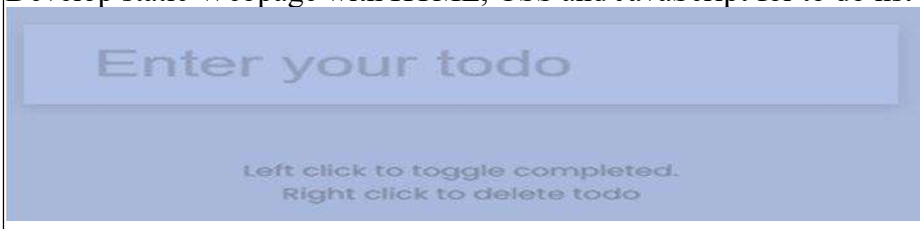
S. No	List of Programs with enhancement of faculty
1	Design the following static webpages required for an online bookstore website. HOMEPAGE: The static home page must contain three frames. Top frame: Logo and the college name and links to Homepage, Login page, Registration page, Catalogue page and Cart page (the description of these pages will be given below). For example: When you click the link "CSE" the catalogue for CSE Books should be displayed in the Right frame. Right frame: The pages to the links in the left frame must be loaded here. Initially this page contains description of the web site.
2	LOGINPAGE: This page looks like below:
3	CATOLOGUE PAGE: The catalogue page should contain the details of all the books available in the website in a table. The details should contain the following: 1. Snap shot of Cover Page. 2. Author Name. 3. Publisher. 4. Price. 5. Add to cart button.
4	CARTPAGE: The cart page contains the details about the books which are added to the cart. The cart page should look like this:
5	REGISTRATION PAGE : Create a" registration form" with the following fields 1)Name (Text field) 2)Password (password field) 3) E-mailid(text field) 4) Phone Number(text field)

Logo	Web Site Name			
Home	Login	Registration	Catalogue	Cart
CSE ECE EEE CIVIL	Description of the WebSite			

Logo	WebSite Name			
Home	Login	Registration	Catalogue	Cart
CSE ECE EEE CIVIL	Login Page User Name: Passwords: Submit Reset			

Logo	WebSite Name			
Home	Login	Registration	Catalogue	Cart
CSE ECE EEE CIVIL	 	Book:XMLBible Author : Winston Publication:Wiely Book :AI Author:S.Russel Publication:Princetonhall	$40.5 $63	Add to cart Add to cart

Logo	Web Site Name			
Home	Login	Registration	Catalogue	Cart
CSE	Book name	Price	Quantity	Amount
ECE	Java 2	$35.5	2	$70
EEE	XML bible	$40.5	1	$40.5
CIVIL	Total amount	- $130.5		

	5) Sex(radio button) 6) Date of birth(3 select boxes) 7) Languages known(checkboxes–English, Telugu, Hindi, Tamil) 8) Address(text area)
6	Js VALIDATION: Write JavaScript to validate the following fields of the above registration page. 1. Name (Name should contains alphabets and the length should not be less than 6 characters). 2. Password (Password should not be less than 6 characters length).
7	Js VALIDATION: 3. E-mailid (should not contain any invalid and must follow the standard pattern(name@domain.com) 4. Phone Number(Phone number should contain 10 digits only).
8	CSS: Design a web page using CSS(Cascading Style Sheets) which includes the following: 1) Use different font, styles: In the style definition you define how each selector should work(font, color etc.). Then, in the body of your pages, you refer to these selectors to activate the styles. 2) Set a background image for both the page and single elements on the page.
9	Design with help of CSS 1) Control the repetition of the image with the background-repeat property. 2) Define styles for links as A:link A:visited A:active A:hover
10	Develop static Webpage with HTML, CSS and JavaScript for to do list 
	List of Value Added Practical
11	Develop static Webpage with HTML, CSS and JavaScript for progress steps. 
12	With the help of HTML, CSS and JavaScript create any game.

INDEX

S. No	Program Name	CO	BTL	DATE	SIGN
1	Design the following static webpages required for an online bookstore website. HOMEPAGE: The static home page must contain three frames. Top frame: Logo and the college name and links to Homepage, Login page, Registration page, Catalogue page and Cart page (the description of these pages will be given below). For example: When you click the link "CSE" the catalogue for CSE Books should be displayed in the Right frame. Right frame: The pages to the links in the left frame must be loaded here. Initially this page contains description of the web site. Logo | Web Site Name Home | Login | Registration | Catalogue | Cart CSE ECE EEE CIVIL Description of the WebSite	CO1	L3		
2	LOGINPAGE: This page looks like below: Logo | WebSite Name Home | Login | Registration | Catalogue | Cart CSE ECE EEE CIVIL Login Page User Name: Passwords: Submit Reset	CO1	L3		
3	CATOLOGUE PAGE: The catalogue page should contain the details of all the books available in the website in a table. The details should contain the following: 1. Snap shot of Cover Page. 2. Author Name. 3. Publisher. 4. Price. 5. Add to cart button. Logo | WebSite Name Home | Login | Registration | Catalogue | Cart CSE ECE EEE CIVIL  Book:XMLBible Author : Winston Publication:Wiely $40.5 Add to cart  Book :AI Author:S.Russel Publication:Princetonhall $63 Add to cart	CO2	L4		
4	CARTPAGE: The cart page contains the details about the books which are added to the cart. The cart page should look like this: Logo | Web Site Name Home | Login | Registration | Catalogue | Cart CSE ECE EEE CIVIL Book name | Price | Quantity | Amount Java 2 | $35.5 | 2 | $70 XML bible | $40.5 | 1 | $40.5 Total amount | - $130.5	CO2	L4		

5	REGISTRATION PAGE : Create a“ registration form“ with the following fields 1)Name (Text field) 2)Password (password field) 3) E-mailid(text field) 4) Phone Number(text field) 5) Sex(radio button) 6) Date of birth(3 select boxes) 7) Languages known(checkboxes–English, Telugu, Hindi, Tamil) 8) Address(text area)	CO3	L5		
6	Js VALIDATION: Write JavaScript to validate the following fields of the above registration page. 1. Name (Name should contains alphabets and the length should not be less than 6 characters). 2. Password (Password should not be less than 6 characters length).	CO4	L3		
7	Js VALIDATION: 3. E-mailid (should not contain any invalid and must follow the standard pattern(name@domain.com) 4. Phone Number(Phone number should contain 10 digits only).	CO4	L3		
8	CSS: Design a web page using CSS(Cascading Style Sheets) which includes the following: 1) Use different font, styles: In the style definition you define how each selector should work(font, color etc.). Then, in the body of your pages, you refer to these selectors to activate the styles. 2) Set a background image for both the page and single elements on the page.	CO3	L5		
9	Design with help of CSS 1) Control the repetition of the image with the background-repeat property. 2) Define styles for links as A:link A:visited A:active A:hover	CO3	L5		
10	Develop static Webpage with HTML, CSS and JavaScript for to do list 	CO5	L6		
11	Develop static Webpage with HTML, CSS and JavaScript for progress steps. 	CO5	L6		
12	With the help of HTML, CSS and JavaScript create any game.	CO5	L6		

Softwares & Hardwares Required

1. Computer/Laptop- Multi-core processor (e.g., Intel i5),Min 8 GB RAM, Min 256 GB SSD storage
2. Display-High-resolution monitor (1080p or higher)
3. Input Devices- keyboard, mouse and Internet Connection-Reliable high-speed internet
4. Development Environment- Operating System: Windows, macOS, or Linux
5. Development Tools: Text editor (e.g., Visual Studio Code, Notepad++ etc), version control (e.g., Git), web browsers (e.g., Chrome, Firefox)

Practical No. 1

1. **Experiment Description:** Home Page: The static home page must contain three frames.
   Top frame: Logo and the college name and links to Home page, Login page, Registration page, Catalogue page and Cart page (the description of these pages will be given below).
   Left frame: At least four links for navigation, which will display the catalogue of respective links . For e.g.: When you click the link "IT" the catalogue for IT Books should be displayed in the Right frame.
   Right frame: The pages to the links in the left frame must be loaded here.
   Initially this page contains description of the web site**.**
2. **Objective:** Design the following static webpage required for an online bookstore website
3. **Code and Procedure:**

home.html:

```
<html>
<frameset rows="15%,9%,76%" bodycolor="pink">
      <frameset cols="20%,80%" borderwork="red">
            <frame src="logo.html" scrolling="no">
            <frame src="title.html" >
</frameset>
     <frame src="menu.html" name="f2" scrolling="no">
     <frameset cols="15%,85%" bordercolor="red">
           <frame src=" branches.html" name="f31">
           <frame src="homedes.html" name="f32">
</frameset></frameset></html>
```

logo.html

```
<html>
<body>
<imgsrc=" C:\Documents and Settings\abc\Desktop\Sunset.jpg" >
</body></html>
```

title.html

```
<html>
<body>
<center<i> ONLINE BOOK SHOP </i></font></center>
</body>
</html>
```

branches.html

```
<html>
<table cellspacing=15>
<tr><td><a href="csechecktest.html" target="f32">CSE </a></td></tr>
      <tr><td><a href="ecechecktest.html" target="f32">  ECE  </a></td></tr>
      <tr><td><a href="eeechecktest.html" target="f32">EEE  </a></td></tr>
      <tr><td><a href="civilchecktest.html" target="f32"> Civil </a></td></tr>
</table></body></html>
```

menu.html

```
<html>
<body>
<table width="100%">
<tr>
        <td><a href="homedes.html"  target="f32">Home </a></td>
        <td><a href="login.html"  target="f32">Login  </a></td>
        <td><a href="registration.html"  target="f32">Registration</a></td>
        <td><a href="catalogue.html"  target="f32">Catalogue  </a></td>
        <td><a href="cart.html"  target="f32">Cart   </a></td>
    </tr>
</table>
</body></html>
```

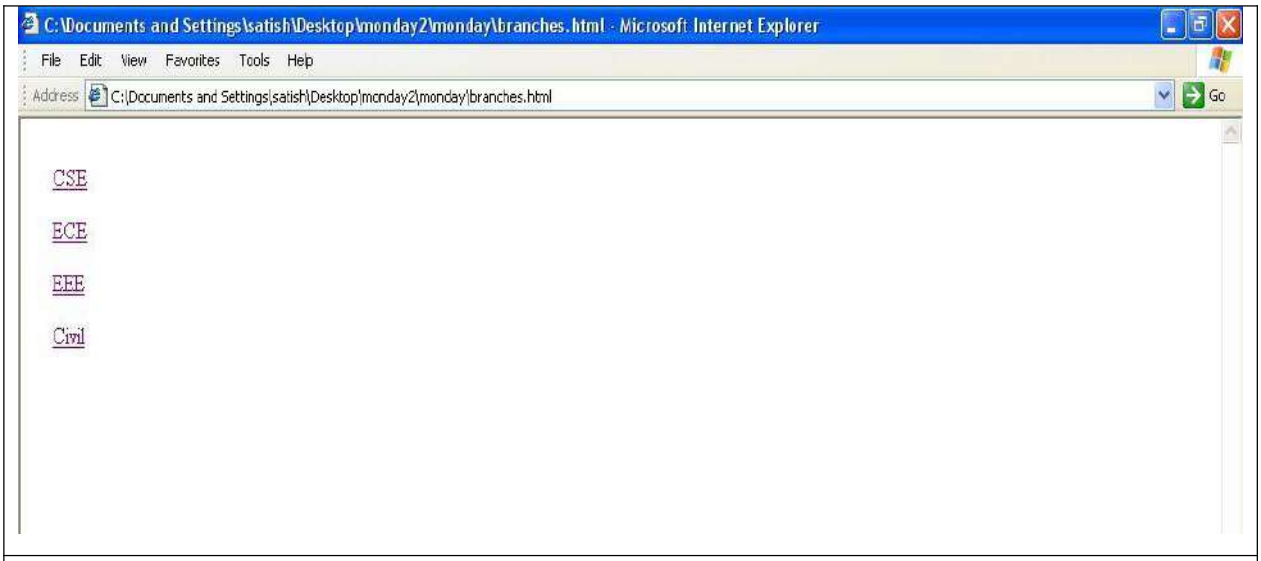
homedes.html

```
<html>
<body>
<center><u> ONLINE BOOK SHOP </u></center>
<p>This website contains various books. </body></html>
```

