



UNIVERSITY OF PIRAEUS - DEPARTMENT OF INFORMATICS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc Informatics

ΠΜΣ «Πληροφορική»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	Development of a Hospital Enterprise Resource Planning (ERP) System Ανάπτυξη Συστήματος Διαχείρισης Πόρων Νοσοκομείου (ERP)
Student's name-surname: Όνοματεπώνυμο Φοιτητή:	Konstantinos Chronopoulos Κωνσταντίνος Χρονόπουλος
Father's name: Πατρώνυμο:	Nikolaos Νικόλαος
Student's ID No: Αριθμός Μητρώου:	ΜΠΠΑ21081
Supervisor: Επιβλέπων	Efthimios Alepis, Professor Ευθύμιος Αλέπης, Καθηγητής

October 2024

3-Member Examination Committee

Efthimios Alepis
Professor

Maria Virvou
Professor

Constantinos Patsakis
Associate Professor

Abstract

1. Introduction
 - 1.1. Project Overview.....8
 - 1.2. Problem Statement: What Issues Does This ERP System Address?.....10
 - 1.3. Objective: What are the main goals of the system?.....11
2. Key Features
 - 2.1. Notices.....14
 - 2.2. Contact message.....15
 - 2.3. Registration.....16
 - 2.4. Sign In.....18
 - 2.5. Dashboard Overview19
 - 2.6. Profile Page.....21
 - 2.7. Patient Management.....23
 - 2.8. Prescription.....25
 - 2.9. Schedule and Appointments.....27
 - 2.10. Pharmacy.....29
3. Database Structure
 - 3.1. Introduction.....31
 - 3.2. Detailed View of the Database Schema.....33
 - 3.3. Functions and Procedures.....44
4. Backend System Design
 - 4.1. Introduction.....48
 - 4.2. Architecture Overview.....50
 - 4.3. RESTful API Design.....53
5. Frontend System Design
 - 5.1. Introduction.....58
 - 5.2. Frontend Architecture Overview.....58
 - 5.3. Components Overview.....59
6. Security
 - 6.1. Security.....62
7. Conclusion and Future Improvements
 - 7.1. Conclusion.....65
 - 7.2. Future Improvements.....66
8. Bibliography.....68

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει την ανάπτυξη ενός ολοκληρωμένου Συστήματος Διαχείρισης Πόρων που έχει σχεδιαστεί για να βελτιστοποιήσει τις διοικητικές και κλινικές λειτουργίες των υγειονομικών ιδρυμάτων. Χρησιμοποιώντας ένα ισχυρό τεχνολογικό stack που περιλαμβάνει Java και Spring για το backend, Angular για το frontend και MySQL για τη βάση δεδομένων, το σύστημα προσφέρει μια σειρά λειτουργιών, όπως ασφαλή σύνδεση και Έλεγχος ταυτότητας βάσει ρόλων, διαχείριση συνταγογράφησης, προγραμματισμό ραντεβού και συνολική διαχείριση πόρων. Το ERP σύστημα στοχεύει στη βελτίωση της αποδοτικότητας των ροών εργασίας του νοσοκομείου, στη μείωση του διοικητικού φόρτου και στη βελτίωση της ποιότητας της παρεχόμενης φροντίδας μέσω της απρόσκοπτης ενσωμάτωσης βασικών υπηρεσιών του νοσοκομείου.

Το έργο αντιμετωπίζει κοινές προκλήσεις που αντιμετωπίζουν οι πάροχοι υγειονομικής περίθαλψης, όπως η διαχείριση μεγάλου όγκου δεδομένων ασθενών, ο συντονισμός των ραντεβού και η διασφάλιση της ασφάλειας και ακεραιότητας των δεδομένων. Με την εφαρμογή μιας modular αρχιτεκτονικής, το σύστημα μπορεί να προσαρμοστεί και να κλιμακωθεί εύκολα ώστε να καλύψει τις ανάγκες διαφόρων τμημάτων του νοσοκομείου. Μελλοντικές βελτιώσεις μπορεί να περιλαμβάνουν προηγμένες αναλύσεις, δυνατότητες τηλεϊατρικής και ενσωμάτωση με εξωτερικά health information systems, παρέχοντας μια επεκτάσιμη και βιώσιμη λύση για τη σύγχρονη διαχείριση υγειονομικής περίθαλψης.

