

**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

Hospital Management System

**A CAPSTONE PROJECT REPORT**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**Computer Science and Engineering**

**Submitted by**

**K. Ravisankar Guptha (192210038)**

**M. Manoj (192211728)**

**M. Mohammad Hussain (192211654)**

**Under the Supervision of**

**Dr. Vivek Balaji**

**JUNE 2024**

**DECLARATION**

We, **Ravisankar Guptha. k, M. Manoj, M. Mohammad Hussain** students of **Bachelor of Engineering in CSE**, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **Hospital management system** is the outcome of our bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

Ravisankar Guptha. k (192210038)

M. Manoj (192211728)

M. Mohammad Hussain (192211654)

Date:

Place:

**CERTIFICATE**

This is to certify that the project entitled **“Hospital Management System”** wassubmitted by **K. Ravisankar Guptha (192210038),** **M. Manoj (192211728), M. Mohammad Hussain (192211654)** has been carried out under my supervision. The project has been submitted per the requirements in the current B. Tech Computer Science semester.

Teacher-in-charge

Dr. Vivek Balaji

**Table of Contents**

|  |  |
| --- | --- |
| **S.NO** | **TOPICS** |
| 1 | **Abstract** |
| 2 | **Introduction** |
| 3 | **Literature survey** |
| 4 | **Project and Problem Description**  About our project and Program to build a website for hospitalmanagmentsystem |
| 5 | **Tool Description**  User interface  Features |
| 6 | **Operations**  Store the details of the patient.  Store the details of the appointment.  Store the details of the doctor.  Store the details of the contact information.  Store the details for login and password. |
| 7 | **Approach / Module Description / Functionalities**  The idea is to form individual functions for every operation. All the functions are unified together to form an application. |
| 8 | **Presentation**  Coding and output screenshots |
| 9 | **Architecture and Flow Diagram**  Represent flow diagram of hospital management system |
| 10 | **Conclusion & References**  Future Enhancement |

**ABSTRACT**

This project presents the design and implementation of an integrated Hospital management system aimed at streamlining hospital operations, improving patient care, and enhancing data management. The front end, crafted with HTML, CSS, and JavaScript, provides an intuitive and responsive user experience, while the backend, developed using Node.js and Express.js, ensures secure and efficient data handling. Key features include patient registration, appointment scheduling, medical records management, billing, and reporting, all supported by a MySQL database for reliable data storage and integrity. Role-based access control protects sensitive information, ensuring privacy and security. Emphasizing scalability, security, and usability, this system adapts to hospitals of various sizes, reducing administrative burdens and enhancing operational efficiency, ultimately improving patient satisfaction. This comprehensive solution addresses the challenges of modern healthcare institutions, demonstrating significant advancements in healthcare technology.

**INTRODUCTION**

In today's rapidly evolving healthcare landscape, the need for efficient and integrated hospital management systems has never been more critical. Traditional paper-based systems and fragmented digital solutions often lead to inefficiencies, errors, and delays, adversely affecting patient care and hospital operations. To address these challenges, this project focuses on developing an integrated frontend and backend web-based hospital management system that enhances operational efficiency, improves patient care, and streamlines administrative processes.

The proposed system aims to bridge the gap between different hospital departments, providing a seamless flow of information and facilitating better communication among healthcare providers, patients, and administrative staff. By leveraging modern web technologies, the system offers a user-friendly frontend interface designed with HTML, CSS, and JavaScript, ensuring a responsive and intuitive experience for all users, including patients, doctors, and administrative personnel.

The backend system utilizes robust server-side technologies such as Node.js and Express.js to handle data processing, storage, and retrieval. The integration of a relational database like MySQL guarantees reliable and secure data management. Key functionalities of the system include patient registration, appointment scheduling, medical records management, billing, and comprehensive reporting.

Security and privacy are paramount in healthcare applications. Therefore, the system incorporates role-based access control to ensure that sensitive information is accessible only to authorized personnel. This feature not only protects patient data but also complies with healthcare regulations and standards.

**Literature Survey**

Hospital Management Systems (HMS) are crucial for modern healthcare, facilitating the management of administrative, financial, and clinical operations within hospitals. These systems have evolved from basic administrative tools to sophisticated platforms integrating Electronic Health Records (EHR), telemedicine, and mobile health applications. Key components of HMS include patient management, clinical management, financial management, administrative management, and decision support systems. The benefits of implementing HMS are numerous, including improved efficiency, enhanced patient care, cost reduction, data accuracy, security, and regulatory compliance. However, challenges such as high costs, the need for extensive training, resistance to change, data privacy concerns, and interoperability issues persist. Current trends in HMS include cloud-based solutions, AI and machine learning, telemedicine integration, mobile health applications, and blockchain technology, pointing towards a future of personalized medicine, IoT integration, enhanced interoperability, patient-centric care, and advanced analytics. Despite the challenges, the continuous advancement in technology is driving HMS towards more integrated, patient-focused, and data-driven healthcare solutions.