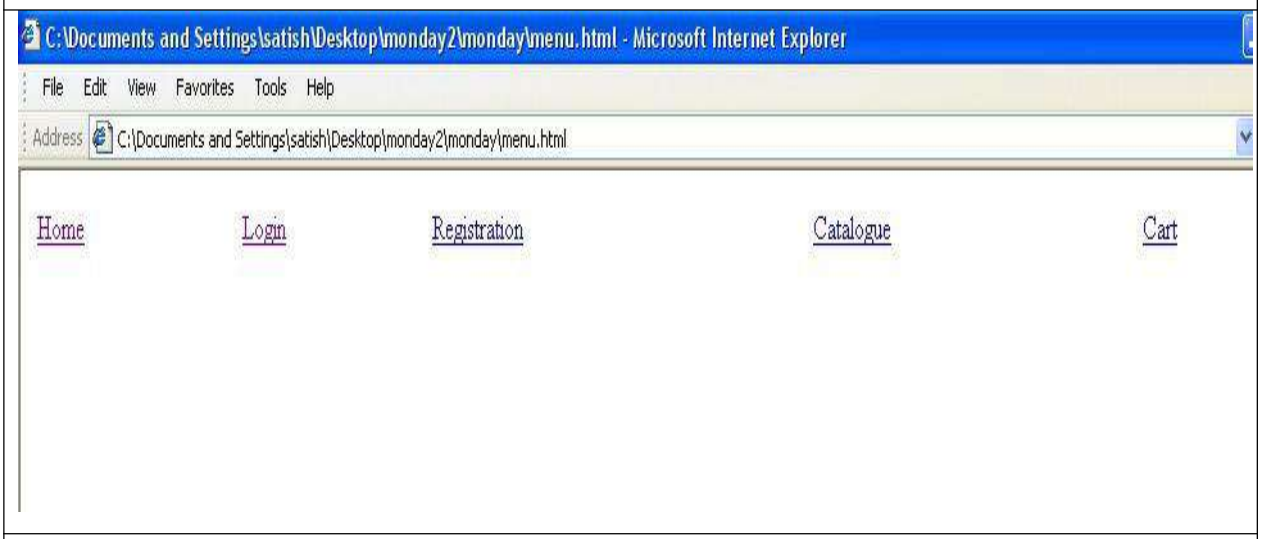
4. **Output:**



1. **Output of Home Page**



2. Output of Left Frame



3. Output of Top Frame

5. **Outcome of the Practical:** The practical outcome is the creation of a static online bookstore homepage featuring a three-frame layout: a top frame for navigation links, a left frame for category links, and a right frame for displaying content based on user selections. This structure enhances user experience by providing easy access to various sections of the website while maintaining a clear and organized interface
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**
   - What is the purpose of using frames in your static home page design, and how are they structured?
   - How did you implement the navigation links in the top frame, and what pages do they link to?

- Can you explain how the left frame faciitates navigation to different catalogues and how the right frame displays the corresponding content?
- What HTML elements and attributes did you use to create the layout of the frames?
- What challenges did you face while designing the static home page, and how did you overcome them?
- How did you ensure that the links in the left frame load the appropriate content in the right frame?
- How did you style the frames and links to enhance the visual appeal of the webpage?
- What considerations did you take into account for user experience when designing the navigation structure?
- How did you test the functionality of the navigation links and ensure they work as intended?
- What are the advantages and disadvantages of using frames in web design, and would you consider using them in future projects?

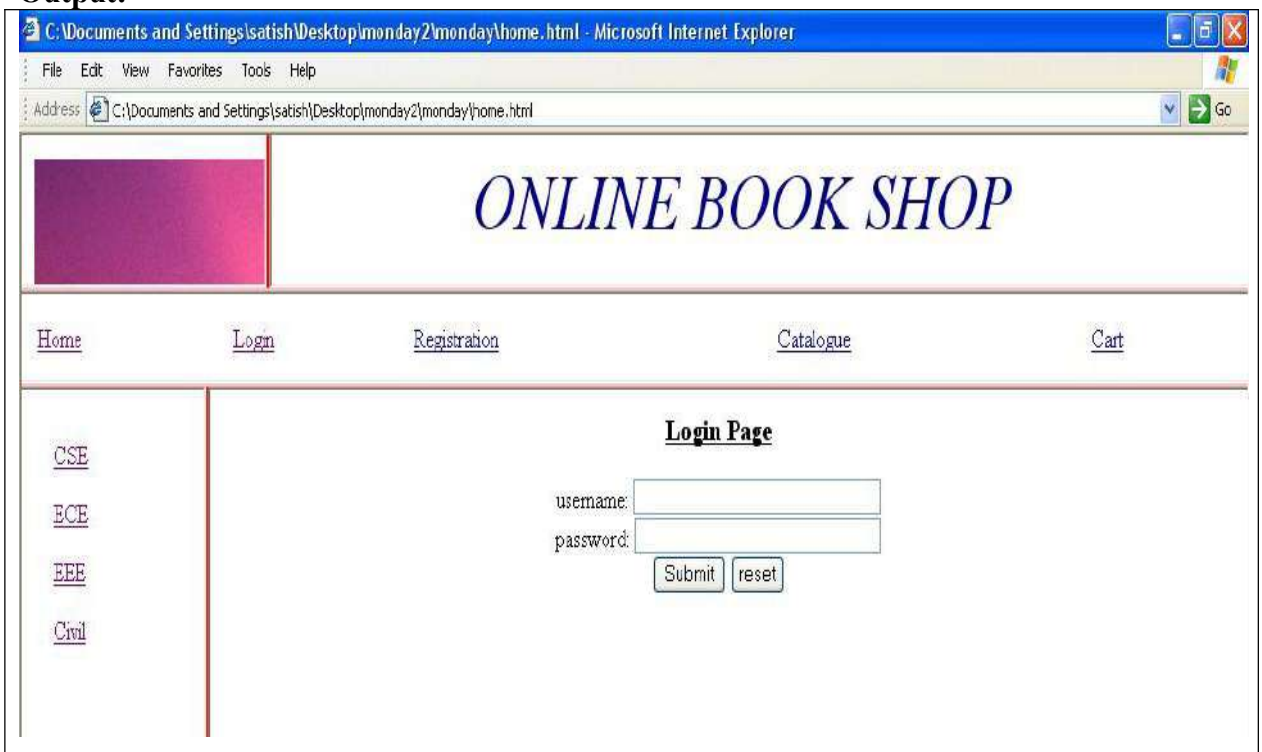
Practical No. 2

1. **Experiment Description:** A login page using HTML involves structuring a simple form that includes fields for user input, such as username and password. You typically use the <form> element to encapsulate the input fields, which are created using <input> elements with appropriate types (e.g., "text" for username and "password" for password). Additionally, you can include a submit button using the <button> or <input type="submit"> element
2. **Objective:** Create a login page
3. **Code and Procedure:**

login.html

```
<html>
<head><title> Login Page </title></head>
<body> <form name="login">
<h3><u> Login Page  </u>
</h3> username: <input type="text" name="uname" >
<br>  password: <input type="password" name="pwd" >
<br> <input type="submit" value="Submit" >
<input type="reset" value="reset">
</center></form></body></html>
```

4. **Output:**



Output of Login Page

5. **Outcome of the Practical:** students learn to use essential HTML tags to create a functional login page, including the <form>, <input>, and <button> elements. They gain hands-on experience in structuring forms, applying input types for better user experience, and incorporating labels for accessibility. This exercise enhances their understanding of HTML form handling and prepares them for more advanced web development concepts.

6. Assessment Criteria:

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. Viva Questions/Review Questions:

- What are the key components of a login page, and how did you structure them using HTML?
- Can you explain the purpose of the <form> element in your login page?
- What types of <input> elements did you use for the username and password fields, and why?
- How did you ensure that the password input is masked when the user types it?
- What attributes did you include in the <input> elements to enhance user experience (e.g., placeholders, required fields)?
- How did you implement the submit button, and what type did you choose for it?
- What validation techniques are available in HTML to ensure that the user inputs are correct before submission?
- How would you structure the layout of the login page using HTML elements?
- What styling techniques did you apply using CSS to make the login page visually appealing?
- How did you test the functionality of the login page to ensure it works as intended?

Practical No. 3

- **Experiment Description:** This practical focuses on designing a comprehensive catalogue page that presents all available books on the website in an organized manner. The page is structured using HTML, which serves as the backbone for the layout. HTML elements such as <table>, <tr>, <td>, and <img> are used to create a visually appealing table that displays essential details for each book, including a snapshot of the cover page, author name, publisher, and price.
  XML is employed to store and manage the book data in a structured format. This allows for easy updates and retrieval of information, ensuring that the catalogue can be dynamically populated with the latest book details without altering the HTML structure. The use of XML facilitates a clear separation between the content (book information) and the presentation (how it is displayed on the webpage).
  JavaScript is integrated to enhance the interactivity of the catalogue page. It enables functionalities such as the "Add to Cart" button for each book, allowing users to select and add items to their shopping cart seamlessly. JavaScript can also be used to validate user inputs, dynamically update the cart display, and manage user interactions, creating a more engaging experience.
  Overall, this practical combines HTML for structure, XML for data management, and JavaScript for interactivity, resulting in a dynamic and user-friendly catalogue page that effectively showcases the available books.
- **Objective:** To design a catalogue page. The catalogue page should contain the details of all the books available in the website in a table.
- **Code and Procedure:**

  catalogue.html

```
<html><head><title>loading xml document</title>
<script language="javascript">
function fun()
        { window.alert("you have selected the book"); }
</script></head>
<body>
<script language="javascript">
  var xmldoc;
 xmldoc = new activexobject("microsoft.xmldom");
xmldoc.load("book.xml");
arr=xmldoc.getelementsbytagname("item");
for(i=0;i<arr.length;i++)
 { document.write("<div align='left'><imgsrc='sunset.jpg' width='50' height='50'> </div>");
document.write("<div align='center'><b> name:</b>
"+xmldoc.getelementsbytagname("name")[i].childnodes[0].nodevalu  e+" <br></div>");
document.write("<div align='center'><b>author:</b>
"+xmldoc.getelementsbytagname("author")[i].childnodes[0].nod evalue+"<br></div>");
document.write("<div align='center'><b>price:</b>
"+xmldoc.getelementsbytagname("price")[i].childnodes[0].nodevalu e+"<br> </div>");
document.write("<div align='center'><b>publisher:</b>
"+xmldoc.getelementsbytagname("publisher")[i].childnodes[0].nodevalue+"<br></div>");
document.write("<div align='center'><b>isbn:</b>
"+xmldoc.getelementsbytagname("isbn")[i].childnodes[0].nodevalue+"<br></div>");
document.write("<div align='right'><input type='button' value='add to cart'
onclick='fun()'</div>"); document.write("<br>"); }
</script></body></html>
```

book.xml

```
<?xml version="1.0"?>
<BOOK>
<ITEM>
      <NAME>   JAVA </NAME>
     <AUTHOR> SCHIELD </AUTHOR>
      <PRICE>  500 </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN>  12345   </ISBN>
</ITEM>
<ITEM>
      <NAME> C++   </NAME>
     <AUTHOR> HERBERT </AUTHOR>
      <PRICE> 499.99  </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN> 12346    </ISBN>
</ITEM>
<ITEM>
      <NAME>DigitalElectronics</NAME>
     <AUTHOR> SCHIELD </AUTHOR>
      <PRICE>  500 </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN>  12345   </ISBN>
</ITEM>
<ITEM>
      <NAME>opticalCommunication</NAME>
     <AUTHOR> HERBERT </AUTHOR>
      <PRICE> 499.99  </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN> 12346    </ISBN>
</ITEM>
<ITEM>
      <NAME>   Electric Theory1 </NAME>
     <AUTHOR> SCHIELD </AUTHOR>
      <PRICE>  500 </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN>  12345   </ISBN>
</ITEM>
<ITEM>
      <NAME> Electric Theory2   </NAME>
     <AUTHOR> HERBERT </AUTHOR>
      <PRICE> 499.99  </PRICE>
     <PUBLISHER> TATA </PUBLISHER>
      <ISBN> 12346    </ISBN>
</ITEM>
</BOOK>
```

