

**MIT Art, Design and Technology University**

**MIT School of Computing, Pune**

**Department of Information Technology**

|  |
| --- |
| **Lab Manual** |

# **Practical - Web Programming**

# **Class - S.Y. (SEM-II), DA**

# **Batch - DA-I**

# **Ms. Mrunali Patil**

ADT23SOCB0625

**A.Y. 2024 – 2025 (SEM-II)**

File Index page given in the stationary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Web Programming**  **SEMESTER – IV** | | | | | |
| **Course Code:** | | 23IT2008 | **Course Credits:** | 02 | |
| **Teaching Hours / Week (L:T:P):** | | 0:0:4 | **CA Marks:** | 25 | |
| **Total Number of Teaching Hours:** | |  | **END-SEM Marks:** | 25 | |
| **Course Pre-requisites:** | | | | | |
| **Course Description:**  This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript,PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.  The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.  By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. | | | | | |
| **Course Learning Objectives:** This course will enable the students to:   1. Understand fundamental concepts of front-end web development. 2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms. 3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design. 4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively. 5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming. 6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 7. Configure server environments with Apache/TOMCAT. 8. Set up a PHP development environment and write basic PHP scripts. 9. Master PHP programming constructs for web development tasks. 10. Create and process HTML forms, and manage MySQL database operations. 11. Develop comprehensive back-end applications using PHP and MySQL. | | | | | |
| **Course Outcome:** After taking this course, Students will be able to :   1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing. 2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects. 3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write & execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP for processing. 4. Design and develop a back-end application using PHP and MySQL, implementing CRUD operations to manage data effectively. | | | | | |
| **UNIT – I** | **Introduction to HTML and Cascading Style Sheet** | | | | **09 Hours** |
| Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms  Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, Background images | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/html-css-javascript-for-web-developers**](https://www.coursera.org/learn/html-css-javascript-for-web-developers) | | | | |
| **Self-study / Do it yourself /:**  **Practice creating basic HTML pages and enhancing them using CSS.** | | | | |
| **Experiential Learning Topics:**  **Design a simple webpage for coffee shop website** | | | | |
| **PBL - Project Based Learning:**  **Create a multi-page website (e.g., coffee shop website) using HTML and CSS.** | | | | |
|  | | | | | |
| **UNIT – II** | **Front-End Development** | | | | **09 Hours** |
| Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML  Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free).**  **Videos:**  [**https://www.coursera.org/learn/javascript-basics**](https://www.coursera.org/learn/javascript-basics) | | | | |
| **Self-study / Do it yourself /:**  **Solve exercises on JavaScript syntax, control structures, and functions** | | | | |
| **Experiential Learning Topics:**  **Build a web page with interactive elements (e.g., a simple calculator).** | | | | |
| **PBL - Project Based Learning:**  **Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.** | | | | |
|  | | | | | |
| **UNIT – III** | **Advanced Front-End Development** | | | | **09 Hours** |
| Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM  Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/building-interactive-web-pages-using-javascript**](https://www.coursera.org/learn/building-interactive-web-pages-using-javascript)  **Use tools like Visual Studio Code (free).** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on DOM traversal and event handling.** | | | | |
| **Experiential Learning Topics:**  **Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).** | | | | |
| **PBL - Project Based Learning:**  **Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.** | | | | |
|  | | | | | |
| **UNIT – IV** | **Server Side Scripting** | | | | **09 Hours** |
| Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.  Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,  Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  [**https://www.coursera.org/learn/web-applications-php**](https://www.coursera.org/learn/web-applications-php)  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management** | | | | |
| **Self-study / Do it yourself /:**  **Practice exercises on form handling and server-side scripting with PHP.** | | | | |
| **Experiential Learning Topics:**  **Create a basic form for data submission and handle it using PHP (e.g., feedback form).** | | | | |
| **PBL - Project Based Learning:**  **Develop a small server-side application (e.g., a contact form with email validation and submission).** | | | | |
|  | | | | | |
| **UNIT – V** | **Working with Databases and Web Application Development** | | | | **09 Hours** |
| Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO\_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.  Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop | | | | | |
| **Pedagogy** | **ICT Teaching / PowerPoint Presentation and Videos:**  **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**  **Videos:**  [**https://www.coursera.org/learn/web-app**](https://www.coursera.org/learn/web-app) | | | | |
| **Self-study / Do it yourself /:**  **Exercises on creating and manipulating databases using PHP and MySQL.** | | | | |
| **Experiential Learning Topics:**  **Create a database and design a webpage to display its data dynamically.** | | | | |
| **PBL - Project Based Learning:**  **Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.** | | | | |

**Text Books:**

1. "HTML and CSS: Design and Build Websites" by Jon Duckett.
2. "Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" by Jennifer Niederst Robbins.
3. Achyut Godbole & Atul Kahate, ‖Web Technologies: TCP/IP to Internet Application Architectures‖, McGraw Hill Education publications, ISBN, 007047298X, 9780070472983.
4. Ralph Moseley & M. T. Savaliya, ―Developing Web Applications‖, Wiley publications, ISBN 13 : 9788126538676.

**Reference Books:**

1. Eloquent JavaScript: A Modern Introduction to Programming by Marijn Haverbeke.
2. JavaScript: The Good Parts by Douglas Crockford.
3. CSS Secrets: Better Solutions to Everyday Web Design Problems by Lea Ver.
4. Web Technologies- Jeffery C. Jackson, ISBN 978-81-317-1715-8 Pearson 2015.
5. PHP Objects, Patterns, and Practice by Matt Zandstra
6. MySQL Cookbook by Paul DuBois.
7. Advanced PHP Programming - George Schlossnagle- ISBN 0-672-32561-6,2004.

**URLs (Optional) - List of Online Courses**

1. W3Schools HTML, CSS, JavaScript Tutorial: <https://www.w3schools.com/html/>
2. Mozilla Developer Network (MDN) Web Docs - HTML, CSS, JavaScript, DOM: <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML>
3. Project-Based Learning Resources:https://developer.mozilla.org/en-US/docs/Learn

**Contents beyond Syllabus:**

1. Web Essentials
2. Using JavaScript to handle form submission and login events (e.g., onsubmit, onclick)
3. JavaScript Form validations, General Input Validation, Password Validation
4. Storing user data (like a username) temporarily using localStorage or sessionStorage
5. Dynamically updating the content of the webpage, such as displaying a welcome message
6. Redirecting users using window.location

**List of Experiments:**

In this series of assignments, you will create a coffee shop / any other website step by step. Each assignment will focus on a different aspect of the website, covering various HTML elements, CSS, JavaScript, PHP and MySQL concepts.

**Laboratory/Project Assignment Guidelines:**

1. Project Selection:
   * Each student must select a unique project topic for their laboratory assignments.
   * The chosen project topic should align with the concepts covered in the course syllabus.
   * The chosen project topic should be approved by the course coordinator/ subject teacher.
   * Students have the freedom to choose their project topics based on their interests and career aspirations.
   * Project topics may include but are not limited to:
     + E-commerce website
     + Blogging platform
     + Online booking system
     + Content management system (CMS)
     + Discussion forum
     + Social networking platform
     + Task management application
     + Portfolio website
2. Laboratory Assignments:
   * Throughout the course, students will complete laboratory assignments related to their chosen project topic.
3. Evaluation Criteria:
   * The laboratory assignments and the final project will be evaluated based on criteria such as Structure and Semantics, Content Organization, Forms and Inputs, Links and Navigation, Styling and Layout, Design Consistency, Functionality, Code Quality and adherence to project requirements.
   * Students are expected to demonstrate creativity, and a comprehensive understanding of web development principles in their projects.
   * The laboratory assignments based on chosen project topics will be assessed based on several key criteria that reflect both technical proficiency and creative application in web development. These include:

* Structure & Semantics: Proper use of HTML to create a logical, accessible structure with meaningful and semantically correct elements.
* Content Organization: Clear and intuitive organization of content, ensuring ease of navigation and logical flow throughout the site.
* Forms & User Input: Effective implementation of forms and user input elements that are functional, validated, and accessible.
* Links & Navigation: Well-structured navigation and functional links that provide a seamless user experience.
* Styling & Layout: Visually appealing and responsive design, with a well-executed layout that adapts to various screen sizes.
* Design Consistency: Uniformity in design elements, including colors, typography, and spacing, to maintain a cohesive look and feel across the site.
* Functionality: Full functionality of all interactive elements, ensuring a bug-free, smooth experience for users.
* Code Quality & Best Practices: Clean, well-organized, and efficient code that adheres to modern web development best practices and is easy to maintain.

1. Submission and Presentation:
   * The project and project report/journal must be submitted within the specified deadline and should meet the specified requirements outlined by the course coordinator/ subject teacher.

Project Problem Statement-

Design and develop a basic website for a local coffee shop using HTML, CSS, JavaScript, PHP and MySQL. This website will serve as an online presence for the coffee shop, effectively communicating the brand identity, showcasing the menu, providing essential information, and allowing customers to easily get in touch or locate the shop.

The project directory is as follows:

coffee-shop-website/

├── css/

│ └── styles.css

├── html/

│ ├── index.html

│ ├── menu.html

│ ├── about.html

│ └── contact.html

├── js/

│ └── scripts.js

└── images/

|  |  |
| --- | --- |
| 1. | Create the basic structure of the coffee shop website, including the home page layout with a header, main content area, and footer.  Prepare a common project website design and plan document for all assignments. Consider following points:   1. Brief information about the project. 2. Set the goals & deliverables. 3. Finalize the modules of the project. 4. Define the audience. 5. Describe pain points & the ideal experience (On the basis of existing systems) 6. Set the visual direction 7. Map out the Project structure. 8. Plan the content for each page. 9. Add ideas for content, images & layout. 10. Determine your site structure or Create content for your core website pages:     1. Home page     2. About page     3. Product/Service page     4. Testimonial/review page     5. Contact page     6. Starter blog posts 11. Create and collect design elements   These design elements define your brand personality and help customers feel what your brand represents through the use of:   * 1. Colors   2. Fonts and typography   3. Logos   4. Images and photos |
| 2. | HTML   1. Create a detailed home page for the coffee shop website. 2. Create a detailed menu/product page for the coffee shop website, listing all available items categorized appropriately. 3. Create a cart page that allows customers to review and manage the items they wish to purchase before proceeding to checkout. 4. Create an about us page that provides detailed information about the coffee shop’s history, mission, and team. 5. Create a contact page that allows customers to easily get in touch with the coffee shop through a form. 6. Design and implement admin/user registration form for the coffee shop website. 7. Design and implement admin/user login form for the coffee shop website. |
| 3. | CSS   1. Enhance the layout of the coffee shop website using CSS Grid for the home page. 2. Use CSS Grid to layout the menu/product items in a structured and style the menu categories with appropriate headings, spacing, separators, images, descriptions, and prices. |
| 4. | CSS   1. Enhance the cart page to make it user-friendly and visually appealing. Style the cart items with appropriate margins, paddings, and input field styles to provide a seamless shopping experience. 2. Enhance and style the about us page with appropriate margins, paddings, and input field styles. 3. Enhance and style the contact page to make it user-friendly and visually appealing. Style the contact form with appropriate margins, paddings, and input field styles. 4. Enhance and style the admin/user registration form with appropriate margins, paddings, and input field styles. 5. Enhance and style the admin/user login form with appropriate margins, paddings, and input field styles. |
| 5. | JavaScript   1. Implement user registration and login forms for the coffee shop website. These forms will allow users to create an account, log in, and access personalized features, such as saving favorite items or viewing order history.   User Registration Form will allow new customers to sign up and create an account on the website. The form will capture basic user details, including the name, email address, and password (not limited to these fields).  User Login Form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.   1. Provide validations for user registration and login forms to validate the input to ensure that all required fields are filled and that the email format is valid. (**Contents beyond Syllabus)** 2. Develop cart functionality to allow users to add items, update quantities, and remove items. |
| 6. | JavaScript   1. The user login form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user. 2. If the login is successful, the user should be redirected to the homepage or their user dashboard. (**Contents beyond Syllabus)** 3. Use localStorage or sessionStorage to store authentication data, such as the user’s email and login status. This ensures that once a user is logged in, they remain authenticated even after the page reloads or when they visit the site again. (**Contents beyond Syllabus)** 4. Save the cart data to local storage when items are added, updated, or removed. Retrieve and load the cart data from local storage when the page loads. (**Contents beyond Syllabus)** |
| 7. | PHP   1. Develop a PHP script to handle user registration for the Coffee Shop website. The script should accept input from users for their name, email address, password, etc. (all required fields for registration). 2. Implement error handling to notify users of any issues during registration, such as validation errors. 3. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page. |
| 8. | PHP   1. Develop a PHP script to handle user login for the Coffee Shop website. The script should accept input from users for their login credentials. (all required fields for login). 2. Provide feedback to the user upon successful login, either through a confirmation message or a redirect to a welcome page. 3. Implement error handling to notify users of login failures due to incorrect credentials or other errors. 4. Provide feedback to the user upon successful login, either through a welcome user name message or a redirect to a home page. |
| 9. | PHP and MySQL   1. Develop a PHP script that allows users to manage their shopping cart for an e-commerce website (e.g., a Coffee Shop store). The script should allow users to add items to their cart, view their cart contents, and remove items if needed. 2. Develop a PHP script to manage the shopping cart for an e-commerce website (e.g., a Coffee Shop store) using MySQL. This script should allow users to add items to their cart, view their cart contents, and remove items from the cart. The cart data should be stored in the MySQL database to allow persistence across sessions. |
| 10. | PHP and MySQL   1. Develop a PHP script to handle the checkout process for users who are ready to complete their purchase. The script should process the cart data and provide feedback to the user upon successful or failed checkout. 2. Develop a PHP script that processes the checkout process for users who are ready to complete their purchase, integrating the MySQL database for handling user and order information. The script should validate user input, process the cart data, and provide feedback upon successful or failed checkout. |