**Project Description**

This project involves the design and development of an integrated frontend and backend web-based hospital management system aimed at revolutionizing how hospitals manage their operations, patient care, and administrative tasks. The system is intended to provide a cohesive and efficient solution to the often fragmented and error-prone traditional methods of hospital management.

**Key Components:**

1. **Frontend Interface:**

- Developed using HTML, CSS, and JavaScript to ensure a responsive, intuitive, and user-friendly experience.

- The interface caters to various users, including patients, doctors, and administrative staff, each with tailored functionalities.

- Patients can easily register, schedule appointments, view medical records, and manage their profiles.

- Doctors can access patient records, update treatment plans, and manage their schedules.

- Administrative staff can handle billing, generate reports, and oversee overall hospital operations.

2. **Backend System:**

- Built using Node.js and Express.js for efficient server-side operations.

- Handles data processing, storage, and retrieval, ensuring smooth and secure interactions between the front end and the database.

- Incorporates a MySQL database for reliable and structured data storage.

- Ensures data integrity and quick access to necessary information through optimized queries and indexing.

3. **Key Functionalities:**

- **Patient Registration**: Streamlines the process of adding new patients to the system, capturing all necessary personal and medical information.

- **Appointment Scheduling:** Allows patients to book appointments with their preferred doctors, view available slots, and receive confirmations.

- **Medical Records Management:** Maintains detailed electronic health records, including patient history, treatments, medications, and test results.

- **Billing and Invoicing**: Automates the billing process, generating accurate invoices for services rendered, and managing payment records.

- **Reporting and Analytics**: Provides comprehensive reports on various aspects of hospital operations, such as patient inflow, doctor performance, and financial summaries.

4. **Security Features:**

- Implements role-based access control to ensure that sensitive data is accessible only to authorized personnel.

- Encrypts sensitive data to protect patient privacy and comply with healthcare regulations.

- Regular audits and logs to monitor system access and detect any unauthorized activities.

5. **Scalability and Usability:**

- Designed to accommodate the needs of hospitals of varying sizes, from small clinics to large medical institutions.

- Emphasis on a modular architecture that allows easy addition of new features and functionalities as needed.

- User feedback incorporated to continuously improve and update the system for better performance and user experience.

This integrated hospital management system is designed to significantly enhance the efficiency of hospital operations, reduce administrative workloads, and improve the overall quality of patient care. By providing a centralized and cohesive platform for managing all aspects of hospital operations, this project aims to contribute to better healthcare delivery and operational excellence in the healthcare industry.

**Problem Description**

In the modern healthcare landscape, hospitals face numerous challenges that impede their ability to deliver efficient and effective patient care. Traditional methods of managing hospital operations often involve fragmented systems, manual processes, and paper-based records, leading to a host of problems including inefficiencies, errors, and delays. These issues can compromise patient care, strain resources, and hinder overall hospital performance. The primary problems addressed by this project include:

1. **Fragmented Systems:**

- Many hospitals rely on multiple disjointed systems for different functions such as patient registration, appointment scheduling, billing, and medical records management. This fragmentation results in data silos, making it difficult to access and share information across departments.

2. **Manual Processes and Errors**:

- Paper-based records and manual processes are prone to human errors, misplacements, and inefficiencies. Manual data entry can lead to inaccuracies in patient information, billing errors, and delays in administrative tasks, which can negatively impact patient care and hospital operations.

3. **Inefficient Communication:**

- Poor communication between different departments and healthcare providers can result in delayed treatments, misinformed decisions, and a lack of coordination in patient care. A lack of centralized information can hinder the ability to provide timely and accurate medical interventions.

4. **Data Security and Privacy Concerns:**

- The handling of sensitive patient data poses significant security and privacy challenges. Inadequate security measures can lead to unauthorized access, data breaches, and non-compliance with healthcare regulations, risking patient confidentiality and legal consequences.

5. **Resource Management:**

- Inefficient management of hospital resources, such as staff, equipment, and facilities, can lead to overbooking, underutilization, and poor patient experiences. Hospitals need a system that can optimize resource allocation and utilization.

6. **Scalability Issues:**

- Many existing hospital management systems are not scalable, making it difficult for hospitals to expand their operations or integrate new functionalities as needed. This lack of scalability can limit the growth and adaptability of healthcare institutions.