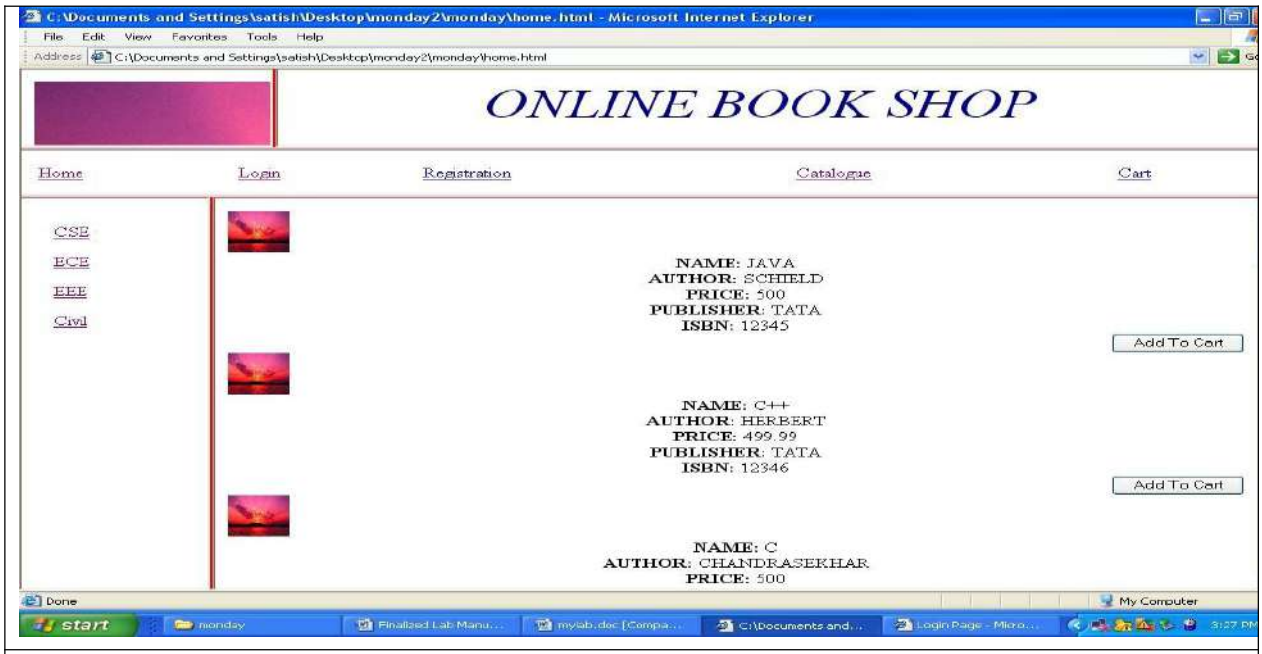
csechecktest.html

```
<html><head><title>loading xml document</title>
<script language="javascript">
function fun()
        { window.alert("you have selected the book"); }
</script></head>
<body>
<script language="javascript">
 var xmldoc;
 xmldoc = new activexobject("microsoft.xmldom");
xmldoc.load("csebook.xml");
arr=xmldoc.getelementsbytagname("item");
for(i=0;i<arr.length;i++)
{ document.write("<div align='left'><imgsrc='sunset.jpg' width='50' height='50'> </div>");
document.write("<div align='center'><b>name:</b>
"+xmldoc.getelementsbytagname("name")[i].childnodes[0].nodevalu e+"<br></div>");
document.write("<div align='center'><b>author:</b>
"+xmldoc.getelementsbytagname("author")[i].childnodes[0].nod evalue+"<br></div>");
document.write("<div align='center'><b>price:</b>"
+xmldoc.getelementsbytagname("price")[i].childnodes[0].nodevalu e+"<br></div>");
document.write("<div align='center'><b>publisher:</b>"
xmldoc.getelementsbytagname("publisher")[i].childnodes[0].nodevalue+"<br></div>");
document.write("<div align='center'><b>isbn:</b>"
xmldoc.getelementsbytagname("isbn")[i].childnodes[0].nodevalue+"<br></div>");
document.write("<div align='right'><input type='button' value='add to cart'
onclick='fun()'</div>"); document.write("<br>"); }
</script></body></html>
```

casebook.xml

```
<?xml version="1.0"?>
<CSEBOOK>
<ITEM>
       <NAME>   JAVA </NAME>
      <AUTHOR> SCHIELD </AUTHOR>
       <PRICE>  500 </PRICE>
      <PUBLISHER> TATA </PUBLISHER>
       <ISBN>  12345   </ISBN>
</ITEM>
<ITEM>
       <NAME> C++   </NAME>
      <AUTHOR> HERBERT </AUTHOR>
       <PRICE> 499.99  </PRICE>
       <PUBLISHER> TATA </PUBLISHER>
       <ISBN> 12346    </ISBN>
</ITEM>
</CSEBOOK>
```

- **Output:**



Output of Catalogue Page: right frame(we will get this output when we click on cse which was in left frame)

- **Outcome of the Practical:** The outcome of this experiment is a fully functional catalogue page that effectively displays a list of available books, showcasing their cover images, author names, publishers, and prices in a well-organized table format. Students gain practical experience in using HTML to create structured layouts, XML to manage and retrieve book data efficiently, and JavaScript to implement interactive features such as the "Add to Cart" button.
- **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. Viva Questions/Review Questions:

- What is the purpose of the catalogue page, and how is it structured using HTML?
- Can you explain the role of the <table>, <tr>, and <td> elements in displaying the book details?
- How did you incorporate images of the book covers into the catalogue, and what HTML element did you use for this?
- What essential details are included for each book in the catalogue, and why are they important for users?
- How does XML facilitate the management of book data, and what advantages does it provide over hardcoding the data in HTML?
- Can you describe how you would update the book information in the XML file without changing the HTML structure?
- What styling techniques did you apply to the catalogue page to enhance its visual appeal?
- How did you ensure that the catalogue page is organized and easy to navigate for users?
- What methods did you use to test the functionality of the catalogue page and ensure that all elements display correctly?
- How would you handle the integration of JavaScript for interactivity in the catalogue page, such as the "Add to Cart" functionality?

Practical No. 4

1. **Experiment Description:** designing a cart page that displays details of books added to a shopping cart, utilizing XML, DTD (Document Type Definition), HTML, and XSL (Extensible Stylesheet Language).
   XML (eXtensible Markup Language) is used to store the cart data in a structured format, allowing for the representation of complex data hierarchies. In this context, XML can include elements such as <book>, <title>, <author>, <price>, and <quantity>, encapsulating all relevant information about each book in the cart. The use of XML provides flexibility, as it can easily accommodate changes in data structure, such as adding new attributes or elements without disrupting the existing data. Additionally, XML is platform-independent, making it suitable for data interchange between different systems.
   DTD (Document Type Definition) is created to define the structure and rules for the XML file. It specifies the allowed elements, their attributes, and the relationships between them, ensuring that the XML document adheres to a consistent format. For example, a DTD for the cart might define that each <book> element must contain a <title>, <author>, <price>, and <quantity>. This validation helps maintain data integrity, ensuring that any XML file conforming to the DTD is well-formed and can be reliably parsed by applications. By enforcing these rules, DTDs help prevent errors and inconsistencies in the data.
   HTML (HyperText Markup Language) is employed to create the layout of the cart page. It structures the content, allowing for the display of book details in a user-friendly format. HTML elements such as <table>, <tr>, and <td> are used to organize the information clearly, presenting each book's details in rows and columns.
   XSL (Extensible Stylesheet Language) is utilized to transform the XML data into a presentable format for the web page. XSLT (XSL Transformations) allows developers to define how the XML content should be displayed, enabling the transformation of XML into HTML. This process involves applying styles and formatting to the XML data, ensuring that the cart page is visually appealing and easy to read. For instance, XSL can be used to format prices, apply styles to book titles, or create conditional formatting based on the quantity of books in the cart. By separating the data (XML) from its presentation (XSL), developers can maintain a clean and organized codebase, making future updates easier.
2. **Objective:** To design a CART PAGE that contains the details about the books which are added to the cart.
3. **Code and Procedure:**

cart.html

```
<html> <body >
<table   cellspacing="50" cellpadding="20">
<tr><td>Book name</td><td>price</td><td>quantity</td><td>amount</td></tr>
<tr><td>java2</td><td>$35.5</td><td>2</td><td>$70</td></tr>
<tr><td>XML bible</td><td>$40.5</td><td>1</td><td>$40.5</td></tr>
<tr><td></td><td></td> <td>Total amount </td> <td>130.5 </td></tr>
</body> </html>
```

Program for creating DTD for the XML file(dtd.xml

```
<?xml version="1.0" ?>
<?xml:stylesheet type="text/xsl" href="Style.xsl"?>
<!DOCTYPE BOOKS[
<!ELEMENT BOOKS (INFORMATION)>
<!ELEMENT INFORMATION (Book)>
<!ELEMENT Book (book_name,Author_name,ISBN_number,publisher,Edition,Price)>
<!ELEMENT book_name (#PCDATA)>
<!ELEMENT Author_name (#PCDATA)>
```

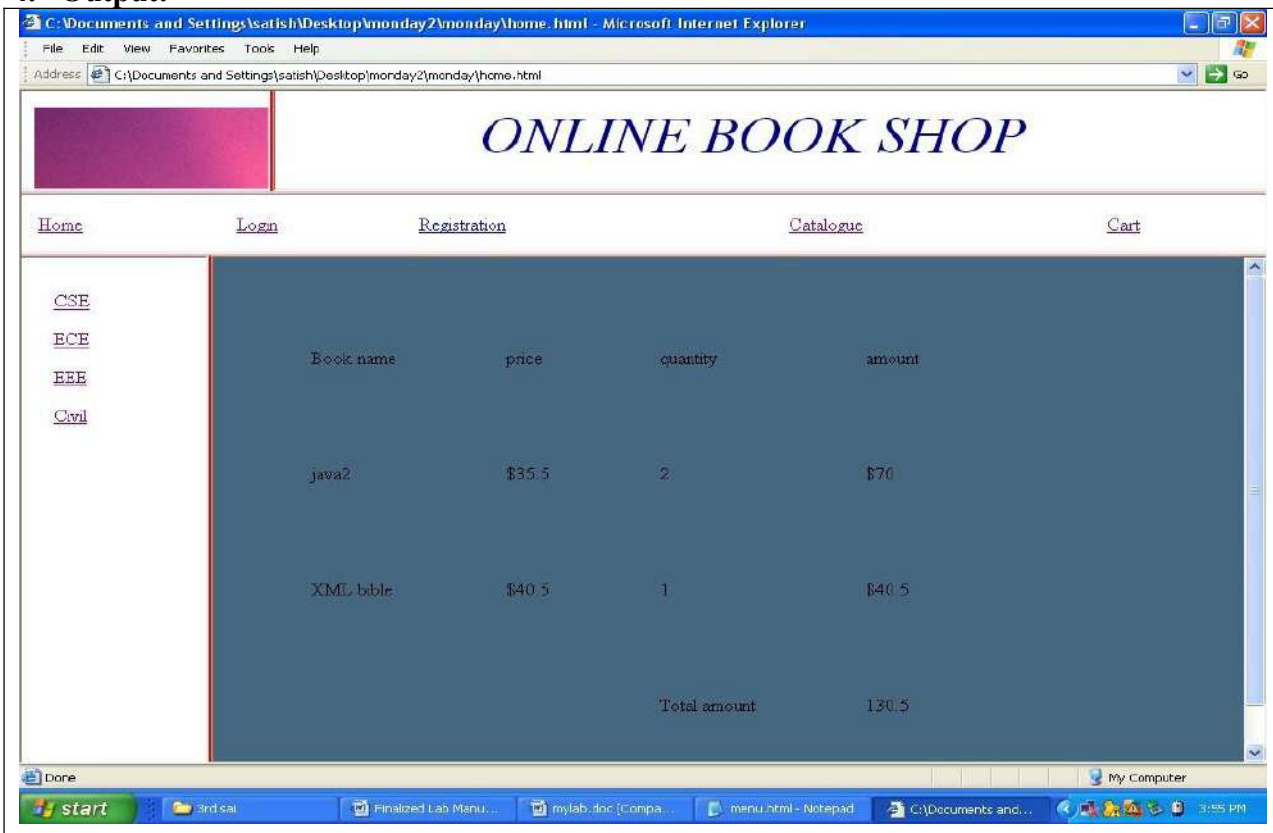
```
<!ELEMENT ISBN_number (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT Edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>]>
<BOOKS>
<INFORMATION>
<Book>
<book_name>XML BIBLE</book_name>
<Author_name>RAJA</Author_name>
<ISBN_number>11530-115</ISBN_number>
<publisher>Pearson</publisher>
<Edition>I-Edition</Edition>
<Price>750</Price>
</Book>
<Book>
<book_name>HTML</book_name>
<Author_name>DIETL</Author_name>
<ISBN_number>11528-115</ISBN_number>
<publisher>wiley</publisher>
<Edition>III-Edition</Edition>
<Price>470</Price>
</Book>
<Book>
<book_name>JAVASCRIPT</book_name>
<Author_name>RANI</Author_name>
<ISBN_number>11525-115</ISBN_number>
<publisher>Techical</publisher><Edition>IV-Edition</Edition>
<Price>270</Price>
</Book>
<Book>
<book_name>WEB TECHNOLOGIES</book_name>
<Author_name>PUNTAMBEKAR</Author_name>
<ISBN_number>11530-115</ISBN_number>
<publisher>Technical</publisher>
<Edition>I-Edition</Edition>
<Price>340</Price>
</Book>
</INFORMATION>
</BOOKS>
```