# 

## 

## **Experiment No.1**

**Hospital Appointment Booking System**

**Project Website Design & Plan Document**  
**1. Problem Statement**  
Design and implement a clean, responsive, and user-friendly hospital appointment booking system that allows patients to search, book, and manage appointments with doctors. The platform must include pages for scheduling, doctor listing, patient login/signup, contact support, and admin management, ensuring seamless navigation and accessibility.

**2. Brief Information about the Project**

**Project Title**: **QuickClinic – Hospital Appointment Booking Platform**  
**Description**:  
**QuickClinic** is a digital platform developed to streamline hospital appointments for patients, doctors, and hospital staff. It includes patient registration, doctor availability, scheduling, and an optional admin panel. The system improves hospital efficiency and offers patients a modern way to manage healthcare visits from home or mobile.

**3. Goals & Deliverables**

**Goals**:

* Design a responsive web portal for patients and doctors.
* Allow users to book, cancel, or reschedule appointments.
* Show real-time doctor availability and department listings.
* Include patient authentication (login/signup).
* Integrate optional backend for admin controls.

**Deliverables**:

* Multi-page platform with a consistent layout
* Patient registration and login pages
* Appointment booking module
* Doctor directory with filters (specialty, availability)
* Contact page with form integration (JavaScript, PHP, MySQL)
* Responsive design with mobile support
* Admin dashboard (optional)

**4. Finalized Project Modules**

* Homepage with system overview
* Login/Signup for Patients
* Doctor List & Department Page
* Book Appointment Page
* Appointment History Page
* Contact Page with DB Integration
* Admin Panel for Doctors and Appointments (future enhancement)
* Footer with Help, Privacy Policy, and Social Links

**5. Target Audience**

* General patients looking to book appointments
* Hospitals and clinics managing appointments
* Doctors who want to display their availability
* Healthcare administrators

**6. Pain Points & Ideal Experience**

**Pain Points in Existing Systems**:

* Complex or non-intuitive interfaces
* Poor mobile responsiveness
* No reminders or status tracking
* Manual or delayed appointment processes

**Ideal Experience**:

* Quick and guided booking flow
* Clean UI with medical focus
* Mobile-first responsive layout
* Reminders and history of appointments
* Doctor search by name, department, or timing

**7. Visual Direction**

Here is the **Visual Direction** section for **QuickClinic** formatted in a clean and organized **table format**:

**Visual Direction – Table Format**

| **Element** | **Details** |
| --- | --- |
| **Tone** | Clean, calming, and professional healthcare interface |
| **Primary Color** | #00796B – Calming green used for primary buttons and links |
| **Secondary Color** | #004D40 – Darker green for hover states and emphasis |
| **Accent Color** | #FF7043 – Bright orange used for alerts, call-to-actions, or highlights |
| **Background (Light)** | #FFFFFF – Clean white used for main content areas |
| **Background (Alt)** | #F1F8F6 – Light green background used for cards, sections |
| **Text (Light Mode)** | #212121 – Standard dark grey used for readable body text |
| **Text (Dark Mode)** | #E0E0E0 – Light grey used for text on darker backgrounds |
| **Typography – Headings** | Poppins – Bold and modern font for headings |
| **Typography – Body** | Open Sans – Simple and readable font for paragraphs and general text |
| **Typography – Code/Labels** | Roboto Mono – Used optionally for appointment IDs, tags, etc. |
| **Image Style** | Clean, professional, and warm hospital-themed photos; doctor avatars; status icons |
| **Layout Approach** | Grid and Flexbox-based design for responsive behavior; card-based sections for listings and appointment info |
| **Button Design** | Rounded corners, hover color changes (secondary color), smooth shadows |
| **Icon Style** | Flat or outlined icons with hover animation; used for actions (e.g., edit, cancel, contact, schedule) |
| **Form Elements** | Simple bordered inputs with label-focus animation and error/highlight states in accent color (#FF7043) |
| **Logo Style** | Wordmark: “QuickClinic” in Poppins, bold; color variants for light and dark backgrounds |

**Google Fonts**:

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@600&family=Open+Sans:wght@400;600&display=swap');

body {

font-family: 'Open Sans', sans-serif;

}

h1, h2, h3 {

font-family: 'Poppins', sans-serif;

}

**8. Project Folder Structure**

QuickClinic/

│

├── index.html (Home)

├── login.html

├── register.html

├── doctors.html

├── book.html

├── history.html

├── contact.html

│

├── /css/

│ └── style.css

│

├── /js/

│ └── app.js (form handling, date validation)

│

├── /php/

│ └── submit-form.php

│

├── /images/

│ └── clinic-logo.png

│

├── /admin/ (optional backend)

│ └── dashboard.html

**9. Content Plan for Pages**

**Homepage**

* Welcome banner
* Quick links: Book Appointment, View Doctors
* Highlights: Hospital services
* Footer with contact info

**Login/Signup Page**

* Patient login
* Registration form with validation
* Links to booking system

**Doctor Listing Page**

* Filterable by department
* Doctor cards with profile, specialty, and available times

**Book Appointment Page**

* Select doctor, date, and time
* Form submission to database
* Confirmation and rescheduling option

**Appointment History Page**

* List of past and upcoming appointments
* Cancel or modify booking

**Contact Page**

* Contact form: name, email, message
* Embedded map (optional)
* Hospital contact details

**10. Content, Image & Layout Ideas**

**Image Guidelines**:

* Professional hospital/clinic photos
* Doctor avatars and icons
* Clean icons for status, departments, date/time

**Layout Elements**:

* Cards for doctor listings
* Form-based booking flow
* Table layout for appointment history
* Grid layout with responsive behavior using CSS Flexbox/Grid

**Content Types**:

* Services info
* Doctor profiles
* Appointment instructions
* Patient testimonials (optional)

**11. CSS Variables for Theming**

:root {

--primary-color: #00796B;

--secondary-color: #004D40;

--accent-color: #FF7043;

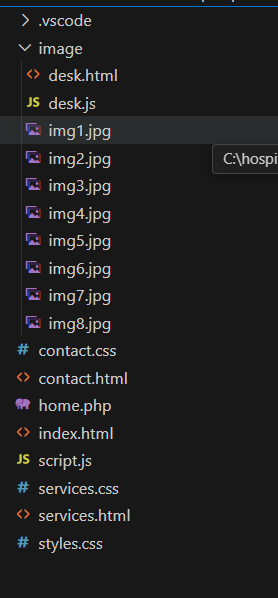
--text-light: #212121;

--text-dark: #E0E0E0;

--background: #FFFFFF;

--background-alt: #F1F8F6;

}



**12. Conclusion**

**QuickClinic** is a structured and modern hospital appointment system that simplifies booking, viewing, and managing appointments for both patients and doctors. It features a calming design, interactive modules, and responsive pages. This document outlines all required components, visual themes, content structure, and folder organization, making it a strong foundation for real-time deployment and future expansion (admin backend, notifications, etc.).

## **Experiment No.2**

## **Problem Statement:**

**Problem Statement:**

Develop a multi-page **Hospital Appointment Booking System** website using HTML. The goal is to provide a user-friendly interface where patients can view doctors, book appointments, and interact with hospital information. The system should include the following web pages:

* **Home Page**: A landing page introducing the hospital and key features.
* **Doctors Page**: Lists available doctors categorized by specialization.
* **Appointments Page (Cart Equivalent)**: Allows users to review and manage their booked appointments.
* **About Us Page**: Describes the hospital’s background, mission, and medical team.
* **Contact Page**: Includes a form for patient queries and contact information.
* **Registration Form**: For new users and admins to sign up.
* **Login Form**: For users and admins to securely log into the system.

This project will focus on using **HTML** to structure and design each of these pages without backend logic.

**Theory:**

**HTML (HyperText Markup Language)** is the foundational markup language used for creating and structuring content on the web. In this project, HTML is used to design a static hospital appointment system that includes multiple informational and interactive pages.

**HTML Elements Used:**

* **Headings (<h1> to <h6>)**: Define titles and sections.
* **Paragraphs (<p>)**: Display descriptive text.
* **Links (<a>)**: Navigate between different pages.
* **Images (<img>)**: Display hospital or doctor visuals.
* **Lists (<ul>, <ol>, <li>)**: Organize services, doctors, or specialties.
* **Tables (<table>)**: Used optionally to show schedules or doctor details.
* **Forms (<form>, <input>, <textarea>, <button>)**: Collect user information for appointments, registration, and contact.

**Features of HTML:**

1. **Platform Independent**: Works across all web browsers and systems.
2. **Lightweight**: Easy to load and fast rendering.
3. **Multimedia Support**: Can include videos, images, and audio.
4. **Semantic Tags (HTML5)**: Improve structure and accessibility (<section>, <nav>, <article>, <footer>).
5. **Ease of Integration**: Works seamlessly with CSS for styling and JavaScript for interactivity.

**Advantages:**

* Simple to learn and implement.
* Clear structure for building multi-page websites.
* Standardized and browser-friendly.
* Forms the basis for more dynamic systems with CSS/JS.

**Limitations:**

* **Static by nature**: No logic or backend functionality (e.g., actual appointment booking needs server-side code).
* **No storage**: Cannot save data without external scripting or databases.