**Proposed Solution:**

This project proposes the development of an integrated frontend and backend web-based hospital management system to address these challenges comprehensively:

1. **Unified System:**

- The system integrates all hospital functions into a single platform, eliminating data silos and enabling seamless information sharing across departments.

2. **Automated Processes:**

- By digitizing and automating processes such as patient registration, appointment scheduling, billing, and medical records management, the system reduces the risk of errors and increases efficiency.

**3. Enhanced Communication:**

- A centralized database ensures that all healthcare providers have access to up-to-date patient information, improving communication and coordination in patient care.

**4. Robust Security:**

- Implementing role-based access control, data encryption, and regular security audits ensures that patient data is protected against unauthorized access and breaches, maintaining privacy and regulatory compliance.

**5. Optimized Resource Management:**

- The system provides tools for effective resource allocation and utilization, improving scheduling and reducing bottlenecks in hospital operations.

**6. Scalable Architecture:**

- Designed with scalability in mind, the system can grow with the hospital's needs, allowing for easy integration of new features and expansion of services.

By addressing these core problems, the integrated hospital management system aims to enhance operational efficiency, improve patient care, and provide a robust and secure solution for modern healthcare institutions.

**Tool Description**

The integrated frontend and backend web-based hospital management system utilizes a combination of modern web technologies and tools to create a robust, efficient, and user-friendly platform. Below is a detailed description of the tools used in the development of the system:

**Frontend Tools**

**1. HTML5:**

- Purpose: Structures the web pages and forms the backbone of the user interface.

- Features: Provides semantic elements, multimedia support, and enhanced form controls.

**2. CSS3:**

- Purpose: Styles the web pages, ensuring a visually appealing and responsive design.

- Features: Flexbox, Grid layout, media queries for responsiveness, and animations.

**3. JavaScript:**

- Purpose: Adds interactivity to the web pages, allowing dynamic content and user interactions.

- Features: DOM manipulation, event handling, and asynchronous operations using AJAX.

**4. React.js (or another JavaScript framework/library):**

- Purpose: Facilitates the building of complex, interactive UIs with reusable components.

- Features: Component-based architecture, state management, and virtual DOM for performance optimization.

**Backend Tools**

1. **XAMPP**:
   * **Purpose**: An open-source cross-platform web server solution stack package that includes Apache, MySQL, and PHP, making it easy to set up a local server environment.
   * **Features**: Integrated package that includes Apache for web serving, MySQL for database management, and PHP for server-side scripting.
2. **PHP**:
   * **Purpose**: A server-side scripting language designed for web development, embedded into HTML.
   * **Features**: Supports database integration, session management, and a wide range of functionalities through various libraries and frameworks.
3. **MySQL**:
   * **Purpose**: A relational database management system for storing and managing structured data.
   * **Features**: ACID compliance, SQL queries, indexing, and support for transactions.

#### Development and Testing Tools

1. **Visual Studio Code (VS Code)**:
   * **Purpose**: A source-code editor used for writing and debugging code.
   * **Features**: Extensions, integrated terminal, version control support, and IntelliSense.
2. **Postman**:
   * **Purpose**: An API testing tool for developing and testing APIs.
   * **Features**: Creating and managing API requests, testing endpoints, and automating tests.
3. **Git and GitHub**:
   * **Purpose**: Version control system and hosting service for source code management.
   * **Features**: Branching, merging, pull requests, and collaboration tools.

#### Security and Performance Tools

1. **JWT (JSON Web Tokens)**:
   * **Purpose**: Provides a secure way to transmit information between parties as a JSON object.
   * **Features**: Used for authentication and information exchange, ensuring data integrity and security.
2. **phpMyAdmin**:
   * **Purpose**: A free and open-source tool written in PHP intended to handle the administration of MySQL over the Web.
   * **Features**: Manages databases, tables, columns, relations, indexes, users, permissions, and supports executing any SQL statement.

#### Deployment and Hosting Tools

1. **Heroku (or another cloud platform like AWS, Azure, or Google Cloud)**:
   * **Purpose**: A cloud platform for deploying and managing applications.
   * **Features**: Supports various programming languages, easy scaling, and integrated database services.
2. **Docker**:
   * **Purpose**: Containerization tool for creating, deploying, and running applications in isolated environments.
   * **Features**: Ensures consistency across different environments, simplifies deployment, and improves scalability.

#### User Interface (UI) and User Experience (UX) Design Tools

1. **Figma (or Adobe XD, Sketch)**:
   * **Purpose**: Design tool for creating UI/UX designs and prototypes.
   * **Features**: Collaborative design, prototyping, and UI component libraries.