Style.xsl

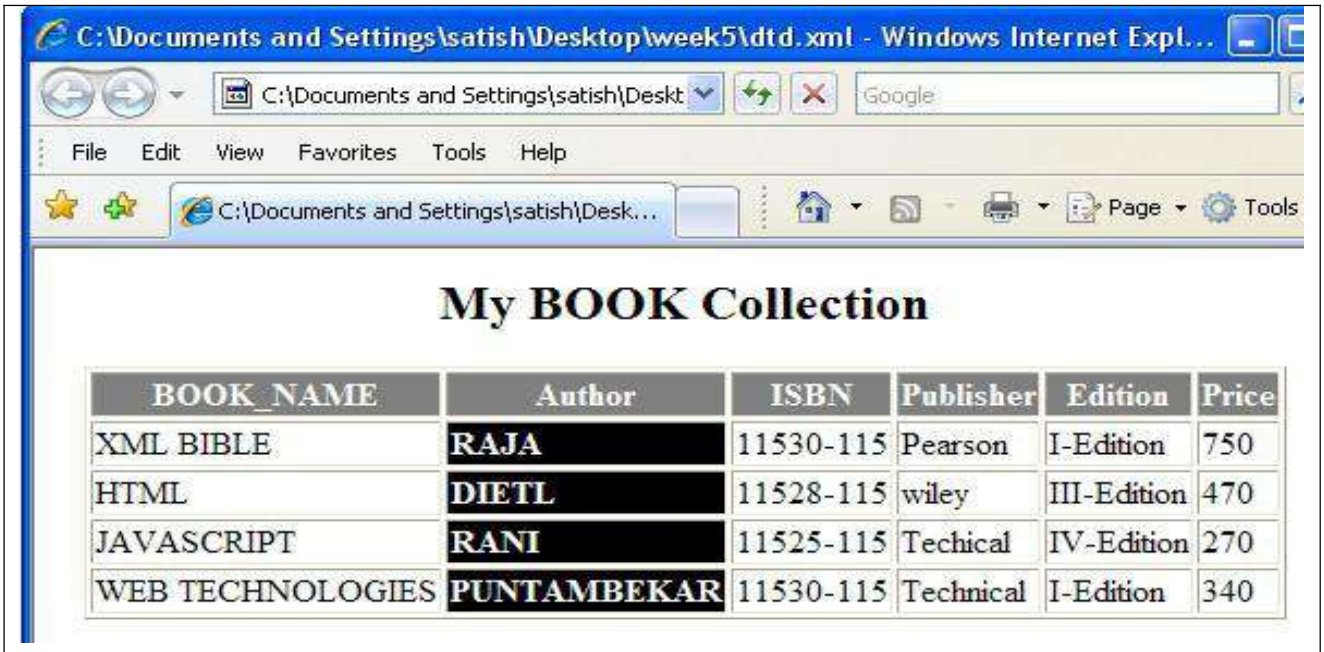
```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2 align="center">My BOOK Collection</h2>
<table border="1" align="center">
<tr bgcolor="GRAY">
<th><font color="white"> BOOK_NAME</font></th>
<th><font color="white"> Author</font></th>
<th><font color="white"> ISBN</font></th>
<th><font color="white"> Publisher</font></th>
```

```
<th><font color="white"> Edition</font></th>
<th><font color="white"> Price</font></th>
</tr>
<xsl:for-each select="BOOKS/INFORMATION/Book">
<tr>
<td><xsl:value-of select="book_name"/></td>
<td bgcolor="black"><b><font color="white">
<xsl:value-of select="Author_name"/></font></b></td>
<td><xsl:value-of select="ISBN_number"/></td>
<td><xsl:value-of select="publisher"/></td>
<td><xsl:value-of select="Edition"/></td>
<td><xsl:value-of select="Price"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

4. Output:



1. **Output of Cart Page**



2. Output of dtd.xml file

5. **Outcome of the Practical:** The outcome of this practical is a comprehensive understanding of how to design and implement a cart page that effectively displays book details using XML, DTD, HTML, and XSL. Students gain hands-on experience in structuring data with XML and defining its rules through DTD, ensuring data integrity and consistency. They learn to create a user-friendly layout with HTML, organizing information in a clear and accessible manner.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. Viva Questions/Review Questions:

- What is the purpose of the cart page, and how does it enhance the user experience in an online bookstore?
- How is XML utilized to store the cart data, and what elements are included in the XML structure?
- Can you explain the role of DTD (Document Type Definition) in your XML file and how it ensures data integrity?
- What specific rules and relationships did you define in the DTD for the cart's XML structure?
- How did you use HTML to create the layout of the cart page, and what elements did you employ to display the book details?
- What is XSL, and how does it facilitate the transformation of XML data into a presentable format for the cart page?
- Can you describe how you applied XSLT (XSL Transformations) to format the XML data for display on the cart page?
- How does the separation of data (XML) and presentation (XSL) benefit the development and maintenance of the cart page?
- What styling techniques did you implement to ensure that the cart page is visually appealing and user-friendly?
- How did you test the functionality of the cart page to ensure that all book details are displayed correctly and that the data is accurately represented?

Practical No. 5

1. **Experiment Description:** The registration page is designed to collect user information for account creation. It includes the following fields:
   - Name (Text Field): A single-line text input where users can enter their full name. This field is required to identify the user.
   - Password (Password Field): A secure input field that masks the characters entered, allowing users to create a password for their account. This field is also required to ensure account security.
   - E-mail ID (Text Field): A text input field for users to enter their email address. This field is required for account verification and communication purposes.
   - Phone Number (Text Field): A text input field where users can provide their phone number. This field is required for additional verification and contact.
   - Sex (Radio Button): A set of radio buttons allowing users to select their gender. Options typically include "Male" and "Female," and one selection is required.
   - Date of Birth (Three Select Boxes): A combination of three dropdown menus for users to select their day, month, and year of birth. This information is required to verify the user's age.
   - Languages Known (Checkboxes): A series of checkboxes that allow users to select multiple languages they are proficient in. Options include English, Telugu, Hindi, and Tamil. Users can check any combination of these languages.
   - Address (Text Area): A larger text area where users can enter their residential address. This field is required for identification and communication purposes.

   HTML is used to create the structure and content of a web page. It consists of elements (tags) that define the content, such as headings, paragraphs, forms, images, and links. HTML defines the structure of input fields, labels, and buttons to collect user information.

   CSS is used to style the HTML elements, making the web page visually appealing. It controls aspects like layout, colors, fonts, spacing, and responsiveness. CSS allows designers to separate content (HTML) from presentation, improving maintainability and flexibility.
2. **Objective:** Create a Registration form having various fields.
3. **Code and Procedure:**

Registration.html

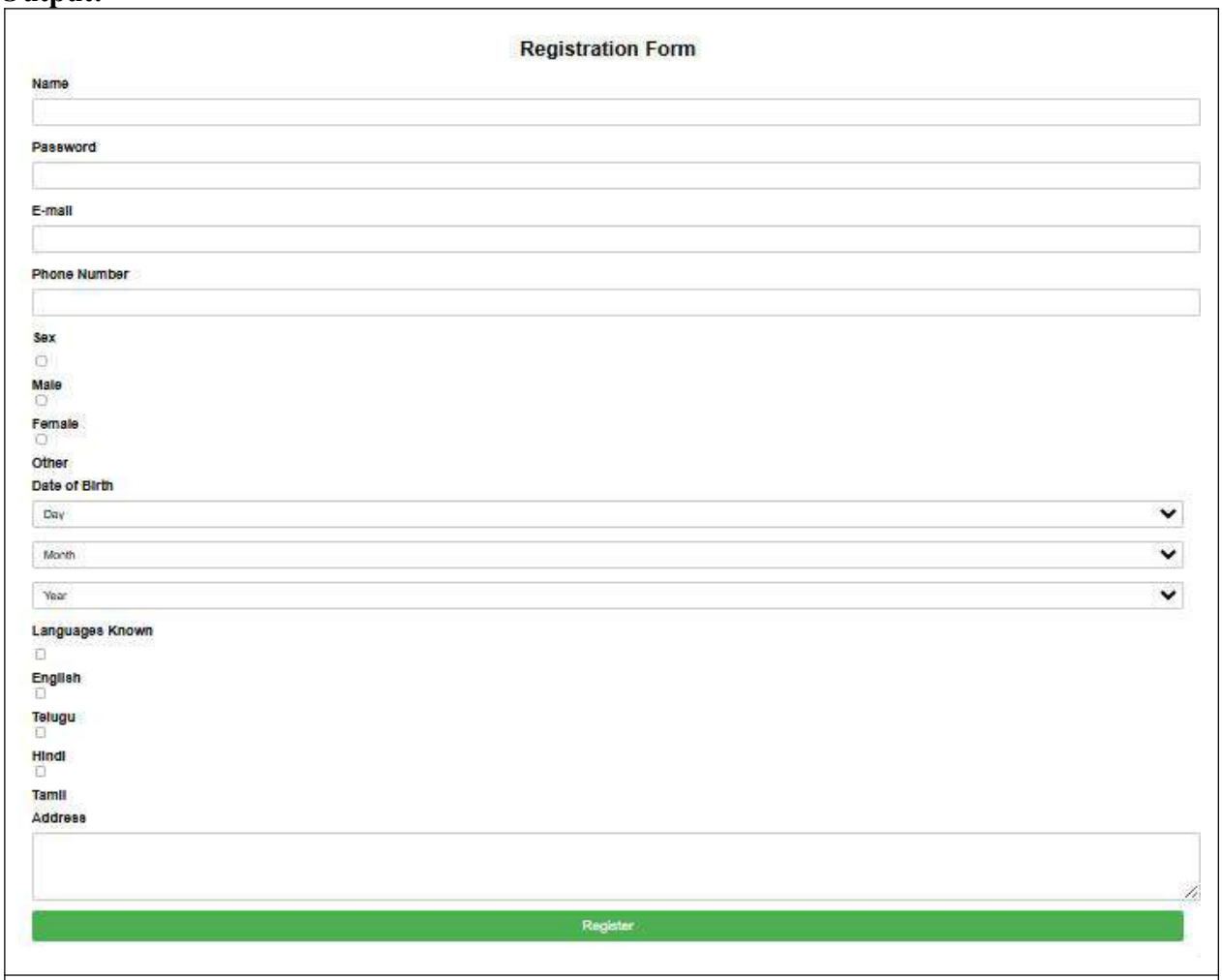
```
<html >
<head> <title>Registration Page</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
 <div class="container">
  <h2>Registration Form</h2>
      <form action="#" method="POST">
        <label for="name">Name</label>
       <input type="text" id="name" name="name" required>
       <label for="password">Password</label>
       <input type="password" id="password" name="password" required>
       <label for="email">E-mail</label>
       <input type="email" id="email" name="email" required>
       <label for="phone">Phone Number</label>
       <input type="text" id="phone" name="phone" required>
       <label>Sex</label>
      <div class="radio-group">
          <input type="radio" id="male" name="sex" value="male">
```