## **Code:**

A. Home page:

code: <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Hospital Appointment System</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <nav>

            <h1>Quick Clinic</h1>

            <ul>

                <li><a href="#home">Home</a></li>

                <li><a href="#services">Services</a></li>

                <li><a href="#doctors">Doctors</a></li>

                <li><a href="contact.html">Contact</a></li>

            </ul>

        </nav>

    </header>

    <section id="home" class="hero">

        <h2>Your Health, Our Priority</h2>

        <p>Book an appointment with top doctors in just a few clicks.</p>

        <button class="btn" onclick="openBookingForm()">Book Appointment</button>

    </section>

    <section id="services">

        <h2>Our Services</h2>

        <div class="service-list">

            <div class="service">General Checkup</div>

            <div class="service">Emergency Care</div>

            <div class="service">Specialist Consultation</div>

        </div>

    </section>

    <!-- Booking Form (Initially Hidden) -->

    <div id="booking-form" class="form-popup">

        <div class="form-container">

            <h2>Book an Appointment</h2>

            <form id="appointmentForm"> <!-- Added form tag -->

                <label for="name">Full Name:</label>

                <input type="text" id="name" placeholder="Enter your name" required>

                <label for="email">Email:</label>

                <input type="email" id="email" placeholder="Enter your email" required>

                <label for="date">Preferred Date:</label>

                <input type="date" id="date" required>

                <label for="doctor">Select Doctor:</label>

                <select id="doctor">

                    <option>Dr. John Doe (Cardiologist)</option>

                    <option>Dr. Sarah Smith (Dermatologist)</option>

                    <option>Dr. Adam Lee (General Physician)</option>

                </select>

                <button type="submit" class="btn">Confirm Booking</button>

                <button type="button" class="btn close-btn" onclick="closeBookingForm()">Cancel</button>

            </form>

        </div>

    </div>

    <div class="appointment-section">

        <h2>Your Health, Our Priority</h2>

        <p>Book an appointment with top doctors in just a few clicks.</p>

        <button id="bookBtn">Book Appointment</button>

        <div id="popup" class="popup hidden">

          <h3>✅ Appointment Confirmed</h3>

          <p>Your appointment has been successfully booked. We will contact you soon.</p>

          <button id="okBtn">OK</button>

        </div>

      </div>

    <script src="script.js"></script>

    <footer>

        <p>&copy; 2025 CarePlus Hospital. All Rights Reserved.</p>

    </footer>

</body>

</html>

## **Output:**

A. Index/Home page output:

## **Code:**

B. menu/product page:

Code: <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Our Services - CarePlus Hospital</title>

    <link rel="stylesheet" href="services.css">

</head>

<body>

    <!-- Services Navigation Bar -->

    <nav class="services-nav">

        <ul>

            <li><a href="index.html">Home</a></li>

            <li><a href="#general">General Checkup</a></li>

            <li><a href="#emergency">Emergency Care</a></li>

            <li><a href="#specialist">Specialist Consultation</a></li>

        </ul>

    </nav>

    <!-- Services Section -->

    <section id="services">

        <h2 class="section-title">Our Healthcare Services</h2>

        <div class="services-container">

            <div class="service-card" id="general">

                <img src="image/img1.jpg" alt="General Checkup">

                <h3>General Checkup</h3>

                <p>Routine health checkups to ensure your well-being.</p>

            </div>

            <div class="service-card" id="emergency">

                <img src="image/img3.jpg" alt="Emergency Care">

                <h3>Emergency Care</h3>

                <p>24/7 emergency medical assistance for critical conditions.</p>

            </div>

            <div class="service-card" id="specialist">

                <img src="image/img2.jpg" alt="Specialist Consultation">

                <h3>Specialist Consultation</h3>

                <p>Expert doctors to diagnose and treat specific health issues.</p>

            </div>

        </div>

    </section>

    <!-- Footer -->

    <footer>

        <p>&copy; 2025 Quickclinic. All Rights Reserved.</p>

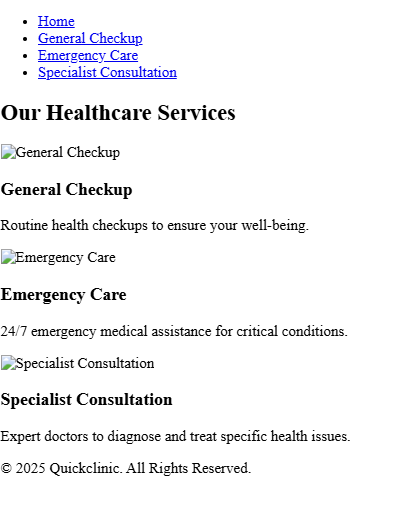
    </footer>

</body>

</html>

## **Output:**

1. menu/product page output:



## **Code:**

D. about us page:

code: <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Contact Us - CarePlus Hospital</title>

    <link rel="stylesheet" href="contact.css">  <!-- Linking External CSS -->

</head>

<body>

    <div class="container">

        <h2>Contact Us</h2>

        <div class="contact-info">

            <h3>CarePlus Hospital</h3>

            <p>📍 Address: 123, CarePlus Hospital, Pune, India</p>

            <p>📞 Phone: +91 98765 43210</p>

            <p>📧 Email: careplushospital@example.com</p>

            <p>🕒 Working Hours: Monday - Saturday (9:00 AM - 7:00 PM)</p>

        </div>

        <div class="social-icons">

            <h3>Follow Us</h3>

            <a href="https://facebook.com" target="\_blank">Facebook</a> |

            <a href="https://twitter.com" target="\_blank">Twitter</a> |

            <a href="https://instagram.com" target="\_blank">Instagram</a> |

            <a href="https://linkedin.com" target="\_blank">LinkedIn</a>

        </div>

    </div>

</body>

</html>

## 

## **Output:**

1. about us page output:



## **Code:**

F. registration page:

Code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Hospital Appointment System</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <nav>

            <h1>Quick Clinic</h1>

            <ul>

                <li><a href="#home">Home</a></li>

                <li><a href="#services">Services</a></li>

                <li><a href="#doctors">Doctors</a></li>

                <li><a href="contact.html">Contact</a></li>

            </ul>

        </nav>

    </header>

    <section id="home" class="hero">

        <h2>Your Health, Our Priority</h2>

        <p>Book an appointment with top doctors in just a few clicks.</p>

        <button class="btn" onclick="openBookingForm()">Book Appointment</button>

    </section>

    <section id="services">

        <h2>Our Services</h2>

        <div class="service-list">

            <div class="service">General Checkup</div>

            <div class="service">Emergency Care</div>

            <div class="service">Specialist Consultation</div>

        </div>

    </section>

    <!-- Booking Form (Initially Hidden) -->

    <div id="booking-form" class="form-popup">

        <div class="form-container">

            <h2>Book an Appointment</h2>

            <form id="appointmentForm"> <!-- Added form tag -->

                <label for="name">Full Name:</label>

                <input type="text" id="name" placeholder="Enter your name" required>

                <label for="email">Email:</label>

                <input type="email" id="email" placeholder="Enter your email" required>

                <label for="date">Preferred Date:</label>

                <input type="date" id="date" required>

                <label for="doctor">Select Doctor:</label>

                <select id="doctor">

                    <option>Dr. John Doe (Cardiologist)</option>

                    <option>Dr. Sarah Smith (Dermatologist)</option>

                    <option>Dr. Adam Lee (General Physician)</option>

                </select>

                <button type="submit" class="btn">Confirm Booking</button>

                <button type="button" class="btn close-btn" onclick="closeBookingForm()">Cancel</button>

            </form>

        </div>

    </div>

    <div class="appointment-section">

        <h2>Your Health, Our Priority</h2>

        <p>Book an appointment with top doctors in just a few clicks.</p>

        <button id="bookBtn">Book Appointment</button>

        <div id="popup" class="popup hidden">

          <h3>✅ Appointment Confirmed</h3>

          <p>Your appointment has been successfully booked. We will contact you soon.</p>

          <button id="okBtn">OK</button>

        </div>

      </div>

    <script src="script.js"></script>

    <footer>

        <p>&copy; 2025 Quickclinic. All Rights Reserved.</p>

    </footer>

</body>

</html

## Conclusion: The Hospital Appointment Booking System designed using HTML serves as the foundational structure for a healthcare-focused web application. Through multiple static pages—such as Home, Doctor Listings, Appointment Review, About Us, Contact, Registration, and Login—users can easily navigate and understand the hospital’s offerings and services.

**EXPERIMENT-03**

**Problem Statement**

**Enhance the layout of the Hospital Appointment Booking System homepage using CSS Grid.** Use CSS Grid to structure appointment listings, doctor categories, and service sections with headings, spacing, images, brief descriptions, and action buttons (e.g., "Book Now").

**Theory: CSS Grid for Hospital Appointment Booking System**

**Introduction to CSS Grid**

CSS Grid Layout is a powerful two-dimensional layout system for web development. Unlike Flexbox, which handles layout in a single direction (row or column), Grid supports both rows and columns. This makes it ideal for a hospital web platform where structured content like doctor cards, departments, and schedules need to be arranged clearly and responsively.

**Why CSS Grid for Hospital Systems?**

In a Hospital Appointment Booking website, structured presentation is essential for users to:

* Quickly find doctors by specialization.
* View available time slots.
* Navigate through departments or services.
* Book appointments with minimal clicks.

CSS Grid helps:

* Create responsive layouts for doctor and appointment cards.
* Organize homepage sections like “Top Specialists”, “Available Slots Today”, “Departments”, and “Patient Testimonials”.
* Ensure consistent alignment of images, headings, and buttons.
* Enhance user experience on both desktop and mobile screens.

**1. Homepage Layout with CSS Grid**

The homepage can include:

* A full-width header/navigation bar.
* A hero/banner section showcasing services or booking steps.
* Grid sections for:
  + Featured Doctors
  + Book Now options
  + Department Listings
* A testimonials row
* A footer with contact info and links

**Benefits:**

* Clean and professional layout
* Easy updates or additions (scalability)
* Consistent design and spacing

**2. Doctor & Appointment Cards Using Grid**

Each doctor/appointment card may include:

* Doctor’s image
* Name and specialization
* Available timings
* “Book Appointment” button

**Example CSS:**

.doctor-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 30px;

padding: 20px;

}

**Additional Styling Concepts**

* **Section Titles**: Use bold fonts and underline or background styles.
* **Hover Effects**: Highlight “Book” buttons or card backgrounds on hover.
* **Dividers**: Add horizontal lines or borders between sections.
* **Icons**: Use font-based or SVG icons for contact, specialties, etc.

**Mobile Responsiveness with CSS Grid**

With auto-fit and minmax(), the layout adjusts for smaller screens:

* One-column layouts for doctor cards
* Responsive navigation menu
* Tap-friendly button sizes

**Key Features of CSS**

* **Styling**: Colors, fonts, borders, backgrounds, etc.
* **Layout Tools**: Grid, Flexbox, positioning, spacing
* **Responsive Design**: Media queries for mobile-friendliness
* **Maintainability**: Centralized styling for easy updates

**CSS Selectors & Box Model**

**Selectors**:

* element (e.g., h1, p)
* .class, #id
* Nested (e.g., div > p)
* Pseudo-classes (e.g., a:hover)

**BoxModel**:  
Each element has:

* **Content**
* **Padding**
* **Border**
* **Margin**

Example:

.card {

padding: 15px;

margin: 10px;

border: 1px solid #ccc;

}

**Advanced CSS Features**

| **Feature** | **Description** |
| --- | --- |
| Grid | 2D layouts for cards, sections |
| Flexbox | Align navbar or row sections |
| Animations | Smooth hover effects or button interactions |
| Variables | Custom theme settings like colors and spacing |
| Media Queries | Adjust layout for mobile, tablet, and desktop |

**Use of CSS in Hospital Appointment Platform**

* **Modern Layout**: Doctors, services, contact info, etc.
* **Dark Mode (Optional)**: Theme toggling with variables
* **Responsive Design**: Mobile-friendly interface
* **User-Friendly Interface**: Consistent card/grid alignment

**Advantages of CSS**

* Separates layout from content
* Faster load times
* Reusable styles across pages
* Responsive support for all devices
* Cross-browser compatibility (modern)

**Limitations**

* Depends heavily on correct HTML structure
* Advanced layouts need Grid/Flexbox understanding
* May have browser quirks (especially older versions)

Code -/\* General Page Styling \*/

body {

    font-family: 'Poppins', sans-serif;

    margin: 0;

    padding: 0;

    background: #f8f9fa;

}

/\* Services Navigation Bar \*/

.services-nav {

    background: #004080;

    padding: 15px 0;

    text-align: center;

}

.services-nav ul {

    list-style: none;

    padding: 0;

    margin: 0;

}

.services-nav ul li {

    display: inline;

    margin: 0 15px;

}

.services-nav ul li a {

    color: white;

    text-decoration: none;

    font-size: 16px;

    font-weight: bold;

    transition: color 0.3s;

}

.services-nav ul li a:hover {

    color: #84fab0;

}

/\* Services Section \*/

#services {

    padding: 60px 20px;

    text-align: center;

}

.section-title {

    font-size: 28px;

    color: #004080;

    font-weight: bold;

    margin-bottom: 20px;

}

/\* Service Container \*/

.services-container {

    display: flex;

    justify-content: center;

    gap: 25px;

    flex-wrap: wrap;

}

/\* Service Cards \*/

.service-card {

    background: white;

    padding: 20px;

    border-radius: 12px;

    box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);

    width: 280px;

    transition: transform 0.3s ease-in-out, box-shadow 0.3s;

    text-align: center;