By leveraging these tools, the integrated frontend and backend hospital management system achieves a high level of functionality, security, and usability, providing an efficient solution for managing hospital operations and enhancing patient care.

**Operations for SQL in Hospital Management System**

In a hospital management system, SQL operations play a crucial role in managing and manipulating the data stored in the database. Here are the key operations and their purposes within the system:

**1. Patient Management**

- Insert New Patient Record: Adding new patients to the system by capturing personal and medical information.

- Update Patient Information: Modifying existing patient details such as address, contact information, or medical history.

- Delete Patient Record: Removing patient records from the database when they are no longer needed.

- Retrieve Patient Information: Accessing patient details to view their personal and medical history.

**2. Appointment Scheduling**

- Insert New Appointment: Scheduling new appointments for patients with doctors, including date, time, and status.

- Update Appointment: Modifying appointment details, such as rescheduling or updating the appointment status.

- Delete Appointment: Canceling and removing appointments from the system.

- Retrieve Appointments: Viewing scheduled appointments for a specific doctor or patient, filtered by date or status.

**3. Medical Records Management**

- Insert New Medical Record: Adding new entries to a patient’s medical record, including diagnosis, treatment, and visit date.

- Update Medical Record: Editing existing medical records to reflect new information or changes in treatment.

- Delete Medical Record: Removing outdated or incorrect medical records from the database.

- Retrieve Medical Records: Accessing medical records for a specific patient to view their treatment history and diagnosis.

**4. Billing and Invoicing**

- Insert New Billing Record: Creating new billing entries for services provided to patients, including the amount and billing date.

- Update Billing Status: Changing the status of billing records to reflect payments received or other adjustments.

- Delete Billing Record: Removing billing entries that are no longer valid or were created in error.

- Retrieve Billing Information: Accessing billing details for patients to view charges, payments, and outstanding balances.

**5. User Management (Admin, Doctors, Staff)**

- Insert New User: Adding new users to the system, including administrators, doctors, and support staff.

- Update User Information: Modifying user details such as password, role, or contact information.

- Delete User: Removing users from the system when they leave the organization or no longer require access.

- Retrieve User Information: Viewing user profiles to manage roles and permissions.

**6. Reports and Analytics**

-Generate Appointment Reports: Create reports on the number of appointments scheduled by each doctor or within a specific time frame.

- Financial Reports: Compiling data on billing and payments to generate revenue reports for financial analysis.

- Patient Statistics: Analyzing patient data to identify trends, such as the number of new patients registered or common diagnoses.

**7. Resource Management**

- Manage Hospital Resources: Tracking and managing hospital resources, such as equipment, rooms, and staff availability.

- Optimize Resource Utilization: Ensuring efficient use of hospital resources by analyzing data on usage patterns and availability.

These SQL operations ensure efficient data management, improve the accessibility and accuracy of information and enhance the overall operational efficiency of the hospital management system.

-

**Module Description and Functionalities**

**1. Frontend Module**

- User Interface (UI)

- Dashboard: Overview of hospital activities, statistics, and quick access to key functions.

- Patient Registration: Forms for registering new patients, including personal details and medical history.

- Appointment Scheduling: Interface for scheduling, rescheduling, and canceling appointments.

- Medical Records: View and update patient medical records, including treatment history and test results.

- Billing and Invoicing: Generate and manage patient bills and invoices.

- Staff Management: Manage staff details, roles, and schedules.

- Technologies: HTML, CSS, JavaScript, React or Angular

This approach ensures that the hospital management system is modular, scalable, secure, and easy to maintain.

**IMPLEMENTATION**

**Coding**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Hospital Management System</title>

    <link rel="stylesheet" href="style1.css">

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

    <style>

        body {

            font-family: Arial, sans-serif;

        }

        header {

            text-align: center;

            padding: 20px;

            background: #007BFF;

            color: white;

        }

        nav ul {

            display: flex;

            justify-content: center;

            list-style-type: none;

            background: #333;

            padding: 0;

        }

        nav ul li {

            margin: 0 15px;

        }

        nav ul li a {

            color: white;

            text-decoration: none;

            padding: 10px 20px;

            display: block;

        }

        nav ul li a:hover {

            background: #575757;

        }

        main {

            padding: 20px;

        }

        .form-section {

            max-width: 500px;

            margin: 20px auto;

        }

        .form-group {

            margin-bottom: 15px;

        }

        .form-group label {

            display: block;

            margin-bottom: 5px;

        }

        .form-group input,

        .form-group select,

        .form-group textarea {

            width: 100%;

            padding: 8px;

            box-sizing: border-box;