```
            <label for="male">Male</label>
            <input type="radio" id="female" name="sex" value="female">
            <label for="female">Female</label>
            <input type="radio" id="other" name="sex" value="other">
            <label for="other">Other</label>
        </div>
    <label for="dob">Date of Birth</label>
  <div>
            <select id="day" name="day" required>
              <option value="" disabled selected>Day</option>
              <option value="1">1</option>
              <option value="2">2</option>
              <option value="3">3</option>
              <!-- Continue this up to 31 -->
            </select>
            <select id="month" name="month" required>
              <option value="" disabled selected>Month</option>
              <option value="1">January</option>
              <option value="2">February</option>
              <option value="3">March</option>
              <!-- Continue for all months -->
            </select>
            <select id="year" name="year" required>
              <option value="" disabled selected>Year</option>
              <option value="2000">2000</option>
              <option value="1999">1999</option>
              <option value="1999">1999</option>
              <!-- Continue for other years -->
            </select>
        </div>
          <label>Languages Known</label>
   <div class="checkbox-group">
            <input type="checkbox" id="english" name="languages" value="English">
            <label for="english">English</label>
            <input type="checkbox" id="telugu" name="languages" value="Telugu">
            <label for="telugu">Telugu</label>
            <input type="checkbox" id="hindi" name="languages" value="Hindi">
            <label for="hindi">Hindi</label>
            <input type="checkbox" id="tamil" name="languages" value="Tamil">
            <label for="tamil">Tamil</label>            </div>
        <label for="address">Address</label>
        <textarea id="address" name="address" rows="4" required></textarea>
        <button type="submit" class="submit-btn">Register</button>
      </form>
</div>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f9;
}
.container
    { width: 50%;
    margin: 50px auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h2 {
    text-align: center;
}
label {
    margin-top: 10px;
    display: block;
    font-weight: bold;
}
input[type="text"], input[type="password"], input[type="email"], select, textarea
    { width: 100%;
    padding: 8px;
    margin: 8px 0;
    border-radius: 4px;
    border: 1px solid #ccc;
}
.radio-group, .checkbox-group
    { margin: 10px 0;
}
input[type="radio"], input[type="checkbox"]
    { margin-right: 10px;
}
.submit-btn {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;  cursor:
    pointer; border-
    radius: 4px; width:
    100%;
    font-size: 16px;
}
```

4. **Output:**



Output of Registration Page

5. **Outcome of the Practical:** The outcome of implementing the described registration page will significantly enhance user experience and facilitate comprehensive data collection. The intuitive design of the form, with clearly labeled fields and a logical flow, ensures that users can easily understand what information is required, making the registration process straightforward and user-friendly. Additionally, the form's responsive layout, achieved through effective CSS styling, guarantees that it is visually appealing and functions well across various devices, including desktops, tablets, and smartphones.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**

- What specific HTML elements did you use to create the input fields for the registration form?
- How did you implement the password masking feature in the password field?
- What validation methods did you use to ensure that the email address entered is in the correct format?

- How did you structure the radio buttons for gender selection, and what attributes did you include?
- What CSS properties did you apply to enhance the layout and design of the registration form?
- How did you ensure that the address field is large enough for users to enter their complete address?
- What approach did you take to make the registration form responsive for different screen sizes?
- How did you organize the select boxes for the date of birth, and what options did you provide for each?
- What techniques did you use to style the checkboxes for language selection to improve user experience?
- How did you test the functionality of the registration form to ensure all fields work as intended?

Practical No. 6

1. **Experiment Description:** The validation process involves creating a JavaScript function that will be triggered when the user submits the registration form. This function will check the following rules:
   Name Validation:The name must contain only alphabetic characters (both uppercase and lowercase).
   The length of the name must be at least 6 characters.
   Password Validation:The password must be at least 6 characters long.
   HTML Structure: The registration form includes input fields for the name and password, along with a submit button. The form should also have an area to display validation messages.
   JavaScript Function: Create a function that retrieves the values of the name and password fields when the form is submitted. This function will perform the following checks:
   For the name, use a regular expression to ensure it contains only alphabetic characters and check the length.For the password, simply check the length.
   Display Validation Messages: If the validation fails, display appropriate error messages to the user, indicating what needs to be corrected. If both validations pass, allow the form to be submitted.
2. **Objective:** To validate the name and password with the given rules using javascript
3. **Code and Procedure:**

validation.html

```
<html >
<head>  <title>Registration Form</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>    <div class="container">
      <h2>Registration Form</h2>
      <form id="registrationForm">
         <div class="form-group">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" required>
            <span class="error" id="nameError"></span>          </div>
         <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <span class="error" id="passwordError"></span>          </div>
         <button type="submit">Register</button>
      </form>    </div>
   <script src="script.js"></script>
</body></html>
```

style.css

```
body {
   font-family: Arial, sans-serif;
   background-color: #f4f4f4;
   margin: 0;
   padding: 20px;
}
.container {
   max-width: 400px;
```

```
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h2 {
    text-align: center;
}
.form-group {
    margin-bottom: 15px;
}
label {
    display: block;
    margin-bottom: 5px;
}
.error {
    color: red;
    font-size: 0.9em;
}
button {
    width: 100%;
    padding: 10px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

script.js

```
document.getElementById('registrationForm').addEventListener('submit', function(event)
    { event.preventDefault();
    document.getElementById('nameError').textContent = '';
    document.getElementById('passwordError').textContent = '';
    const name = document.getElementById('name').value;
    const password = document.getElementById('password').value;
    let valid = true;
    // Validate Name
    if (!/^[a-zA-Z]+$/.test(name) || name.length < 6)
        { document.getElementById('nameError').textContent = 'Name must be at least 6 characters
        long
and contain only alphabets.';
        valid = false;
    }
    // Validate Password
    if (password.length < 6) {
        document.getElementById('passwordError').textContent = 'Password must be at least 6
characters long.';
        valid = false;
    }
```

// If valid, you can submit the form or perform further actions

```
(valid) {
   alert('Registration successful!');  // Here you can submit the form or perform other actions
   // e.g., this.submit();
}});
```

4. **Output:**



Output of Showing Validation of Name and Password rules

5. **Outcome of the Practical:** The implementation of JavaScript validation for the name and password fields significantly enhances user experience by providing immediate feedback on input errors, allowing users to correct mistakes before form submission. By enforcing rules that require the name to be at least 6 characters long and contain only alphabetic characters, as well as ensuring the password meets a minimum length, the validation process helps maintain data integrity and security. Clear error messages guide users in correcting their entries, reducing frustration and confusion.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**
   - What is the purpose of the JavaScript function in the validation process of the registration form?
   - How do you retrieve the values of the name and password fields when the form is submitted?
   - What regular expression did you use to validate that the name contains only alphabetic characters?
   - How do you check the length of the name to ensure it meets the minimum requirement of 6 characters?
   - What validation rules did you implement for the password, and how do you check its length?
   - How do you display validation messages to the user if the name or password validation fails?
   - What happens if both validations pass? How do you allow the form to be submitted?
   - How did you structure the HTML to include an area for displaying validation messages?
   - What techniques did you use to ensure that the validation function is triggered upon form submission?
   - How do you handle cases where the user enters invalid data multiple times, and what feedback do you provide

Practical No. 7

1. **Experiment Description:** The validation process involves creating a JavaScript function that will be triggered when the user submits the registration form. This function will check the following rules:
   Email ID Validation:The email must follow the standard format of name@domain.com.
   It should not contain any invalid characters, such as spaces or special characters that are not allowed in email addresses.
   Phone Number Validation:The phone number must consist of exactly 10 digits.
   It should not contain any letters, special characters, or spaces.
   HTML Structure: Ensure that the registration form includes input fields for the email ID and phone number, along with a submit button. The form should also have an area to display validation messages.
   JavaScript Function: Create a function that retrieves the values of the email and phone number fields when the form is submitted. This function will perform the following checks:
   For the email, use a regular expression to validate the format.
   For the phone number, check that it consists of exactly 10 digits.
   Display Validation Messages: If the validation fails, display appropriate error messages to the user, indicating what needs to be corrected. If both validations pass, allow the form to be submitted.
2. **Objective:** To validate the email and phone number with the given rules using javascript
3. **Code and Procedure:**

validation.html

```
<html >
<head>
   <title>Registration Form</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>    <div class="container">
      <h2>Registration Form</h2>
      <form id="registrationForm">
         <div class="form-group">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" required>
            <span class="error" id="nameError"></span>          </div>
         <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>
            <span class="error" id="emailError"></span>          </div>
         <div class="form-group">
            <label for="phone">Phone Number:</label>
            <input type="text" id="phone" name="phone" required>
            <span class="error" id="phoneError"></span>          </div>
         <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <span class="error" id="passwordError"></span>          </div>
         <button type="submit">Register</button>       </form>    </div>
   <script src="script.js"></script></body></html>
```

style.css

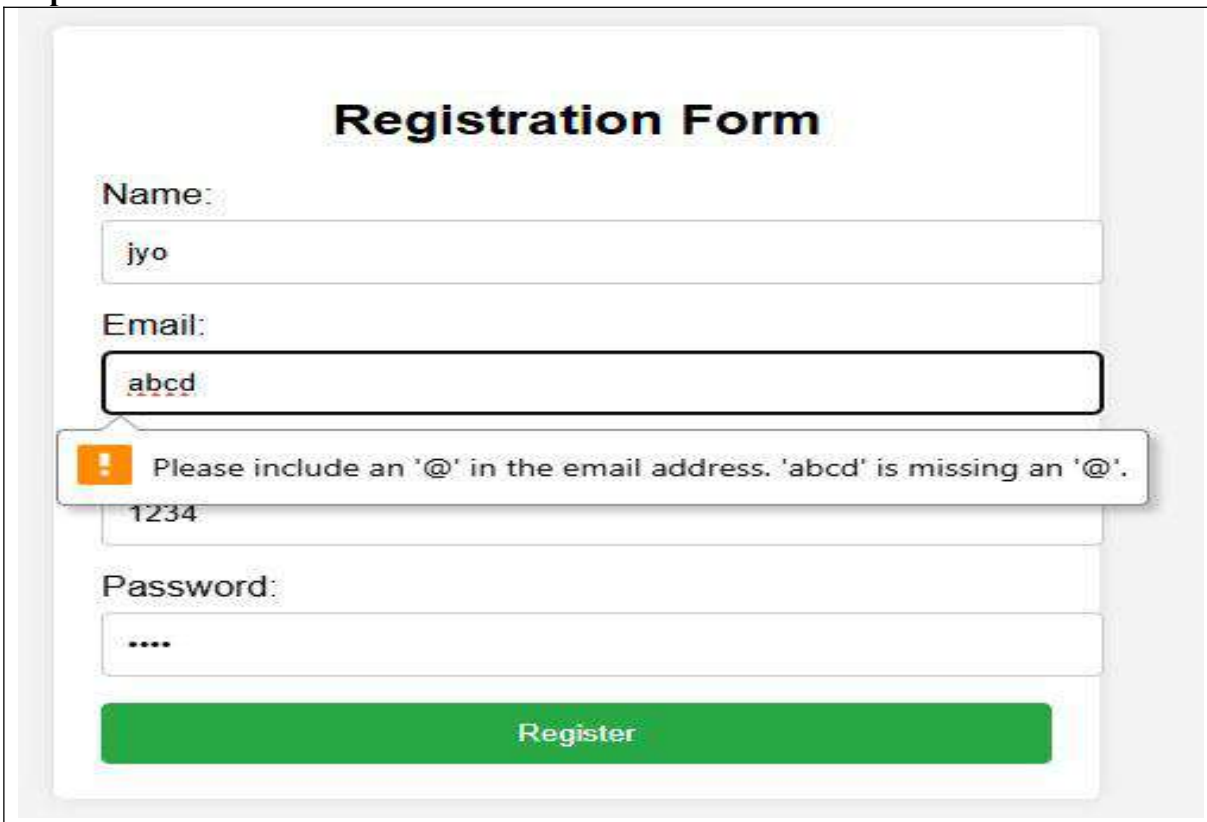
```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}
.container {
    max-width: 400px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h2 {
    text-align: center;
}
.form-group {
    margin-bottom: 15px;
}
label {
    display: block;
    margin-bottom: 5px;
}
.error {
    color: red;
    font-size: 0.9em;
}
button {
    width: 100%;
    padding: 10px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

script.js