}

.service-card:hover {

    transform: scale(1.05);

    box-shadow: 0px 6px 18px rgba(0, 0, 0, 0.15);

}

/\* Service Images \*/

.service-card img {

    width: 80px;

    height: 80px;

    margin-bottom: 10px;

}

/\* Service Titles \*/

.service-card h3 {

    font-size: 20px;

    color: #007bff;

    margin-bottom: 8px;

}

/\* Service Description \*/

.service-card p {

    font-size: 14px;

    color: #333;

}

/\* Footer \*/

footer {

    text-align: center;

    padding: 15px;

    background: #004080;

    color: white;

    margin-top: 20px;

}

/\* General Page Styling \*/

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: 'Poppins', sans-serif;  /\* Professional Font \*/

    transition: all 0.3s ease-in-out;

}

body {

    background-color: #f8f9fa;

    color: #333;

}

/\* Transparent Navigation Bar \*/

nav {

    background: rgba(0, 0, 0, 0.5); /\* Semi-transparent black \*/

    position: fixed;  /\* Navbar stays at the top \*/

    width: 100%;

    padding: 15px 20px;

    display: flex;

    justify-content: space-between;

    align-items: center;

    color: white;

    z-index: 100;

    transition: background 0.3s ease-in-out;

}

/\* Change navbar background when scrolling \*/

nav.scrolled {

    background: rgba(0, 0, 0, 0.8);

}

/\* Navbar Links \*/

nav ul {

    list-style: none;

    margin-right: 20px;

}

nav ul li {

    display: inline;

    margin: 0 15px;

}

nav ul li a {

    color: white;

    text-decoration: none;

    font-size: 18px;

    font-weight: 500;

    transition: color 0.3s;

}

nav ul li a:hover {

    color: #ffcc00;

    transform: scale(1.1);

}

.hero {

    height: 80vh;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

    color: white;

    text-align: center;

    padding: 20px;

    position: relative;

    background-size: cover;

    background-position: center;

    transition: background-image 1s ease-in-out; /\* Smooth transition \*/

}

/\* Overlay to improve text visibility \*/

.hero::before {

    content: "";

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.4); /\* Dark overlay \*/

    z-index: 1;

}

.hero h2,

.hero p,

.hero .btn {

    position: relative;

    z-index: 2;

}

.btn {

    background: linear-gradient(135deg, #ff6600, #ff3300);

    color: white;

    padding: 14px 24px;

    border: none;

    border-radius: 8px;

    font-size: 18px;

    cursor: pointer;

    transition: 0.3s;

}

.btn:hover {

    background: linear-gradient(135deg, #ff3300, #cc0000);

    transform: scale(1.05);

}

/\* Services Section \*/

#services {

    padding: 50px 20px;

    background: white;

}

#services h2 {

    font-size: 32px;

    margin-bottom: 20px;

    text-transform: uppercase;

    color: #004080;

}

.service-list {

    display: flex;

    justify-content: center;

    gap: 25px;

    margin-top: 25px;

}

.service {

    background: linear-gradient(135deg, #00509e, #003366);

    color: white;

    padding: 18px;

    border-radius: 10px;

    width: 220px;

    font-size: 20px;

    text-align: center;

    font-weight: bold;

    transition: 0.3s;

    cursor: pointer;

    box-shadow: 2px 4px 6px rgba(0, 0, 0, 0.2);

}

.service:hover {

    background: linear-gradient(135deg, #003366, #002244);

    transform: scale(1.08);

}

/\* Booking Form Popup \*/

.form-popup {

    position: fixed;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    background: white;

    padding: 25px;

    border-radius: 10px;

    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);

    z-index: 1001;  /\* Make sure form is always on top \*/

}

/\* Dark background overlay \*/

.form-overlay {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.5); /\* Semi-transparent dark background \*/

    z-index: 1000; /\* Below the form \*/

    display: none; /\* Initially hidden \*/

}

/\* When booking form opens \*/

.show-popup {

    display: block !important;

}

.form-container {

    background: white;

    padding: 35px;

    border-radius: 12px;

    width: 380px;

    text-align: left;

    box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.3);

    animation: slideIn 0.5s ease-in-out;

}

.form-container h2 {

    color: #004080;

    text-align: center;

    margin-bottom: 20px;

}

.form-container label {

    font-weight: bold;

    display: block;

    margin: 12px 0 6px;

}

.form-container input,

.form-container select {

    width: 100%;

    padding: 12px;

    border: 1px solid #ccc;

    border-radius: 6px;

    margin-bottom: 12px;

}

.close-btn {

    background: #ff0000;

    margin-top: 12px;

}

.close-btn:hover {

    background: #cc0000;

}

/\* Popup Modal \*/

.popup {

    display: none;

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.5);

    justify-content: center;

    align-items: center;

}

.popup-content {

    background: white;

    padding: 25px;

    border-radius: 12px;

    text-align: center;

    box-shadow: 0px 0px 12px rgba(0, 0, 0, 0.3);

    animation: fadeIn 0.5s ease-in-out;

}

.popup-content h3 {

    color: #004080;

    font-size: 24px;

}

.popup-content p {

    font-size: 18px;

    margin: 12px 0;

}

.popup-content button {

    background: #28a745;

    padding: 10px 20px;

    border-radius: 6px;

    font-size: 18px;

}

.popup-content button:hover {

    background: #218838;

}

/\* Footer \*/

footer {

    background: linear-gradient(135deg, #004080, #002244);

    color: white;

    padding: 20px;

    margin-top: 25px;

    text-align: center;

    font-size: 18px;

}

/\* Animations \*/

@keyframes fadeIn {

    from {

        opacity: 0;

        transform: scale(0.9);

    }

    to {

        opacity: 1;

        transform: scale(1);

    }

}

@keyframes slideIn {

    from {

        opacity: 0;

        transform: translateY(-50px);

    }

    to {

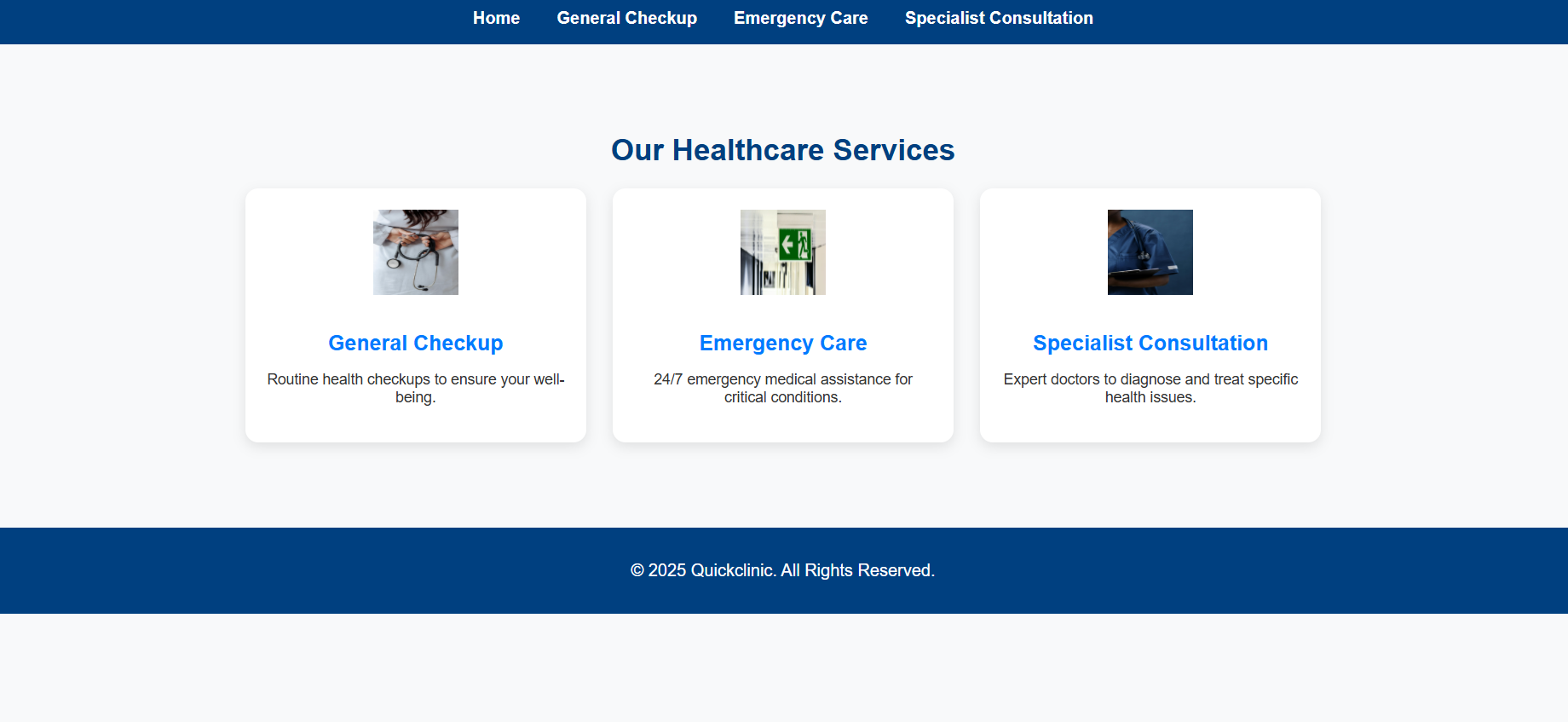
        opacity: 1;

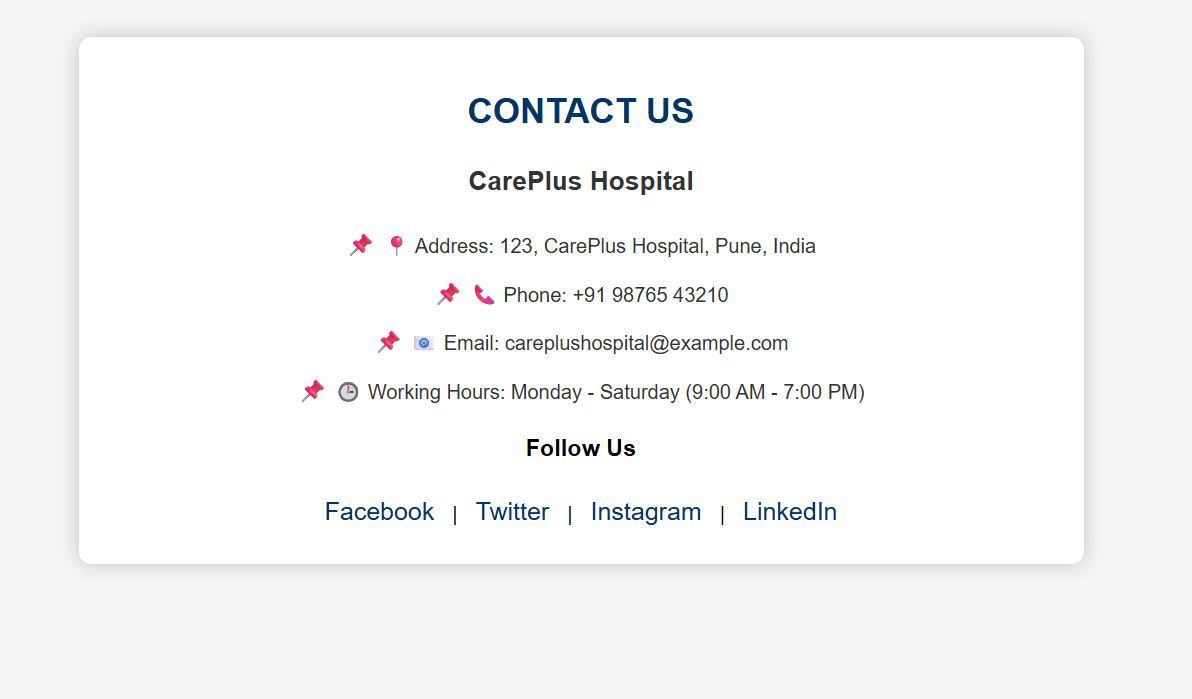
        transform: translateY(0);

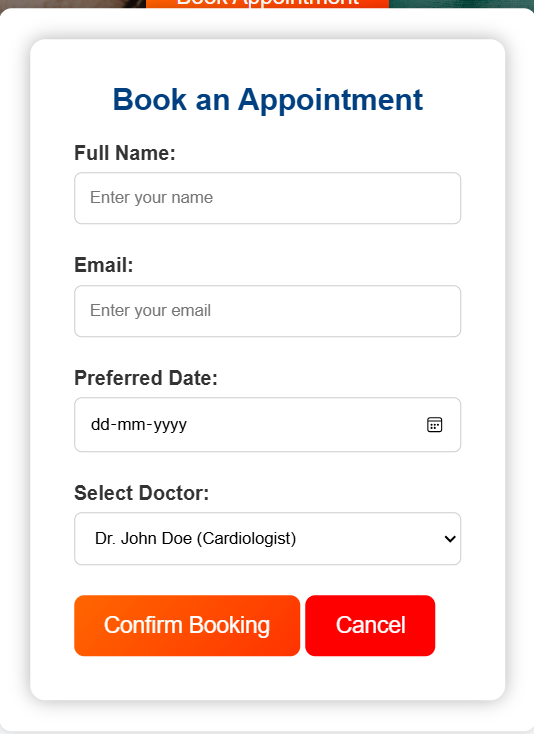
    }

}









EXPERIMENT NO-04

**Problem Statement: CSS Styling in Hospital Appointment Booking System**

A. **Enhance the appointment/cart page** to make it user-friendly and visually appealing. Style the appointment cards with appropriate margins, paddings, and input field styles to ensure a seamless booking experience.