        }

        footer {

            text-align: center;

            padding: 10px;

            background: #007BFF;

            color: white;

            position: fixed;

            width: 100%;

            bottom: 0;

        }

    </style>

</head>

<body>

    <!-- Login Form -->

    <div id="login-form" class="form-section">

        <h1>Login</h1>

        <form id="login" onsubmit="return authenticate(event)">

            <div class="form-group">

                <label for="username">Username:</label>

                <input type="text" id="username" name="username" required>

            </div>

            <div class="form-group">

                <label for="password">Password:</label>

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

            </div>

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

        </form>

    </div>

    <!-- Frontend Page -->

    <div id="main-content" style="display: none;">

        <header>

            <h1>Hospital Management System</h1>

            <p>Welcome, <span id="user-display"></span> | <a href="#" onclick="logout()">Logout</a></p>

        </header>

        <nav>

            <ul>

                <li><a href="#home" onclick="showSection('home')">Home</a></li>

                <li><a href="#patients" onclick="showSection('patients')">Patients</a></li>

                <li><a href="#doctors" onclick="showSection('doctors')">Doctors</a></li>

                <li><a href="#appointments" onclick="showSection('appointments')">Appointments</a></li>

                <li><a href="#billing" onclick="showSection('billing')">Billing</a></li>

                <li><a href="#contact" onclick="showSection('contact')">Contact</a></li>

            </ul>

        </nav>

        <main>

            <!-- Home Section -->

            <section id="home" class="section-content">

                <h2>Welcome to Our Hospital</h2>

                <p>We provide the best medical care.</p>

            </section>

            <!-- Patients Section -->

            <section id="patients" class="section-content">

                <h2>Patient Information</h2>

                <form id="patients-form" class="form-section" action="process\_form.php" method="POST">

                    <input type="hidden" name="form\_type" value="patients">

                    <div class="form-group">

                        <label for="patient-name">Name:</label>

                        <input type="text" id="patient-name" name="patient-name" required>

                    </div>

                    <div class="form-group">

                        <label for="patient-age">Age:</label>

                        <input type="number" id="patient-age" name="patient-age" required>

                    </div>

                    <div class="form-group">

                        <label for="patient-gender">Gender:</label>

                        <select id="patient-gender" name="patient-gender" required>

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

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

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

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="patient-address">Address:</label>

                        <textarea id="patient-address" name="patient-address" rows="4" required></textarea>

                    </div>

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

                </form>

            </section>

            <!-- Doctors Section -->

            <section id="doctors" class="section-content">

                <h2>Doctor Information</h2>

                <form id="doctors-form" class="form-section" action="process\_form.php" method="POST">

                    <input type="hidden" name="form\_type" value="doctors">

                    <div class="form-group">

                        <label for="doctor-name">Name:</label>

                        <input type="text" id="doctor-name" name="doctor-name" required>

                    </div>

                    <div class="form-group">

                        <label for="doctor-specialty">Specialty:</label>

                        <input type="text" id="doctor-specialty" name="doctor-specialty" required>

                    </div>

                    <div class="form-group">

                        <label for="doctor-phone">Phone:</label>

                        <input type="tel" id="doctor-phone" name="doctor-phone" required>

                    </div>

                    <div class="form-group">

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

                        <input type="email" id="doctor-email" name="doctor-email" required>

                    </div>

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

                </form>

            </section>

            <!-- Appointments Section -->

            <section id="appointments" class="section-content">

                <h2>Appointments</h2>

                <form id="appointments-form" class="form-section" action="process\_form.php" method="POST">

                    <input type="hidden" name="form\_type" value="appointments">

                    <div class="form-group">

                        <label for="appointment-date">Date:</label>

                        <input type="date" id="appointment-date" name="appointment-date" required>

                    </div>

                    <div class="form-group">

                        <label for="appointment-time">Time:</label>

                        <input type="time" id="appointment-time" name="appointment-time" required>

                    </div>

                    <div class="form-group">

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

                        <select id="doctor-select" name="doctor-select" required>

                            <option value="dr-smith">Dr. Smith</option>

                            <option value="dr-jones">Dr. Jones</option>

                            <option value="dr-doe">Dr. Doe</option>

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="patient-id">Patient ID:</label>

                        <input type="text" id="patient-id" name="patient-id" required>

                    </div>

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

                </form>

            </section>

            <!-- Billing Section -->

            <section id="billing" class="section-content">

                <h2>Billing</h2>

                <form id="billing-form" class="form-section" action="process\_form.php" method="POST">