```
document.getElementById('registrationForm').addEventListener('submit', function(event)
    { event.preventDefault();
    document.getElementById('nameError').textContent = '';
    document.getElementById('emailError').textContent = '';
    document.getElementById('phoneError').textContent = '';
    document.getElementById('passwordError').textContent = '';
    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const phone = document.getElementById('phone').value;
    const password = document.getElementById('password').value;
```

```
    let valid = true;
    // Validate Email
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; // Basic email validation
    if (!emailRegex.test(email)) {
        document.getElementById('emailError').textContent = 'Please enter a valid email address.';
        valid = false;
    }
    // Validate Phone Number
const phoneRegex = /^\d{10}$/; // Exactly 10 digits
     if (!phoneRegex.test(phone)) {
        document.getElementById('phoneError').textContent = 'Phone number must contain exactly 10
digits.';
        valid = false;
    }
```

4. **Output:**



Output of Showing Validation Email and Phone number rules

5. **Outcome of the Practical:** The implementation of JavaScript validation for the email and phone number fields significantly enhances user experience by providing immediate feedback on input errors, allowing users to correct mistakes before form submission. By enforcing rules that require the email format, as well as ensuring the phone number meets a minimum length and no inclusion of other characters, the validation process helps maintain data integrity and security. Clear error messages guide users in correcting their entries, reducing frustration and confusion.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**
   - What specific regular expression did you use to validate the format of the email address?
   - How do you ensure that the email validation checks for the presence of both the "@" symbol and a valid domain?
   - What criteria did you establish to determine if the email contains invalid characters?
   - How do you retrieve the value of the phone number field when the form is submitted?
   - What method do you use to check that the phone number consists of exactly 10 digits?
   - How do you handle cases where the phone number input contains letters or special characters?
   - What feedback mechanism do you implement to inform users of specific errors in their email or phone number inputs?
   - How did you structure the HTML to include input fields for both the email ID and phone number, as well as an area for validation messages?
   - What steps do you take to prevent the form from submitting if either the email or phone number validation fails?
   - How do you test the validation function to ensure it works correctly with various valid and invalid email and phone number inputs?

Practical No. 8

1. **Experiment Description:** Design a web page using CSS(Cascading Style Sheets) which includes the following:

- Use different font, styles: In the style definition you define how each selector should work(font, color etc.). Then, in the body of your pages, you refer to these selectors to activate the styles.
- Set a background image for both the page and single elements on the page.

2. **Objective:** Use of font styles and background styles of CSS
3. **Code and Procedure:**

inline.html (code for inline css styling)

```
<html>
<head><title>inline style sheet</title>
</head>
<body>normal bold:<b>web technologies </b>
<br> after applying inline style
<b style="color:red;font-size:22px;font-family:arial;text-decoration:underline">Web technologies</b>
</body></html>
```

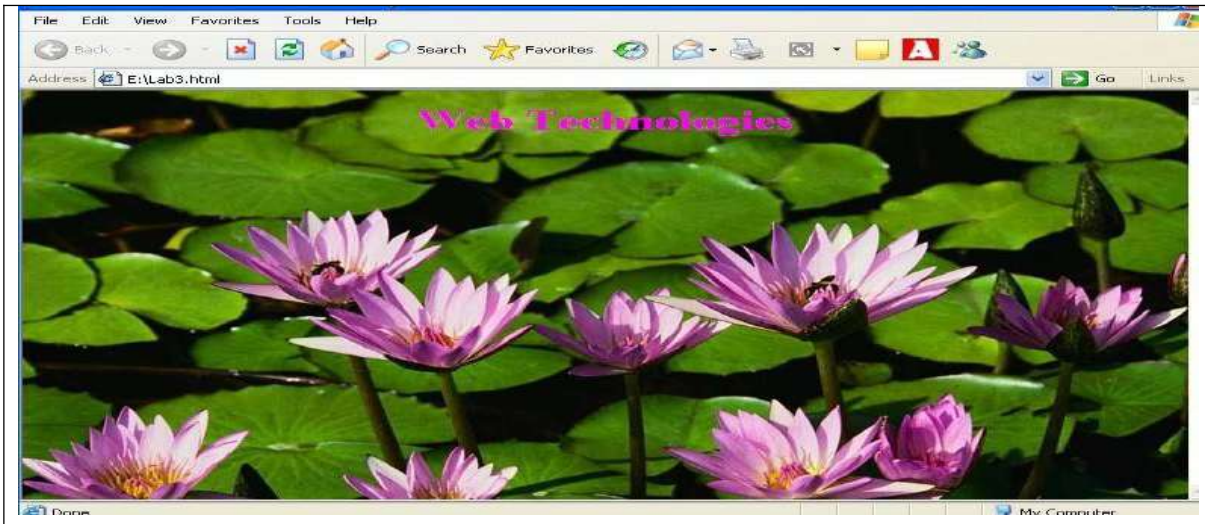
background.html (code for background css styling)

```
<html> <head>
<style type="text/css">
body
{
background-image: url("C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Sunset.jpg"); background-repeat:no-repeat
}
</style></head>
<body><center> <font color=fuchsia>
<h1>WebTechnologies</h1></center></body></html>
```

4. **Output:**



1.Output of inline text styling



3. Output of Background styling

5. **Outcome of the Practical:** The outcome of learning inline CSS styling for text decoration and background properties enables students to effectively manipulate the visual appearance of web elements. By applying inline styles, they gain hands-on experience with properties like text-decoration to create effects such as underlining or striking through text. Additionally, they learn to enhance designs using background properties like background-color and background-image. This practical knowledge fosters creativity and confidence in web design, providing a solid foundation for more advanced CSS techniques in future projects.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**
   - What are the different font styles you used in your CSS, and how did you apply them to various elements on the web page?
   - How do you define and use CSS selectors to apply styles to specific HTML elements?
   - What properties did you use to change the font color, size, and weight in your CSS styles?
   - How did you set a background image for the entire web page, and what CSS property did you use for this?
   - What techniques did you employ to ensure that the background image is responsive and displays correctly on different screen sizes?
   - How did you apply a background image to specific elements on the page, and what considerations did you take into account for visibility and readability?
   - What CSS properties did you use to control the positioning and repetition of the background images?
   - How do you ensure that the styles defined in your CSS are applied consistently across different browsers?
   - What methods did you use to test the visual appearance of your web page and ensure that the styles render as intended?
   - How did you organize your CSS code to maintain readability and ease of maintenance, especially when using multiple styles and selectors?

Practical No. 9

1. **Experiment Description:** Design with help of CSS
   - Control the repetition of the image with the background-repeat property.
   - Define styles for links as A:link A:visited A:active A:hover
2. **Objective:** Using CSS Control the repetition of the image with the background-repeat property and define styles for links.
3. **Code and Procedure:**

imagerepeat.html (Code for image repletion using CSS)

```
<html><head>
<styl type="text/css">
body{
background-image:url("D:\Sample Pictures\Winter.jpg");
background-repeat: repeat;
}
</style></head>
<body><center>
<font color=fuchsia><h1>satish</h1></font>
</center></body></html>
```

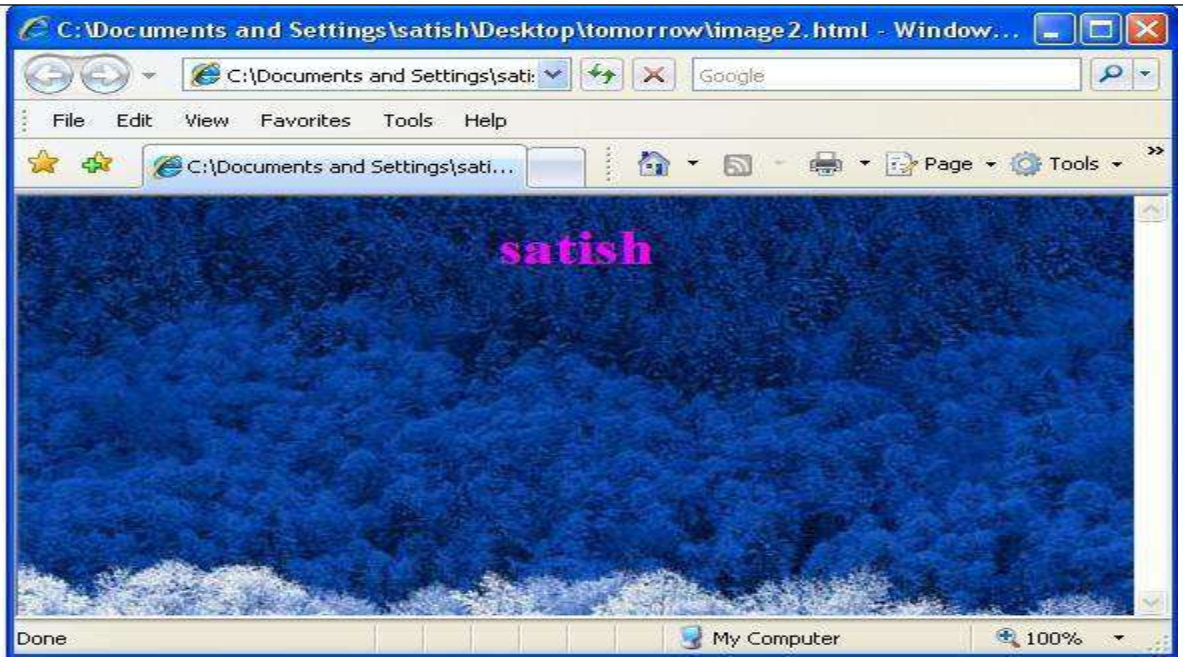
pseudoclassuse.html (code for the use of pseudoclass)

```
<html><head>
<title>external style sheet</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body bgcolor="black">
<font color=white >normal bold : <b>Web technologies</b><br>after applied embedded style :
<b class="headline">Web Technologies</b><br>
<b><a href="login.html" class="xlink">CROSS LINK</a><br>
<a href="login.html" class="hlink">HELP LINK</a><br><br>
<a href="login.html">another LINK</a></b>
</font></body></html>
```

style.css

```
b.headline {color:green; font-size:22px; font-family:arial; text-decoration:underline}
.xlink {cursor:crosshair;color:cyan}
.hlink{cursor:help;color:purple}
a:link {text-decoration:none;color:green} /* unvisited link */
a:visited {text-decoration: none;color:yellow} /* visited link */
a:active {text-decoration: none;color:blue} /* selected link */
a:hover {text-decoration: underline; color: red} /* mouse over link */
```

4. **Output:**



1. **Output of Image Repeat using CSS**



2. **Output of Pseudoclass links**

5. **Outcome of the Practical:** The outcome of using CSS to control background image repetition and define link styles with pseudo-classes enhances the visual appeal and usability of web pages. By applying the background-repeat property, students learn to manage how background images are displayed, creating seamless patterns or specific layouts. Additionally, defining styles for links using A:link, A:visited, A:active, and A:hover allows students to improve user interaction by providing clear visual feedback for different link states.

6. Assessment Criteria:

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. Viva Questions/Review Questions:

- What is the purpose of the background-repeat property in CSS, and how did you use it to control the repetition of background images?
- What values can be assigned to the background-repeat property, and how do they affect the display of the background image?
- How did you apply the background-repeat property to ensure that the background image appears only once or repeats in a specific direction?
- What are the different states of links in CSS, and how did you define styles for A:link, A:visited, A:active, and A:hover?
- How do the styles for A:link and A:visited differ, and why is it important to distinguish between them?
- What visual changes did you implement for the A:hover state to enhance user experience when interacting with links?
- How did you ensure that the styles for links are consistent across different browsers and devices?
- What CSS properties did you use to style the links, such as color, text-decoration, and font-weight?
- How do you test the functionality of the link styles to ensure they behave as expected in different states?
- What best practices did you follow when organizing your CSS code for link styles and background properties to maintain clarity and efficiency?

Practical No. 10

1. Experiment Description: Creating a To-Do List application using HTML, CSS, and JavaScript involves building a simple yet functional interface that allows users to add, view, and remove tasks. The To-Do List application will consist of three main components

HTML Structure: The HTML will provide the foundational structure of the To-Do List application. An input field where users can type in their tasks, A button to add the task to the list, An unordered list (<ul>) to display the tasks.

CSS Styling: The CSS will be used to style the application, making it visually appealing and user-friendly.

JavaScript Functionality: Adding Tasks: Implement a function that captures the input from the user when they click the "Add" button or press the Enter key. This function will create a new list item (<li>) containing the task text, Append the new item to the unordered list.

Removing Tasks: Each task will have a delete button (or icon) that allows users to remove the task from the list. Implement a function that removes the corresponding list item when the delete button is clicked.

Task Completion: Allow users to mark tasks as completed by clicking on the task itself. This can be achieved by toggling a class that applies a strikethrough style to the completed tasks

2. Objective: Develop static Webpage with HTML, CSS and JavaScript for to do list

3. Code and Procedure:

todo.html