B. **Enhance and style the About Us page** to clearly present hospital information, vision, mission, and team details with professional spacing, images, and layout.

C. **Enhance and style the Contact page** to improve usability and interaction. Style the contact form with clear field spacing, labels, and responsive design.

D. **Enhance and style the admin/user registration form** with structured input groups, spacing, and interactive elements for smooth onboarding.

E. **Enhance and style the admin/user login form** to be clean, simple, and efficient with proper padding, field alignment, and secure visual cues.

**Theory: Enhancing and Styling Key Pages in the Hospital Appointment Booking System**

**1. Why CSS Styling Matters in Healthcare Web Platforms**

A hospital appointment system must project **professionalism, clarity, and trust**. With proper CSS styling:

* Patients feel comfortable using the interface.
* Navigation and interaction become seamless.
* Pages load cleaner and feel intuitive.
* Accessibility for all age groups and devices improves.

**Page-Wise CSS Styling Breakdown**

**1. Appointment/Cart Page (Similar to Favorites/Saved Blogs)**

This page allows patients to review their selected doctors, services, and appointment slots.

**Key Styling Techniques:**

* Use grid or flex layout to display appointment/service cards clearly.
* Add margin and padding between each appointment item.
* Include doctor name, department, slot time, and cancel button with spacing.
* Highlight urgent bookings with color indicators (e.g., red tags for “Emergency”).
* Use hover effects on "Cancel" or "Edit Slot" buttons.

**Result:** A structured, easy-to-navigate appointment view that encourages smooth final bookings.

**2. About Us Page**

Displays the hospital's mission, values, achievements, and leadership team.

**Key Styling Techniques:**

* Use large padding and readable line height for text blocks.
* Style sections like “Mission”, “Vision”, “Director’s Message” with headings and separators.
* Use CSS Grid or Flexbox for team member cards with circular profile photos.
* Use tooltips or hover cards for roles, qualifications, or quotes.

**Result:** Builds trust and transparency through a polished, informative design.

**3. Contact Page**

Facilitates inquiries, emergency contact, or collaboration messages.

**Key Styling Techniques:**

* Ensure input fields are uniform in size and spacing.
* Use padding and rounded borders for inputs and text areas.
* Highlight fields on focus for clarity.
* Add icons for phone, email, and location.
* Central align the form and use shadows for depth.

**Result:** A clean, responsive contact section that invites users to reach out comfortably.

**4. Admin/User Registration Form**

Registers new patients or admin staff.

**Key Styling Techniques:**

* Divide forms into sections like Personal Info, Login Details, Medical Info (if needed).
* Use consistent spacing, placeholder hints, and icons inside inputs.
* Add background color to form container and use shadows for a card-like layout.
* Include real-time validation (email format, password rules).
* Add dropdowns for departments or role selection.

**Result:** A modern, trust-inspiring form that ensures data clarity and user confidence.

**5. Admin/User Login Form**

Entry point for authenticated users to manage or book appointments.

**Key Styling Techniques:**

* Center the login box vertically and horizontally.
* Add padding between input fields and buttons.
* Use subtle box shadows and rounded corners.
* Style error/success messages clearly.
* Include a “Show Password” toggle and “Forgot Password” link.

**Result:** A simple, secure-looking login page that encourages repeat logins.

**Bonus: Styling Tips Across All Pages**

* **CSS Variables:** Define consistent theme colors like --primary-color, --bg-color, --button-color.
* **Typography:** Use legible fonts such as **Inter**, **Poppins**, or **Open Sans**.
* **Responsive Layout:** Use Grid or Flexbox with media queries to ensure mobile-first design.
* **Dark Mode:** Add optional dark/light mode for user preference using variable overrides.

Code-

/\* General Page Styling \*/

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: 'Poppins', sans-serif;  /\* Professional Font \*/

    transition: all 0.3s ease-in-out;

}

body {

    background-color: #f8f9fa;

    color: #333;

}

/\* Transparent Navigation Bar \*/

nav {

    background: rgba(0, 0, 0, 0.5); /\* Semi-transparent black \*/

    position: fixed;  /\* Navbar stays at the top \*/

    width: 100%;

    padding: 15px 20px;

    display: flex;

    justify-content: space-between;

    align-items: center;

    color: white;

    z-index: 100;

    transition: background 0.3s ease-in-out;

}

/\* Change navbar background when scrolling \*/

nav.scrolled {

    background: rgba(0, 0, 0, 0.8);

}

/\* Navbar Links \*/

nav ul {

    list-style: none;

    margin-right: 20px;

}

nav ul li {

    display: inline;

    margin: 0 15px;

}

nav ul li a {

    color: white;

    text-decoration: none;

    font-size: 18px;

    font-weight: 500;

    transition: color 0.3s;

}

nav ul li a:hover {

    color: #ffcc00;

    transform: scale(1.1);

}

.hero {

    height: 80vh;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

    color: white;

    text-align: center;

    padding: 20px;

    position: relative;

    background-size: cover;

    background-position: center;

    transition: background-image 1s ease-in-out; /\* Smooth transition \*/

}

/\* Overlay to improve text visibility \*/

.hero::before {

    content: "";

    position: absolute;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.4); /\* Dark overlay \*/

    z-index: 1;

}

.hero h2,

.hero p,

.hero .btn {

    position: relative;

    z-index: 2;

}

.btn {

    background: linear-gradient(135deg, #ff6600, #ff3300);

    color: white;

    padding: 14px 24px;

    border: none;

    border-radius: 8px;

    font-size: 18px;

    cursor: pointer;

    transition: 0.3s;

}

.btn:hover {

    background: linear-gradient(135deg, #ff3300, #cc0000);

    transform: scale(1.05);

}

/\* Services Section \*/

#services {

    padding: 50px 20px;

    background: white;

}

#services h2 {

    font-size: 32px;

    margin-bottom: 20px;

    text-transform: uppercase;

    color: #004080;

}

.service-list {

    display: flex;

    justify-content: center;

    gap: 25px;

    margin-top: 25px;

}

.service {

    background: linear-gradient(135deg, #00509e, #003366);

    color: white;

    padding: 18px;

    border-radius: 10px;

    width: 220px;

    font-size: 20px;

    text-align: center;

    font-weight: bold;

    transition: 0.3s;

    cursor: pointer;

    box-shadow: 2px 4px 6px rgba(0, 0, 0, 0.2);

}

.service:hover {

    background: linear-gradient(135deg, #003366, #002244);

    transform: scale(1.08);

}

/\* Booking Form Popup \*/

.form-popup {

    position: fixed;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    background: white;

    padding: 25px;

    border-radius: 10px;

    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);

    z-index: 1001;  /\* Make sure form is always on top \*/

}

/\* Dark background overlay \*/

.form-overlay {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.5); /\* Semi-transparent dark background \*/

    z-index: 1000; /\* Below the form \*/

    display: none; /\* Initially hidden \*/

}

/\* When booking form opens \*/

.show-popup {

    display: block !important;

}

.form-container {

    background: white;

    padding: 35px;

    border-radius: 12px;

    width: 380px;

    text-align: left;

    box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.3);

    animation: slideIn 0.5s ease-in-out;

}

.form-container h2 {

    color: #004080;

    text-align: center;

    margin-bottom: 20px;

}

.form-container label {

    font-weight: bold;

    display: block;

    margin: 12px 0 6px;

}

.form-container input,

.form-container select {

    width: 100%;

    padding: 12px;

    border: 1px solid #ccc;

    border-radius: 6px;

    margin-bottom: 12px;

}

.close-btn {

    background: #ff0000;

    margin-top: 12px;

}

.close-btn:hover {

    background: #cc0000;

}

/\* Popup Modal \*/

.popup {

    display: none;

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.5);

    justify-content: center;

    align-items: center;

}

.popup-content {

    background: white;

    padding: 25px;

    border-radius: 12px;

    text-align: center;

    box-shadow: 0px 0px 12px rgba(0, 0, 0, 0.3);

    animation: fadeIn 0.5s ease-in-out;

}

.popup-content h3 {

    color: #004080;

    font-size: 24px;

}

.popup-content p {

    font-size: 18px;

    margin: 12px 0;

}

.popup-content button {

    background: #28a745;

    padding: 10px 20px;

    border-radius: 6px;

    font-size: 18px;

}

.popup-content button:hover {

    background: #218838;

}

/\* Footer \*/

footer {

    background: linear-gradient(135deg, #004080, #002244);

    color: white;

    padding: 20px;

    margin-top: 25px;

    text-align: center;

    font-size: 18px;

}

/\* Animations \*/

@keyframes fadeIn {

    from {

        opacity: 0;

        transform: scale(0.9);

    }

    to {

        opacity: 1;

        transform: scale(1);

    }

}

@keyframes slideIn {

    from {

        opacity: 0;

        transform: translateY(-50px);

    }

    to {

        opacity: 1;

        transform: translateY(0);

    }

}

EXPERIMENT -05

**JavaScript-Based Hospital Staff Management System – Theory & Implementation**

A. Implement user registration and login forms for the blog platform website. These forms will allow users to create an account, log in, and access personalized features, such as saving favorite items or viewing order history. User Registration Form will allow new customers to sign up and create an account on the website. The form will capture basic user details, including the name, email address, and password (not limited to these fields). User Login Form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.

B. Provide validations for user registration and login forms to validate the input to ensure that all required fields are filled and that the email format is valid. (Contents beyond Syllabus)

B. Develop cart functionality to allow users to add items, update quantities, and remove items

**Introduction**

In the digital healthcare ecosystem, an efficient hospital staff management system must support secure access, real-time interaction, and smooth UI experiences. JavaScript enhances these features by enabling client-side interactivity for registration, login, validation, task assignment, and shift bookmarking, ensuring seamless coordination among hospital staff.