                    <input type="hidden" name="form\_type" value="billing">

                    <div class="form-group">

                        <label for="billing-amount">Amount:</label>

                        <input type="number" id="billing-amount" name="billing-amount" required>

                    </div>

                    <div class="form-group">

                        <label for="billing-method">Payment Method:</label>

                        <select id="billing-method" name="billing-method" required>

                            <option value="credit">Credit Card</option>

                            <option value="debit">Debit Card</option>

                            <option value="cash">Cash</option>

                        </select>

                    </div>

                    <div class="form-group">

                        <label for="billing-notes">Notes:</label>

                        <textarea id="billing-notes" name="billing-notes" rows="4"></textarea>

                    </div>

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

                </form>

            </section>

            <!-- Contact Section -->

            <section id="contact" class="section-content">

                <h2>Contact Us</h2>

                <form id="contact-form" class="form-section" action="process\_form.php" method="POST">

                    <input type="hidden" name="form\_type" value="contact">

                    <div class="form-group">

                        <label for="contact-name">Name:</label>

                        <input type="text" id="contact-name" name="contact-name" required>

                    </div>

                    <div class="form-group">

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

                        <input type="email" id="contact-email" name="contact-email" required>

                    </div>

                    <div class="form-group">

                        <label for="contact-message">Message:</label>

                        <textarea id="contact-message" name="contact-message" rows="4" required></textarea>

                    </div>

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

                </form>

            </section>

        </main>

    </div>

    <footer>

        <p>&copy; 2024 Hospital Management System</p>

    </footer>

    <script>

        function authenticate(event) {

            event.preventDefault();

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

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

            if (username === 'admin' && password === 'password') { // Example credentials

                document.getElementById('login-form').style.display = 'none';

                document.getElementById('main-content').style.display = 'block';

                document.getElementById('user-display').textContent = username;

            } else {

                alert('Invalid username or password');

            }

        }

        function logout() {

            document.getElementById('login-form').style.display = 'block';

            document.getElementById('main-content').style.display = 'none';

        }

        function showSection(sectionId) {

            document.querySelectorAll('.section-content').forEach(section => {

                section.style.display = 'none';

            });

            document.getElementById(sectionId).style.display = 'block';