```
<html ><head>
   <title>To-Do List</title>
   <link rel="stylesheet" href="style.css"></head>
<body>
   <div class="container">        <h1>To-Do List</h1>
      <input type="text" id="taskInput" placeholder="Add a new task...">
      <button id="addTaskButton">Add Task</button>
      <ul id="taskList"></ul>    </div>
   <script src="script.js"></script>
</body></html>
```

style.css

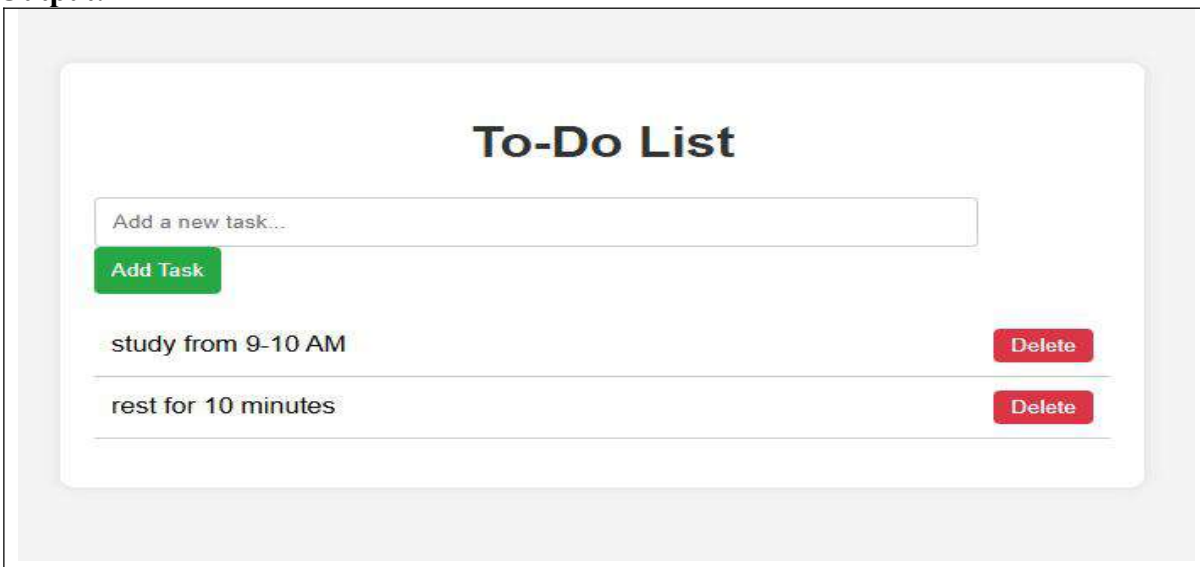
```
body {
   font-family: Arial, sans-serif;
   background-color: #f4f4f4;
   margin: 0;
   padding: 0;
}
.container {
   max-width: 600px;
   margin: 50px auto;
   padding: 20px;
   background: white;
   border-radius: 8px;
   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
   text-align: center;
   color: #333;
```

```
button {
    padding: 10px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
ul {
    list-style-type: none;
    padding: 0;
}
li {
    padding: 10px;
    border-bottom: 1px solid #ccc;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
li button {
    background-color: #dc3545;
    border: none;
    color: white;
    padding: 5px 10px;
    border-radius: 4px;
    cursor: pointer;
}
```

script.js

```
document.getElementById('addTaskButton').addEventListener('click', function()
    { const taskInput = document.getElementById('taskInput');
    const taskText = taskInput.value.trim();
    if (taskText === '') {
        alert('Please enter a task!');
        return;
    }
    const taskList = document.getElementById('taskList');
    const listItem = document.createElement('li');
    listItem.textContent = taskText;
    const deleteButton = document.createElement('button');
    deleteButton.textContent = 'Delete';
    deleteButton.addEventListener('click', function() {
        taskList.removeChild(listItem);
    });
    listItem.appendChild(deleteButton);
    taskList.appendChild(listItem);
    taskInput.value = ''; // Clear the input field
});
```

4. **Output:**



Output of To-Do List Static Web Page

5. **Outcome of the Practical:** Creating a To-Do List application using HTML, CSS, and JavaScript involves building a simple interface that allows users to manage tasks effectively. The HTML structure includes an input field for entering tasks, a button to add tasks, and an unordered list to display them. CSS is used to style the application, making it visually appealing and ensuring responsiveness across devices. JavaScript adds functionality by enabling users to add tasks to the list, remove tasks with a delete button, and mark tasks as completed by clicking on them.
6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**
   - What HTML elements did you use to create the structure of the To-Do List application, and how do they contribute to its functionality?
   - How did you implement the input field for users to type in their tasks, and what attributes did you include to enhance usability?
   - What JavaScript function did you create to capture user input when adding a new task, and how does it handle both button clicks and the Enter key?
   - How do you dynamically create a new list item (<li>) in JavaScript, and what steps are involved in appending it to the unordered list (<ul>)?
   - What method did you use to implement the delete functionality for each task, and how does the delete button interact with the corresponding list item?
   - How did you allow users to mark tasks as completed, and what CSS class did you toggle to visually indicate the completion of a task?
   - What CSS styles did you apply to make the To-Do List application visually appealing and user-friendly?
   - How did you ensure that the application is responsive and works well on different screen sizes?
   - What testing methods did you use to verify that the adding, removing, and completing tasks functionalities work as intended?
   - How did you organize your JavaScript code to maintain clarity and efficiency, especially when handling multiple functionalities within the To-Do List application?

Value Added Practicals
Practical No. 11

1. **Experiment Description:** To create a static webpage that visually represents a multi-step process. Users can navigate through the steps using "Next" and "Previous" buttons, and the current step will be highlighted. The progress bar will visually indicate the user's current position in the process.
   HTML Structure: Use a series of elements to represent each step in the process. Each step will contain a title and description. Include "Previous" and "Next" buttons to allow users to navigate through the steps.
   CSS Styling: Style the progress steps and buttons to make them visually appealing. This includes background colors, borders, and padding. Create a horizontal progress bar that visually indicates the user's current position in the process.
   JavaScript Functionality:Step Navigation: Implement functions to handle the navigation between steps. When the user clicks "Next," the current step should be hidden, and the next step should be displayed. Similarly, clicking "Previous" should return the user to the previous step. Update the progress bar based on the current step.
2. **Objective:** Develop static Webpage with HTML, CSS and JavaScript for progress steps.
3. **Code and Procedure:**

steps.html

```
<html ><head>    <title>Progress Steps</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <div class="container">
      <h1>Progress Steps</h1>
      <div class="progress-container">
         <div class="step" id="step1">
            <div class="step-number">1</div>
            <div class="step-title">Step 1</div> </div>
         <div class="step" id="step2">
            <div class="step-number">2</div>
           <div class="step-title">Step 2</div> </div>
         <div class="step" id="step3">
            <div class="step-number">3</div>
            <div class="step-title">Step 3</div> </div>
         <div class="step" id="step4">
            <div class="step-number">4</div>
            <div class="step-title">Step 4</div></div> </div>
      <div class="button-container">
         <button id="prevButton" disabled>Previous</button>
         <button id="nextButton">Next</button> </div> </div>
   <script src="script.js"></script>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}
.container {
    max-width: 600px;
    margin: 50px auto;
    padding: 20px;
    background: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
    text-align: center;
    color: #333;
}
.progress-container
    { display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
}
.step {
    flex: 1;
    text-align: center;
    position: relative;
}
.step-number {
    background-color: #e0e0e0;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    line-height: 40px;
    margin: 0 auto;
    font-weight: bold;
    color: #333;
}
.step-title {
    margin-top: 10px;
    color: #777;
}
.step.active .step-number
    { background-color:
    #4caf50; color: white;
}
.button-container
    { display: flex;
```

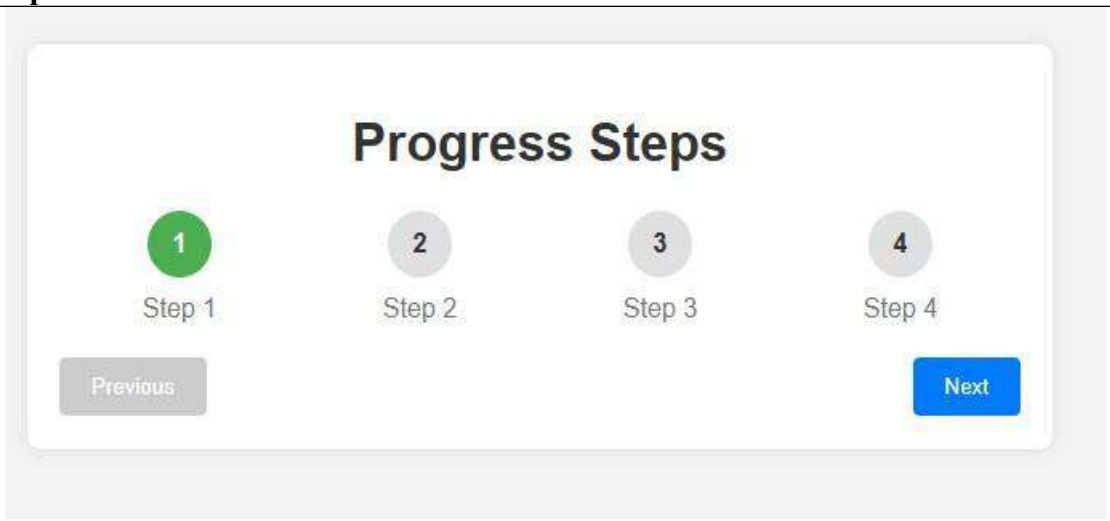
justify-

```
    content: space-between;
}
button {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #007bff;
    color: white;
}
button:disabled
    { background-color:
    #ccc; cursor: not-
    allowed;
}
```

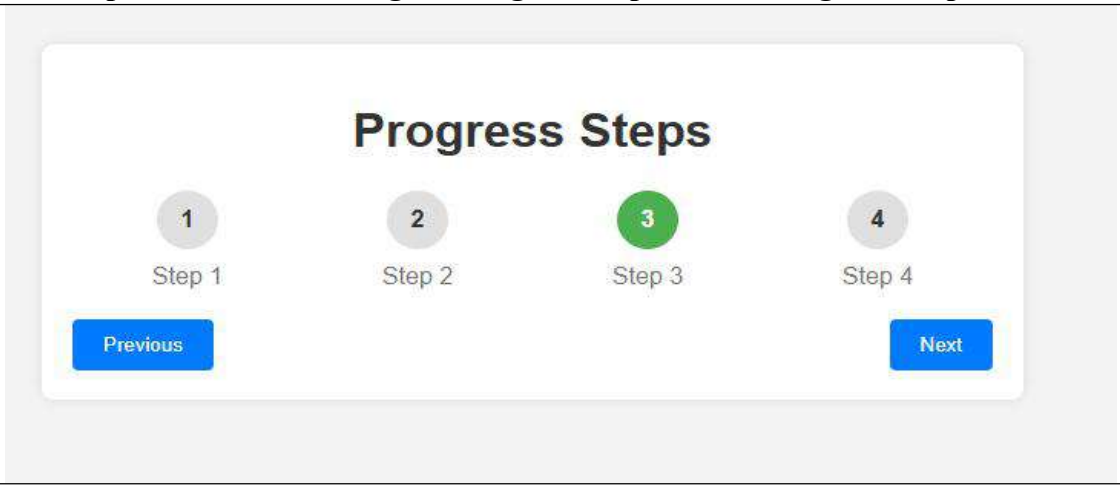
script.js