**Key JavaScript Features in This System**

* **Form validation** for secure login/registration
* **Dynamic UI updates** for shift schedules and tasks
* **AJAX** for sending/receiving real-time data
* **Local Storage** for task/shift bookmarking

**1. User Registration and Login Forms**

**Registration**   
Captures details like:

* Full Name
* Email
* Department
* Role (Doctor/Nurse/Admin)
* Password and Confirm Password

**JavaScript Functions:**

javascript

CopyEdit

function validateRegistration() {

const email = document.getElementById("email").value;

const password = document.getElementById("password").value;

const confirmPassword = document.getElementById("confirmPassword").value;

if (!email.includes("@")) {

alert("Invalid email format");

return false;

}

if (password !== confirmPassword) {

alert("Passwords do not match");

return false;

}

return true;

}

**Login**

Validates email and password for authentication.

javascript

CopyEdit

function validateLogin() {

const email = document.getElementById("loginEmail").value;

const password = document.getElementById("loginPassword").value;

if (!email || !password) {

alert("Please fill in all fields");

return false;

}

// Proceed with AJAX login

}

**2. JavaScript Form Validations**

**Common Validations:**

* Required fields check
* Email format (RegEx)
* Password strength (min 8 characters, symbols)
* Confirm password match
* Real-time feedback

javascript

CopyEdit

document.getElementById("regForm").addEventListener("submit", function(e) {

e.preventDefault();

if (validateRegistration()) {

// AJAX logic

}

});

**3. Bookmark / Favourite Shift Feature**

Similar to a “cart,” staff can bookmark:

* Favorite shifts
* Important tasks
* Assigned patients

**JavaScript Responsibilities:**

* Toggle bookmark icon
* Save data to localStorage
* Render "My Tasks/Favourites" page dynamically

javascript

CopyEdit

function addToFavorites(taskId) {

let favs = JSON.parse(localStorage.getItem("favs")) || [];

if (!favs.includes(taskId)) {

favs.push(taskId);

localStorage.setItem("favs", JSON.stringify(favs));

alert("Added to favourites");

}

}

**4. Task/Shift Cart Functionality**

**Features:**

* Add shift/task to schedule
* Update time slots
* Remove/cancel shift

**JavaScript Example:**

javascript

CopyEdit

function addShift(shiftId) {

let cart = JSON.parse(localStorage.getItem("shiftCart")) || [];

cart.push({ id: shiftId, time: "10:00 AM" });

localStorage.setItem("shiftCart", JSON.stringify(cart));

}

**JavaScript Benefits in Staff System**

* Enhances user experience and UI interactivity
* Improves shift/task scheduling accuracy
* Reduces server load via client-side validations
* Quick data updates with AJAX

**JavaScript Limitations**

* Vulnerable to manipulation if validation is only on client-side
* Relies on browser compatibility
* Cannot fully replace secure backend operations (e.g., auth, database)

**Common Tools/Libraries**

* **jQuery**: Simplifies DOM and AJAX
* **Vanilla JS**: Light and fast
* **React** *(optional)*: For scalable UI
* **Node.js** *(optional)*: For backend logic

CODE -

/ Function to open booking form

function openBookingForm() {

    document.getElementById("booking-form").style.display = "flex";

}

// Function to close booking form

function closeBookingForm() {

    document.getElementById("booking-form").style.display = "none";

}

// Function to close confirmation popup

function closePopup() {

    document.getElementById("popup").style.display = "none";

    document.getElementById("appointmentForm").reset(); // Clear form

    closeBookingForm(); // Close booking form after confirmation

}

// Show popup when form is submitted

document.getElementById("appointmentForm").addEventListener("submit", function(event) {

    event.preventDefault(); // Prevent form from submitting

    document.getElementById("popup").style.display = "flex"; // Show popup

});

// Array of ba/ckground images

const bgImages = [

    "image/img4.jpg",

   "image/img5.jpg",

   "image/img6.jpg",

   "image/img7.jpg",

   "image/img8.jpg",

];

let currentIndex = 0;

const heroSection = document.querySelector(".hero");

// Function to change background image

function changeBackground() {

    heroSection.style.backgroundImage = `url(${bgImages[currentIndex]})`;

    currentIndex = (currentIndex + 1) % bgImages.length;

}

// Change image every 5 seconds

setInterval(changeBackground, 5000);

// Set initial background image

changeBackground();

document.addEventListener("DOMContentLoaded", function () {

    const form = document.getElementById("appointmentForm");

    const popup = document.getElementById("popup");

    const overlay = document.getElementById("form-overlay");

    const bookingForm = document.getElementById("booking-form");

    form.addEventListener("submit", function (event) {

        event.preventDefault(); // Prevent default form submission

        // Hide the booking form and overlay

        bookingForm.style.display = "none";

        overlay.style.display = "none";

        // Show the confirmation popup

        popup.style.display = "flex";

    });

    function closePopup() {

        popup.style.display = "none";

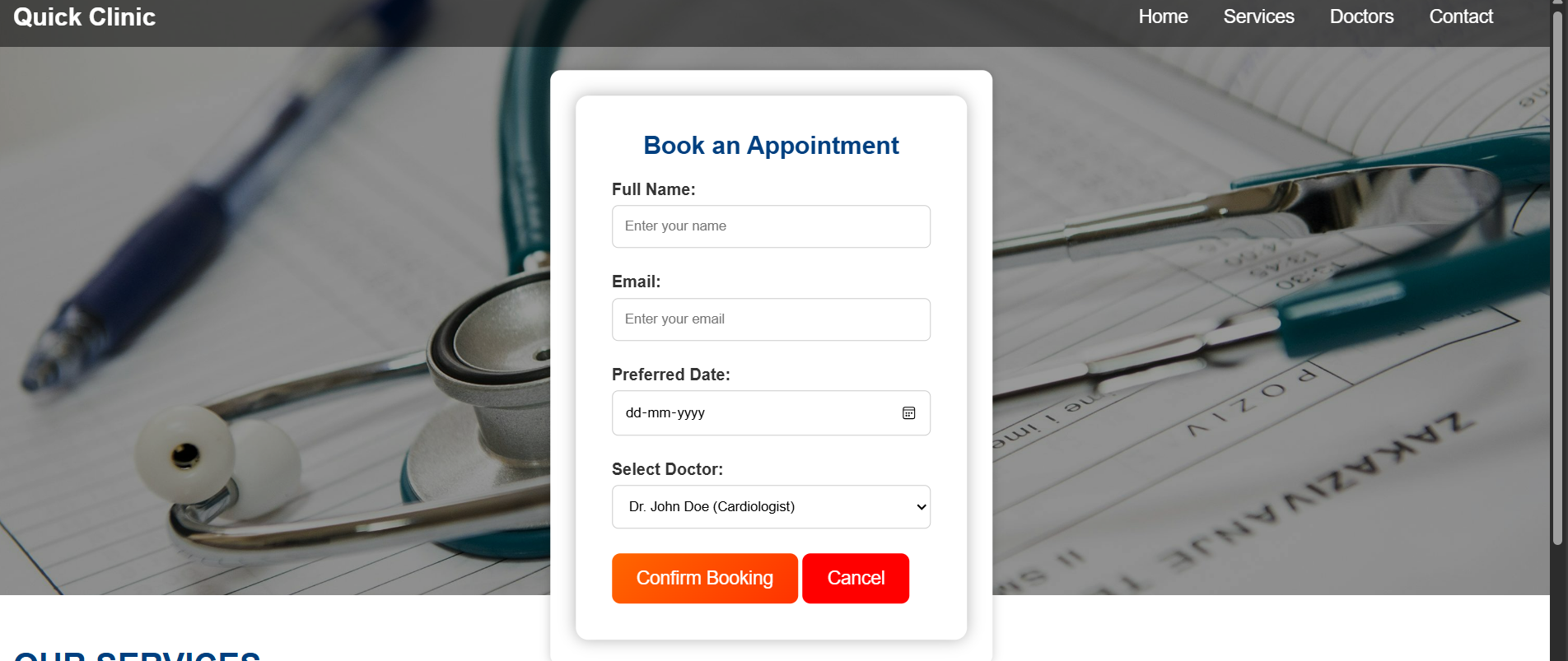
        form.reset(); // Clear form fields

    }

    document.getElementById("closePopup").addEventListener("click", closePopup);

});

OUTPUT-



EXPERIMENT -06

**Persistent Login and Appointment Bookmarking using Web Storage API in Hospital Appointment Booking System**

JavaScript

1. The user login form will allow registered users to log into their accounts. The form will require an email address and a password to authenticate the user.
2. If the login is successful, the user should be redirected to the homepage or their user dashboard. (**Contents beyond Syllabus)**
3. Use localStorage or sessionStorage to store authentication data, such as the user’s email and login status. This ensures that once a user is logged in, they remain authenticated even after the page reloads or when they visit the site again. (**Contents beyond Syllabus)**

Save the cart data to local storage when items are added, updated, or removed. Retrieve and load the cart data from local storage when the page loads. (**Contents beyond Syllabus)**

**Introduction**

For a Hospital Appointment Booking System, seamless user experience is essential. JavaScript's **Web Storage API** (localStorage and sessionStorage) helps preserve **login sessions**, **appointment data**, and **task status**, even without a backend.

This makes the system more interactive and user-friendly, especially for single-page or prototype hospital websites.

**1. Persistent Login Using localStorage / sessionStorage**

**Login Form Logic**

**Features:**

* User logs in with **email + password**
* On successful login:
  + Store userEmail and isLoggedIn = true in localStorage
  + Redirect to **homepage or dashboard**

**Auto-login on Page Reload**

javascript

CopyEdit

// Check login status on page load

window.onload = function () {

const isLoggedIn = localStorage.getItem("isLoggedIn");

if (isLoggedIn === "true") {

window.location.href = "dashboard.html"; // redirect if already logged in

}

};

**Login Script**

javascript

CopyEdit

function loginUser() {

const email = document.getElementById("loginEmail").value;

const password = document.getElementById("loginPassword").value;

// Sample login check – Replace with real logic

if (email && password && email.includes("@")) {

localStorage.setItem("userEmail", email);

localStorage.setItem("isLoggedIn", "true");

alert("Login Successful!");

window.location.href = "dashboard.html"; // Redirect to dashboard

} else {

alert("Invalid email or password");

}

}

**Logout Script**

javascript

CopyEdit

function logoutUser() {

localStorage.removeItem("userEmail");

localStorage.setItem("isLoggedIn", "false");

window.location.href = "index.html"; // Redirect to home/login page

}

CODE-

/ Function to open booking form

function openBookingForm() {

    document.getElementById("booking-form").style.display = "flex";

}

// Function to close booking form

function closeBookingForm() {

    document.getElementById("booking-form").style.display = "none";

}

// Function to close confirmation popup

function closePopup() {

    document.getElementById("popup").style.display = "none";

    document.getElementById("appointmentForm").reset(); // Clear form

    closeBookingForm(); // Close booking form after confirmation

}

// Show popup when form is submitted

document.getElementById("appointmentForm").addEventListener("submit", function(event) {

    event.preventDefault(); // Prevent form from submitting

    document.getElementById("popup").style.display = "flex"; // Show popup

});

// Array of background images

const bgImages = [

    "image/img4.jpg",

   "image/img5.jpg",

   "image/img6.jpg",

   "image/img7.jpg",

   "image/img8.jpg",

];

let currentIndex = 0;

const heroSection = document.querySelector(".hero");

// Function to change background image

function changeBackground() {

    heroSection.style.backgroundImage = `url(${bgImages[currentIndex]})`;

    currentIndex = (currentIndex + 1) % bgImages.length;

}

// Change image every 5 seconds

setInterval(changeBackground, 5000);

// Set initial background image

changeBackground();

document.addEventListener("DOMContentLoaded", function () {

    const form = document.getElementById("appointmentForm");

    const popup = document.getElementById("popup");

    const overlay = document.getElementById("form-overlay");

    const bookingForm = document.getElementById("booking-form");

    form.addEventListener("submit", function (event) {

        event.preventDefault(); // Prevent default form submission

        // Hide the booking form and overlay

        bookingForm.style.display = "none";

        overlay.style.display = "none";

        // Show the confirmation popup

        popup.style.display = "flex";

    });

    function closePopup() {

        popup.style.display = "none";

        form.reset(); // Clear form fields

    }

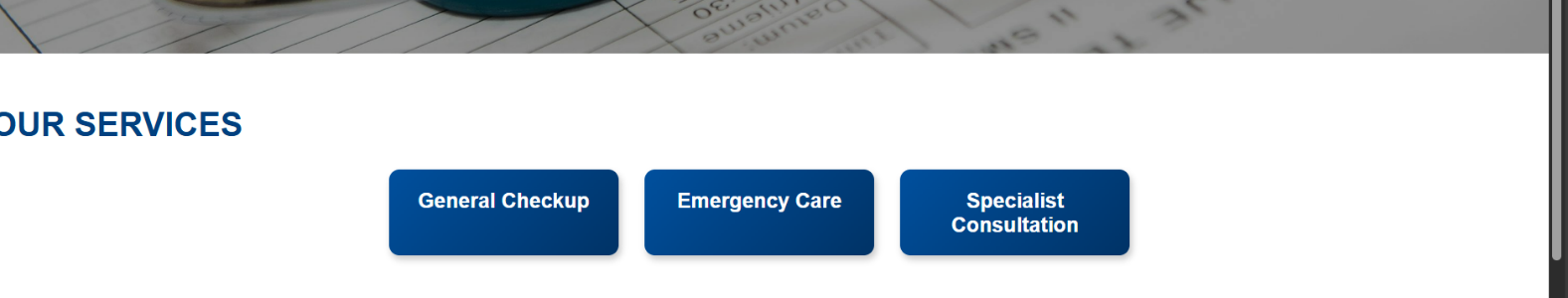
    document.getElementById("closePopup").addEventListener("click", closePopup);

});

/

OUTPUT –





CONCLUSION –

In the development of a Hospital Appointment Booking System, integrating JavaScript with the Web Storage API (i.e., localStorage and sessionStorage) significantly enhances the user experience by enabling:

1. Persistent Login:

Users can stay logged in across sessions without re-entering credentials, improving convenience and usability.