        }

        document.addEventListener('DOMContentLoaded', () => {

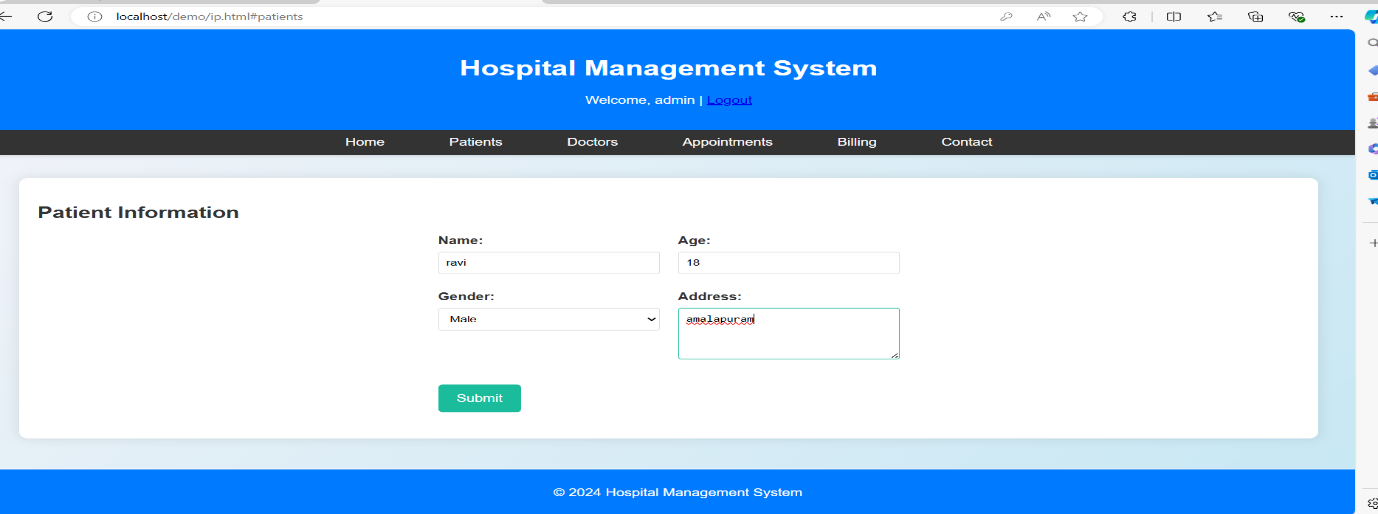
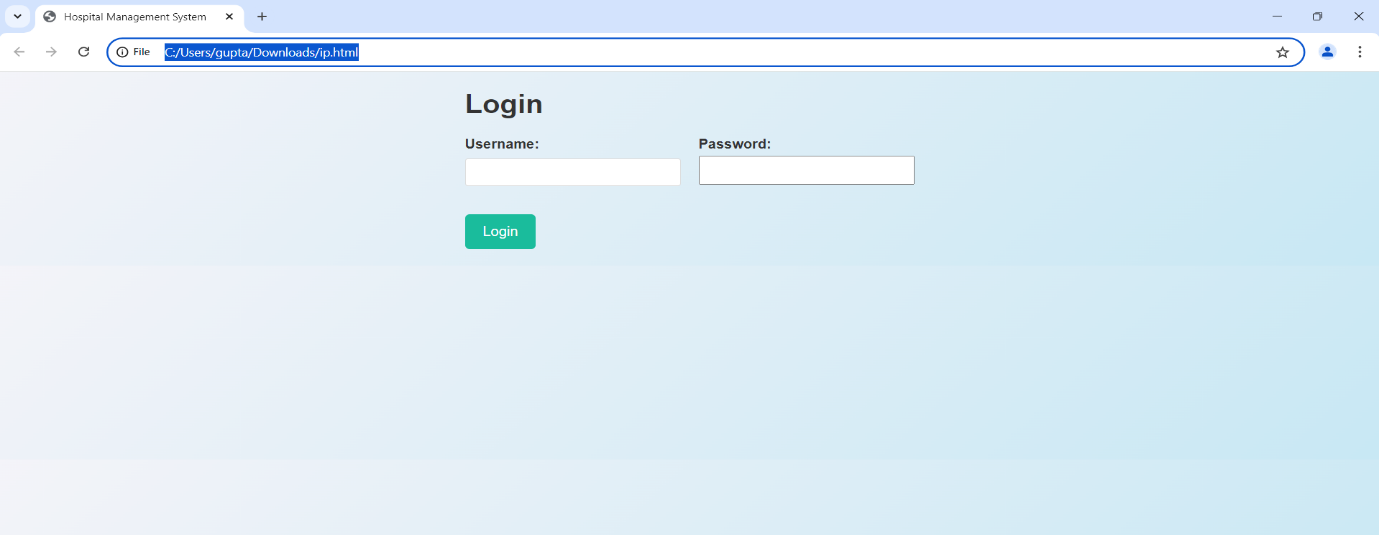
            showSection('home');

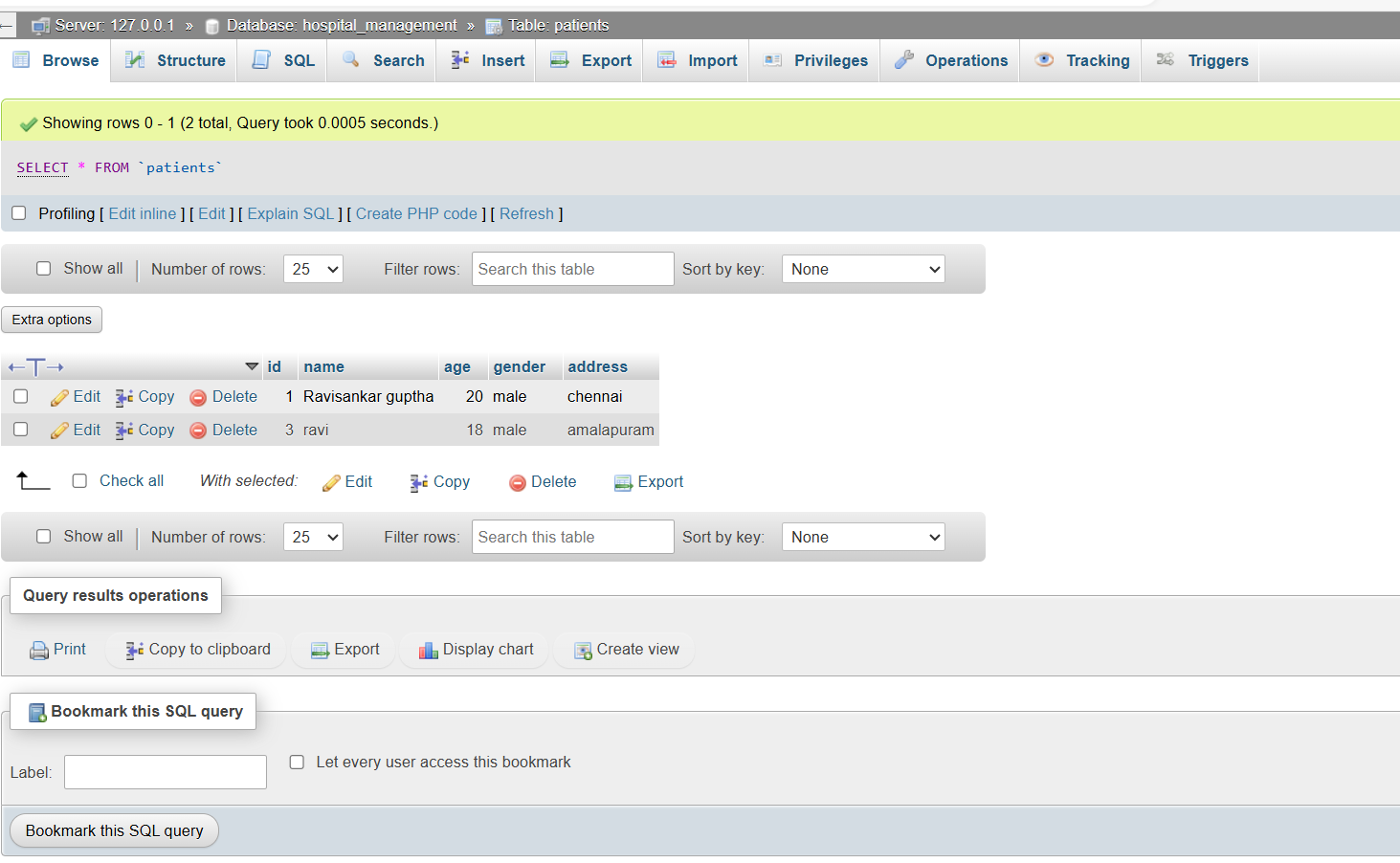
        });