```
let currentStep = 0;
const steps = document.querySelectorAll('.step');
const nextButton = document.getElementById('nextButton');
const prevButton = document.getElementById('prevButton');
function updateStepDisplay() {
    steps.forEach((step, index) =>
        { step.classList.remove('active')
        ; if (index === currentStep) {
            step.classList.add('active');
        }
    });
    prevButton.disabled = currentStep === 0;
    nextButton.disabled = currentStep === steps.length - 1;
}
nextButton.addEventListener('click', () =>
    { if (currentStep < steps.length - 1) {
        currentStep++;
        updateStepDisplay();
    }
});
prevButton.addEventListener('click', () =>
    { if (currentStep > 0) {
        currentStep--;
        updateStepDisplay();
    }
});
// Initialize the display
updateStepDisplay();
    listItem.appendChild(deleteButton);
    taskList.appendChild(listItem);
    taskInput.value = ''; // Clear the input field
});
```

4. **Output:**



1. **Output of Static Web Page of Progress Steps On Clicking next or previous**



2. **On Clicking next or previous**



3. **On reaching At last step 4**

5. **Outcome of the Practical:** progress steps webpage results in an interactive and user-friendly interface that effectively guides users through a multi-step process. Users can navigate seamlessly between steps using "Next" and "Previous" buttons, with the current step clearly highlighted. The dynamic progress bar visually indicates the user's progress, enhancing the overall experience. The design is responsive and visually appealing, ensuring usability across various devices. Additionally, the JavaScript functionality ensures smooth transitions between steps and updates the progress bar accordingly.

6. Assessment Criteria:

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. Viva Questions/Review Questions:

- What HTML structure did you use to represent each step in the multi-step process, and how did you organize the content within elements?
- How did you implement the "Previous" and "Next" buttons in your HTML, and what attributes did you include to enhance their functionality?
- What CSS styles did you apply to the progress steps and buttons to make them visually appealing, and how did you ensure consistency across different steps?
- How did you create a horizontal progress bar, and what CSS properties did you use to visually indicate the user's current position in the process?
- What JavaScript functions did you implement to handle the navigation between steps, and how do these functions manage the visibility of the current step?
- How do you update the progress bar dynamically based on the current step, and what calculations are involved in determining the progress percentage?
- What methods did you use to ensure that the current step is highlighted visually, and how does this improve user experience?
- How did you handle edge cases, such as preventing navigation beyond the first or last steps?
- What testing strategies did you employ to verify that the navigation and progress bar functionalities work correctly across different browsers?
- How did you organize your JavaScript code to maintain clarity and efficiency, especially when managing multiple steps and their corresponding functionalities?

Practical No. 12

1. **Experiment Description**: To create a web-based Rock-Paper-Scissors game where users can select one of the three options (rock, paper, or scissors) and compete against a computer opponent. The game will consist of three main components: HTML for structure, CSS for styling, and JavaScript for game logic and interactivity.

   HTML Structure : Game Layout: The HTML will provide the foundational structure of the game, including: A title or heading to indicate the game, Three buttons for the user to select their choice: Rock, Paper, or Scissor, A display area to show the user's choice, the computer's choice, and the result of the game (win, lose, or draw).Scoreboard: Optionally, include a scoreboard to keep track of the number of wins, losses, and draws.

   CSS Styling: Visual Design: Use CSS to style the game interface, making it visually appealing and user-friendly. This can include: Background colors, fonts, and button styles to enhance the overall look,Responsive design to ensure the game works well on various screen sizes.

   JavaScript Functionality: Game Logic: Implement JavaScript functions to handle the game mechanics. This includes: Generating a random choice for the computer (rock, paper, or scissors).

   Comparing the user's choice with the computer's choice to determine the winner based on the game's rules: Rock beats Scissors, Scissors beats Paper, Paper beats Rock

   Display Results: Update the display area to show the user's choice, the computer's choice, and the result of the game after each round.

   Score Tracking: If a scoreboard is included, update the scores based on the outcome of each round.

2. **Objective:** With the help of HTML, CSS and JavaScript create any game (refering rock paper scissor game)

3. **Code and Procedure:**

game.html

```
<html ><head> <title>Rock, Paper, Scissors Game</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <div class="container">
      <h1>Rock, Paper, Scissors</h1>
      <div class="choices">
         <button class="choice" id="rock">Rock</button>
         <button class="choice" id="paper">Paper</button>
         <button class="choice" id="scissors">Scissors</button> </div>
      <div id="resultMessage"></div>
      <button id="restartGame" style="display: none;">Play Again</button> </div>
   <script src="script.js"></script>
</body> </html>
```

style.css

```
body {
   font-family: Arial, sans-serif;
   background-color: #f4f4f4;
   margin: 0;
   padding: 0;
}
.container {
```

```
    max-width: 400px;
    margin: 100px auto;
    padding: 20px;
    background: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.choices {
    margin: 20px 0;
}
.choice {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #007bff;
    color: white;
    margin: 0 10px;
}
#resultMessage
    { margin-top:
    20px; font-weight:
    bold;
}
#restartGame
    { margin-top:
    20px;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    background-color: #28a745;
    color: white;
}
```

script.js

```
const choices = ['rock', 'paper', 'scissors'];
const resultMessage = document.getElementById('resultMessage');
const restartGame = document.getElementById('restartGame');
document.querySelectorAll('.choice').forEach(button => {
    button.addEventListener('click', () =>
        { const userChoice = button.id;
        const computerChoice = choices[Math.floor(Math.random() * choices.length)];
        determineWinner(userChoice, computerChoice);
    });
});
function determineWinner(userChoice, computerChoice)
    { if (userChoice === computerChoice) {
        resultMessage.textContent = `It's a tie! You both chose ${userChoice}.`;
    } else if (
        (userChoice === 'rock' && computerChoice === 'scissors') ||
        (userChoice === 'paper' && computerChoice === 'rock') ||
```

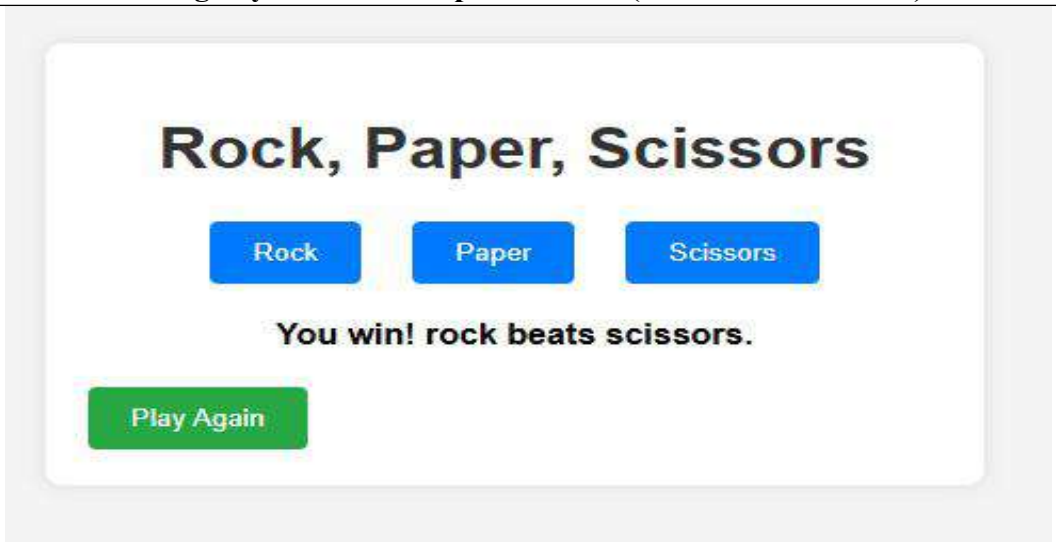
(userChoice === 'scissors' && computerChoice === 'paper')

```
) {
        resultMessage.textContent = `You win! ${userChoice} beats ${computerChoice}.`;
    } else {
        resultMessage.textContent = `You lose! ${computerChoice} beats ${userChoice}.`;
    }
    restartGame.style.display = 'block';}
restartGame.addEventListener('click', () => {
    resultMessage.textContent = '';
    restartGame.style.display = 'none';
});
```

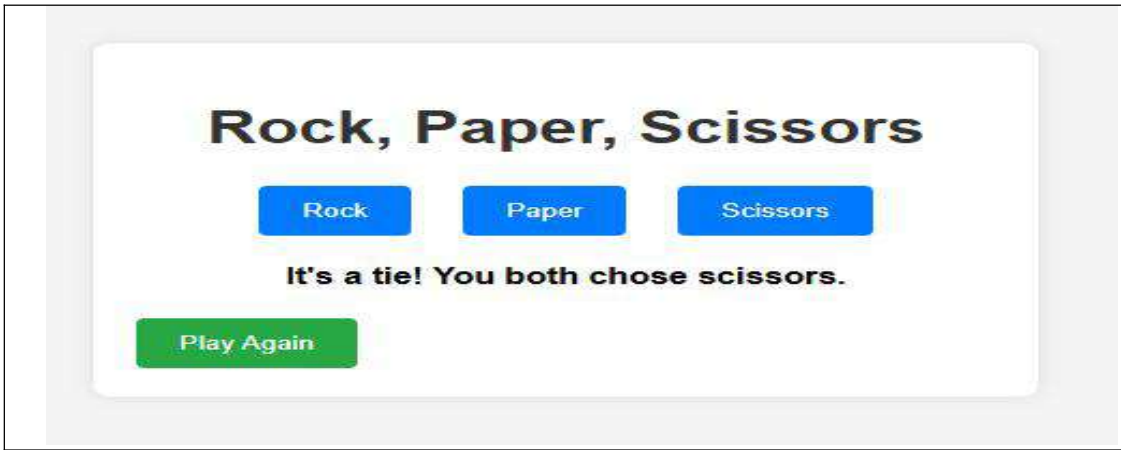
4. Output:



1. **On clicking any one Rock/ Paper / Scissor (here rock is choosen)**



2. **On playing again and choosing scissors**



3. Showing result of a tie game

5. **Outcome of the Practical:** The Rock-Paper-Scissors game results in an engaging and interactive web application that allows users to play the classic game against a computer opponent. Users can easily select their choice of rock, paper, or scissors through intuitive buttons, enhancing the overall user experience. The game logic effectively determines the winner based on the established rules, providing immediate feedback on the outcome of each round.

6. **Assessment Criteria:**

Record	3 Marks
Execution	4 Marks
Viva	3 Marks

7. **Viva Questions/Review Questions:**

- What HTML elements did you use to structure the Rock-Paper-Scissors game, and how do they contribute to the overall layout?
- How did you implement the buttons for user choices (Rock, Paper, Scissors), and what attributes did you include to enhance their functionality?
- What CSS styles did you apply to the game interface to make it visually appealing, and how did you ensure a user-friendly design?
- How did you ensure that the game interface is responsive and works well on various screen sizes?
- What JavaScript functions did you create to handle the game logic, and how do they generate a random choice for the computer?
- How do you compare the user's choice with the computer's choice to determine the winner, and what conditions are checked in the comparison?
- How did you update the display area to show the user's choice, the computer's choice, and the result of the game after each round?
- If a scoreboard is included, how do you implement the logic to track and update the number of wins, losses, and draws?
- What methods did you use to test the functionality of the game and ensure that the game logic works correctly?
- How did you organize your JavaScript code to maintain clarity and efficiency, especially when handling multiple game rounds and updating the scoreboard?

Textbooks and References

Textbooks

Book Title	Author	Description
HTML & CSS: The Complete Reference	Thomas Powell	A detailed reference for both beginners and advanced users.
Learning Web Design	Jennifer Niederst Robbins	A beginner-friendly guide that covers essential concepts of web design, including HTML and CSS.
Head First HTML and CSS	Elisabeth Robson, Eric Freeman	An engaging and visually rich introduction to HTML and CSS, ideal for beginners.
HTML5: The Missing Manual	Matthew MacDonald	An essential guide for understanding HTML5 and its features.

References

Book Title	Author	Description
HTML, CSS, and JavaScript All in One, Sams Teach Yourself	Julie C. Meloni	A comprehensive guide covering HTML, CSS, and JavaScript in a step-by-step format.
JavaScript: The Good Parts	Douglas Crockford	A deep dive into JavaScript, focusing on the language's best features and practices.
Eloquent JavaScript	Marijn Haverbeke	A modern introduction to programming using JavaScript, covering both basic and advanced topics.
JavaScript and JQuery: Interactive Front-End Web Development	Jon Duckett	A practical guide to using JavaScript and jQuery for interactive web development.