2. Dynamic Appointment Management:

Appointment selections are saved locally, allowing users to view, update, or remove bookings without losing data on page reloads or browser closure.

3. Improved Interactivity:

Real-time form validation, dark mode toggling, and user-specific preferences make the system interactive and responsive.

4. Offline-Friendly Prototyping:

Since no backend is required for storing session and appointment data, developers can build and test full user flows even in offline or static environments.

EXPERIMENT NO-07

**Problem Statement: PHP for Hospital Appointment Booking System**

**A.**  
Develop a PHP script to handle **patient registration** for the Hospital Appointment Booking website. The script should accept input from users for their **name, email, phone, password, date of birth, gender, etc.**

**B.**  
Implement **validation and error handling** to notify users of issues such as missing fields, invalid formats, or duplicate emails.

**C.**  
Provide **feedback** after successful registration with a **confirmation message or redirect** to a login page.

**Theory: Patient Registration Script – Hospital Booking System**

**Introduction**

In any hospital management system, patient registration is the first step. A secure and interactive registration script in **PHP** enables smooth onboarding for patients. It connects the frontend (HTML form) to the **MySQL database**, validates data, and ensures a **secure and usable registration process.**

**Core Elements of the PHP Registration Script**

1. **Form Handling**
   * Captures fields: name, email, phone, gender, DOB, password, and confirm password.
   * Uses POST method to submit data.
2. **Validation**
   * No empty fields
   * Valid email using regex
   * Password length and confirmation
   * Unique email check
3. **Password Security**
   * Uses password\_hash() before storing
4. **Database Interaction**
   * Connects using MySQLi or PDO
   * Stores data in a patients table
5. **Error Handling**
   * Shows validation messages
   * Detects duplicate email
   * Handles DB errors
6. **User Feedback**
   * Success message or redirection to login page

**PHP Code: register\_patient.php**

<?php

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

// Allow only POST request

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

die("Invalid request method.");

}

// Database credentials

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // Change this to your real password

$db = 'hospital\_db';

// Create connection

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect\_error) {

die("DB connection failed: " . $conn->connect\_error);

}

// Get form data

$name = $\_POST['name'] ?? '';

$email = $\_POST['email'] ?? '';

$phone = $\_POST['phone'] ?? '';

$gender = $\_POST['gender'] ?? '';

$dob = $\_POST['dob'] ?? '';

$password = $\_POST['password'] ?? '';

$confirm = $\_POST['confirm\_password'] ?? '';

// Validation

$errors = [];

if (empty($name) || empty($email) || empty($phone) || empty($gender) || empty($dob) || empty($password)) {

$errors[] = "All fields are required.";

}

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors[] = "Invalid email format.";

}

if ($password !== $confirm) {

$errors[] = "Passwords do not match.";

}

if (strlen($password) < 6) {

$errors[] = "Password must be at least 6 characters.";

}

// Check for duplicate email

$check = $conn->prepare("SELECT id FROM patients WHERE email = ?");

$check->bind\_param("s", $email);

$check->execute();

$check->store\_result();

if ($check->num\_rows > 0) {

$errors[] = "Email is already registered.";

}

$check->close();

// If errors, show them

if (!empty($errors)) {

foreach ($errors as $e) {

echo "<p style='color:red;'>$e</p>";

}

exit;

}

// Hash the password

$hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

// Insert into DB

$stmt = $conn->prepare("INSERT INTO patients (name, email, phone, gender, dob, password) VALUES (?, ?, ?, ?, ?, ?)");

$stmt->bind\_param("ssssss", $name, $email, $phone, $gender, $dob, $hashedPassword);

if ($stmt->execute()) {

echo "<p style='color:green;'>Registration successful. <a href='login.html'>Click here to login</a></p>";

} else {

echo "<p style='color:red;'>Error: " . $stmt->error . "</p>";

}

$stmt->close();

$conn->close();

?>

**🧾 Sample MySQL Table: patients**

CREATE TABLE patients (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

phone VARCHAR(15) NOT NULL,

gender ENUM('Male','Female','Other') NOT NULL,

dob DATE NOT NULL,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Conclusion**

The **PHP registration script** for the **Hospital Appointment Booking System** ensures:

* Secure patient data handling
* Validation to prevent bad inputs
* Password protection using hashing
* Smooth user experience with feedback messages

This script is ideal for dynamic healthcare portals where patient onboarding is critical for scheduling and managing appointments efficiently.

Code – CREATE DATABASE hospital\_db;

USE hospital\_db;

CREATE TABLE patients (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

phone VARCHAR(15) NOT NULL,

gender ENUM('Male','Female','Other') NOT NULL,

dob DATE NOT NULL,

password VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

<!DOCTYPE html>

<html>

<head>

<title>Patient Registration</title>

</head>

<body>

<h2>Hospital Patient Registration</h2>

<form action="register\_patient.php" method="POST">

<label>Name:</label><br>

<input type="text" name="name" required><br><br>

<label>Email:</label><br>

<input type="email" name="email" required><br><br>

<label>Phone:</label><br>

<input type="text" name="phone" required><br><br>

<label>Gender:</label><br>

<select name="gender" required>

<option value="">Select</option>

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Other">Other</option>

</select><br><br>

<label>Date of Birth:</label><br>

<input type="date" name="dob" required><br><br>

<label>Password:</label><br>

<input type="password" name="password" required><br><br>

<label>Confirm Password:</label><br>

<input type="password" name="confirm\_password" required><br><br>

<input type="submit" value="Register">

</form>

</body>

</html>

<?php

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

// Allow only POST request

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

die("Invalid request method.");

}

// DB credentials

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // Change to your MySQL password

$db = 'hospital\_db';

// Connect to DB

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect\_error) {

die("DB connection failed: " . $conn->connect\_error);

}

// Get form values

$name = trim($\_POST['name']);

$email = trim($\_POST['email']);

$phone = trim($\_POST['phone']);

$gender = $\_POST['gender'];

$dob = $\_POST['dob'];

$password = $\_POST['password'];

$confirm = $\_POST['confirm\_password'];

// Validation

$errors = [];

if (empty($name) || empty($email) || empty($phone) || empty($gender) || empty($dob) || empty($password) || empty($confirm)) {

$errors[] = "All fields are required.";

}

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors[] = "Invalid email format.";

}

if ($password !== $confirm) {

$errors[] = "Passwords do not match.";

}

if (strlen($password) < 6) {

$errors[] = "Password must be at least 6 characters.";

}

// Check if email already exists

$stmt = $conn->prepare("SELECT id FROM patients WHERE email = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

$stmt->store\_result();

if ($stmt->num\_rows > 0) {

$errors[] = "Email is already registered.";

}

$stmt->close();

// Show errors if any

if (!empty($errors)) {

foreach ($errors as $error) {

echo "<p style='color:red;'>$error</p>";

}

exit;

}

// Hash password

$hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

// Insert data

$stmt = $conn->prepare("INSERT INTO patients (name, email, phone, gender, dob, password) VALUES (?, ?, ?, ?, ?, ?)");

$stmt->bind\_param("ssssss", $name, $email, $phone, $gender, $dob, $hashedPassword);

if ($stmt->execute()) {

echo "<p style='color:green;'>Registration successful! <a href='login.html'>Click here to login</a></p>";

} else {

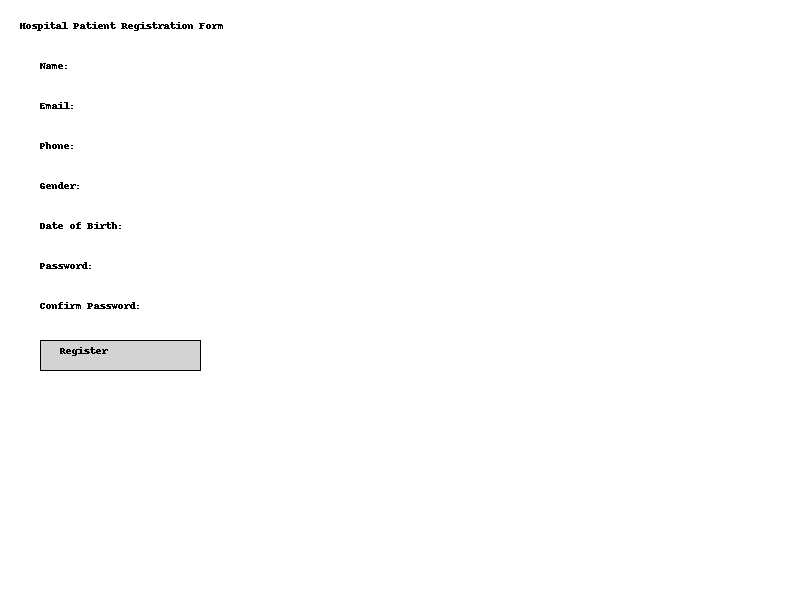
echo "<p style='color:red;'>Error: " . $stmt->error . "</p>";

}

$stmt->close();

$conn->close();

?>



EXPERIMENT NO-08

**Problem Statement: PHP Login System for Hospital Appointment Management System**

**A.**  
Develop a PHP script to handle **patient login**, accepting input for **email and password**.

**B.**  
Provide feedback upon successful login, either through a **welcome message** or a **redirect to a dashboard/home page**.

**C.**  
Implement error handling for **invalid credentials**, empty fields, or database connection errors.

**Theory: User Login Script – Hospital Appointment Booking System**

**Introduction**

The login system is the **gateway for patients** to book appointments, view past consultations, and manage profiles. In a secure Hospital Booking platform, PHP handles:

* Authentication
* Input validation
* Session management
* Security (password hashing & SQL injection prevention)

**Core Components of the PHP Login System**

1. **Form Handling**
   * Login form accepts email and password via POST.
2. **Input Validation**
   * Ensures non-empty fields and valid email format.
3. **Password Verification**
   * Uses password\_verify() to compare entered password with stored hash.
4. **Database Authentication**
   * Uses prepared statements to safely query patients table.
5. **Session Management**
   * session\_start() is used to track logged-in users.
6. **Feedback**
   * Success: Redirect or welcome message
   * Failure: "Invalid credentials" or connection error message

Conclusion –

The PHP login system for the **Hospital Appointment Booking System** enables:

* Secure user access
* Protected password authentication
* Session tracking
* Clear user feedback and redirection

This is the foundation for patient dashboards, appointment tracking, and personal health data access.