    </script>

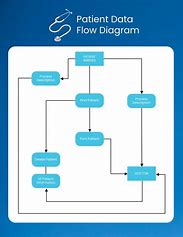
</body>

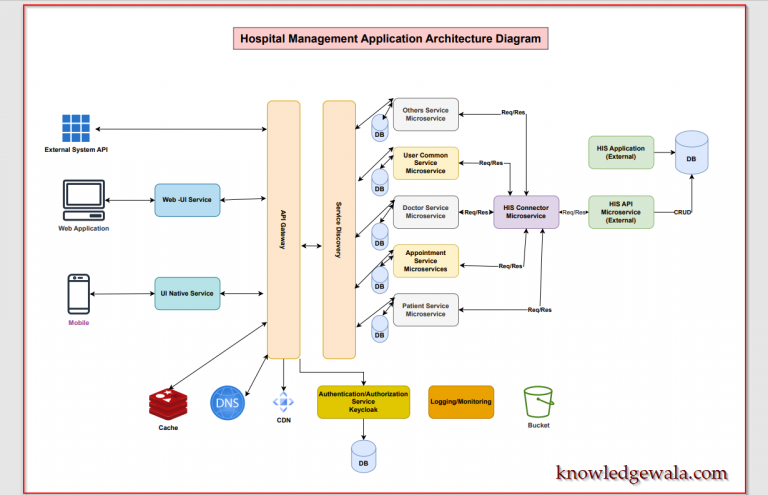
</html>

**Output**



**Architecture and flow diagram**





**PRESENTATION**

**Conclusion**

An integrated frontend and backend webpage successfully combines the user interface with server-side logic to deliver a seamless and interactive experience. By utilizing modern frameworks and technologies, such as React or Angular for the frontend and Node.js or Django for the backend, developers can create robust and efficient web applications. This integration allows for real-time data updates, improved user experience, and streamlined development processes. The resulting product is a cohesive system where the frontend and backend communicate efficiently, ensuring data integrity, security, and responsiveness, ultimately leading to higher user satisfaction and better overall performance.

**References**

 *Full stack Vue: The Complete Guide to Vue.js* by Hassan Djirdeh, Nate Murray, and Ari Lerner

 *Full-Stack React, TypeScript, and Node: Build Cloud-Ready Web Applications Using React 17 with Hooks and GraphQL* by David Choi

Full-Stack Web Development with React and Node.js" by Eric Bishard Digital Ocean

 A Practical Guide to Node.js and React" by Vasan Subramanian SitePoint

 Full-Stack Web Development with React Specialization" by The Hong Kong University of Science and Technology on [Coursera](https://www.coursera.org/specializations/full-stack-react)

Full-Stack Web Development with Node.js, Express, and MongoDB" by Colt Steele on Udemy