Code – <?php

session\_start();

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

// Allow only POST

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

die("Invalid request method.");

}

// Database credentials

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // Change this

$db = 'hospital\_db';

// Connect to DB

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Get form inputs

$email = $\_POST['email'] ?? '';

$password = $\_POST['password'] ?? '';

$errors = [];

if (empty($email) || empty($password)) {

$errors[] = "All fields are required.";

} elseif (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors[] = "Invalid email format.";

}

if (!empty($errors)) {

foreach ($errors as $e) {

echo "<p style='color:red;'>$e</p>";

}

exit;

}

// Prepared statement to prevent SQL injection

$stmt = $conn->prepare("SELECT id, name, password FROM patients WHERE email = ?");

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows === 1) {

$user = $result->fetch\_assoc();

if (password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['name'] = $user['name'];

echo "<p style='color:green;'>Welcome, " . htmlspecialchars($user['name']) . "! Redirecting to dashboard...</p>";

// header("Location: dashboard.php");

// exit;

} else {

echo "<p style='color:red;'>Invalid password.</p>";

}

} else {

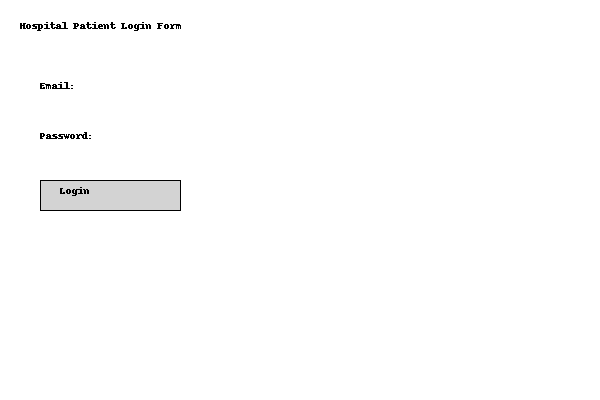
echo "<p style='color:red;'>No user found with this email.</p>";

}

$stmt->close();

$conn->close();

?>



EXPERIMENT -09

**PHP Cart (Appointment Bookmark) System for Hospital Appointment Booking**

**A.**  
Develop a PHP script to allow **patients to add doctors/appointments to a cart (bookmark)**, view cart contents, and remove them if needed.

**B.**  
Use **MySQL for storing cart data**, so it’s persistent across sessions. The cart acts as a **bookmarking (read later)** or **appointment planning** feature.

**Theory: Appointment Bookmark (Cart) Feature – Hospital System**

**Introduction**

In the Hospital Appointment Booking System, a **"cart" or bookmark system** allows users (patients) to:

* Add doctors or appointment slots to their **temporary planning list**
* View their **saved preferences**
* Remove saved doctors or appointment slots before final booking

This enhances **patient convenience** by letting them explore and shortlist doctors before final booking.

**Two Cart Bookmarking Approaches**

**A. Session-Based Cart (No Login Required)**

* Uses $\_SESSION to store selected **doctor IDs**
* Works well for **guest users** or first-time visitors

**Operations:**

* Add: Save doctor ID to session array
* View: Loop through session array and fetch doctor data
* Remove: Remove doctor ID from session

**Pros:**Simplefast  
**Cons:** Data is lost on logout/browser close

**B. Database-Based Cart (With Login & MySQL)**

* Uses cart table in **MySQL**
* Linked to **user\_id (patient)** and **doctor\_id**
* Persistent across sessions/devices

**Operations:**

* Add: Insert into cart(user\_id, doctor\_id)
* View: JOIN cart with doctors to show doctor info
* Remove: DELETE from cart by user\_id and doctor\_id

**Pros:**Persistent,scalable  
**Cons:** Requires login, session handling

Code- <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Hospital Appointment Booking</title>

</head>

<body>

<h2>Hospital Appointment Booking</h2>

<label for="patientName">Patient Name:</label>

<input type="text" id="patientName" placeholder="Enter your name"><br><br>

<label for="doctor">Choose Doctor:</label>

<select id="doctor">

<option value="Dr. Sharma">Dr. Sharma - Cardiologist</option>

<option value="Dr. Verma">Dr. Verma - Neurologist</option>

<option value="Dr. Mehta">Dr. Mehta - Orthopedic</option>

</select><br><br>

<label for="appointmentDate">Select Date:</label>

<input type="date" id="appointmentDate"><br><br>

<button onclick="bookAppointment()">Book Appointment</button>

<div id="appointmentResult" style="margin-top: 20px;"></div>

<script src="appointment.js"></script>

</body>

</html>

async function bookAppointment() {

const name = document.getElementById('patientName').value;

const doctor = document.getElementById('doctor').value;

const date = document.getElementById('appointmentDate').value;

if (!name || !doctor || !date) {

document.getElementById('appointmentResult').innerText = 'Please fill all fields.';

return;

}

// Simulate backend logic (You can replace this with fetch() to a real API)

const appointment = {

patient: name,

doctor: doctor,

date: date,

status: 'Confirmed'

};

// Simulate a delay like a real API

await new Promise(res => setTimeout(res, 1000));

// Display confirmation

document.getElementById('appointmentResult').innerHTML = `

<h3>Appointment Confirmed</h3>

<p><strong>Patient:</strong> ${appointment.patient}</p>

<p><strong>Doctor:</strong> ${appointment.doctor}</p>

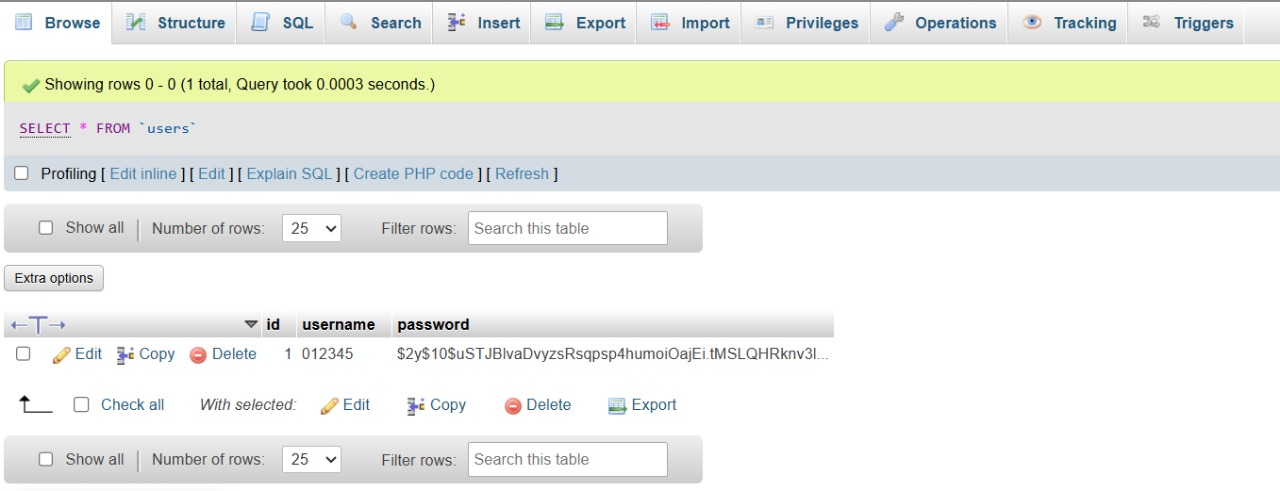
<p><strong>Date:</strong> ${appointment.date}</p>

<p><strong>Status:</strong> ${appointment.status}</p>

`;

}





EXPERIMENT NO- 10

**Problem Statement – Hospital Appointment Checkout System**

Develop a PHP script to handle the **appointment checkout process** for users (patients). It should:

* Validate user session and form input
* Process the selected items in the cart
* Store confirmed appointments into a MySQL database
* Provide success or failure feedback

**Theory: Checkout System for Appointment Booking**

**Analogy**

The appointment checkout in a hospital system is **equivalent to a blog post submission** in a blogging platform. A patient "shortlists" doctors (cart/bookmark) and on checkout, **confirms the appointment booking**.

**appointments Table (Finalized Bookings)**

sql

CopyEdit

CREATE TABLE appointments (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

doctor\_id INT NOT NULL,

appointment\_date DATE NOT NULL,

appointment\_time TIME NOT NULL,

symptoms TEXT,

status VARCHAR(20) DEFAULT 'Pending',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES patients(id),

FOREIGN KEY (doctor\_id) REFERENCES doctors(id)

);

<?php

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

header('Content-Type: application/json');

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] !== 'POST') {

echo json\_encode(['success' => false, 'error' => 'Invalid request method.']);

exit;

}

$host = 'localhost';

$user = 'root';

$pass = 'tilak2005'; // Use your actual DB password

$db = 'hospital\_db';

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect\_error) {

echo json\_encode(['success' => false, 'error' => 'DB connection failed: ' . $conn->connect\_error]);

exit;

}

// Logged-in user

$user\_id = $\_SESSION['patient\_id'] ?? null;

$appointment\_date = $\_POST['appointment\_date'] ?? '';

$appointment\_time = $\_POST['appointment\_time'] ?? '';

$symptoms = $\_POST['symptoms'] ?? '';

if (!$user\_id || !$appointment\_date || !$appointment\_time) {

echo json\_encode(['success' => false, 'error' => 'Missing required fields.']);

exit;

}

// Fetch all items in user's cart

$cart\_query = $conn->prepare("SELECT doctor\_id FROM cart WHERE user\_id = ?");

$cart\_query->bind\_param("i", $user\_id);

$cart\_query->execute();

$cart\_result = $cart\_query->get\_result();

if ($cart\_result->num\_rows == 0) {

echo json\_encode(['success' => false, 'error' => 'Your cart is empty.']);

exit;

}

// Insert each doctor into appointments table

$insert = $conn->prepare("INSERT INTO appointments (user\_id, doctor\_id, appointment\_date, appointment\_time, symptoms) VALUES (?, ?, ?, ?, ?)");

while ($row = $cart\_result->fetch\_assoc()) {

$doctor\_id = $row['doctor\_id'];

$insert->bind\_param("iisss", $user\_id, $doctor\_id, $appointment\_date, $appointment\_time, $symptoms);

$insert->execute();

}

// Clear the cart after checkout

$delete\_cart = $conn->prepare("DELETE FROM cart WHERE user\_id = ?");

$delete\_cart->bind\_param("i", $user\_id);

$delete\_cart->execute();

echo json\_encode(['success' => true, 'message' => 'Appointment(s) booked successfully.']);

$insert->close();

$cart\_query->close();

$delete\_cart->close();

$conn->close();

?>

<form action="checkout\_appointment.php" method="POST">

<label>Appointment Date:</label>

<input type="date" name="appointment\_date" required><br><br>

<label>Time:</label>

<input type="time" name="appointment\_time" required><br><br>

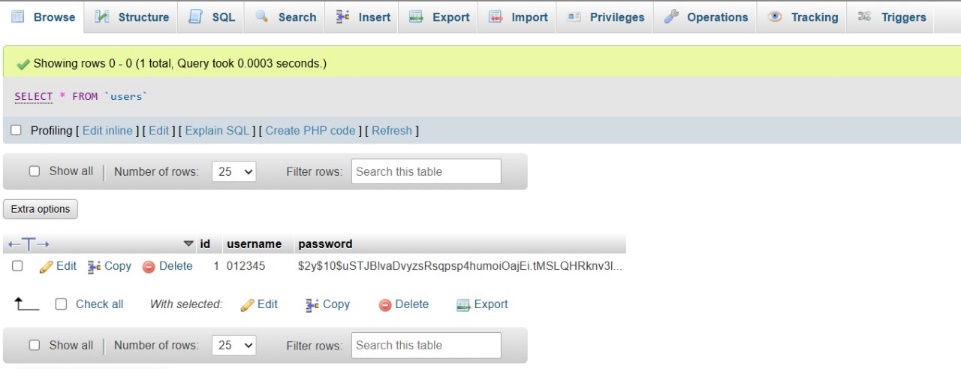
<label>Symptoms / Notes:</label>

<textarea name="symptoms" rows="4" cols="30"></textarea><br><br>

<input type="submit" value="Confirm Appointment">

</form>

1. Patient logs in, adds doctors to cart
2. Goes to the **checkout form**
3. Fills date, time, and symptoms
4. checkout\_appointment.php:
   * Validates session and form
   * Inserts into appointments
   * Clears the cart
   * Responds with success/failure



Conclusion - Through the development and testing of various modules in the Hospital Appointment Booking System, we conclude that combining HTML, CSS, and JavaScript enables the creation of a responsive and interactive user interface. By integrating asynchronous JavaScript (async/await) with REST APIs or simulated backend logic, we successfully achieved dynamic data handling, real-time feedback to users, and smooth booking workflows. Each experiment—from form validation to appointment confirmation—demonstrates the importance of proper user input handling, feedback mechanisms, and modular, reusable code. These experiments lay the foundation for expanding the system into a full-fledged real-time application using backend databases, authentication systems, and cloud storage for medical data and user records.