Abstract

This thesis presents the development of a comprehensive Hospital Enterprise Resource Planning (ERP) system designed to streamline the administrative and clinical operations of healthcare institutions. Utilizing a robust technology stack comprising Java and Spring for the backend, Angular for the frontend, and MySQL for the database, the system offers a range of functionalities including secure login and role-based authentication, prescription management, appointment scheduling, and overall resource management. The ERP system aims to enhance the efficiency of hospital workflows, reduce administrative burdens, and improve the quality of patient care through seamless integration of essential hospital services.

The project addresses common challenges faced by healthcare providers, such as managing large volumes of patient data, coordinating appointments, and ensuring data security and integrity. By implementing a modular architecture, the system can be easily adapted and scaled to meet the needs of various hospital departments. Future enhancements may include advanced analytics, telemedicine capabilities, and integration with external health information systems, providing a scalable and sustainable solution for modern healthcare management.

1.1. Project Overview

The Hospital ERP System is a comprehensive software solution designed to streamline and automate the administrative and clinical operations of a healthcare institution. Built using Java with the Spring framework for the backend and Angular for the frontend, the system provides an integrated platform for managing hospital resources, patient information, appointment scheduling, and prescription management.

Core Features:

1. User Management:
 - Secure login and authentication for multiple user roles, including administrators, doctors, and patients.
 - Role-based access control to ensure data security and appropriate permissions.
2. Patient and Appointment Management:
 - Efficient scheduling and management of patient appointments.
 - Doctors can access, update, and manage patient records and appointment history.
3. Prescription Management:
 - Creation and management of prescriptions, including tracking of drugs and dosages.
 - Integration with pharmacy modules for streamlined medication dispensing.
4. Doctor Scheduling:
 - Dynamic scheduling system allowing doctors to set availability and manage their time slots.
 - Integration with appointment booking to prevent scheduling conflicts.
5. Communication and Notifications:
 - Automated email notifications for account verification, and password recovery.
 - Contact management for inquiries and communication between staff and patients.

Technical Architecture:

1. Backend: Java with Spring Boot, Spring Security for authentication, JPA/Hibernate for database interactions.
2. Frontend: Angular web application for user interfaces, enabling responsive and dynamic interactions.
3. Database: MySQL for managing structured hospital data, ensuring data integrity and performance.
4. Security: Implementation of CSRF protection and secure token-based authentication for data security.

Package Structure:

- Config: Security and authentication configurations (ProjectSecurityConfig, UsernamePwdAuthenticationProvider).
- Controller: RESTful APIs for managing accounts, appointments, prescriptions, etc.
- Service and Service Implementation: Business logic and service layer to manage operations like user registration, scheduling, and drug management.
- Repository: Data access layer for database operations, leveraging Spring Data JPA.

- Model: Data entities representing hospital resources such as User, Appointment, Drug, and more.
- DTO (Data Transfer Objects): Simplified data structures for communication between backend and frontend.
- Filter and Scheduler: Custom filters for security and scheduling mechanisms for automated tasks.

Deployment and Scalability: The system is designed to be deployed on cloud platforms, supporting horizontal scaling to handle increased load. It is optimized for performance and can be adapted to meet the specific needs of various healthcare institutions.

Future Enhancements:

- Integration with electronic medical records (EMR) systems.
- Mobile application for patient engagement and appointment booking.
- Advanced analytics for predictive healthcare management.

1.2. Problem Statement: What Issues Does This ERP System Address?

A hospital Enterprise Resource Planning (ERP) system aims to address several critical challenges faced by healthcare institutions in managing their resources, data, and operational processes. The key issues that this system addresses include:

- **Fragmented Data Management:** Hospitals often struggle with managing patient data, medical records, and administrative information across different departments and systems. The ERP system provides a unified platform for storing and accessing this data, ensuring that all departments have real-time access to accurate and consistent information.
- **Inefficient Appointment Scheduling:** Traditional methods of scheduling appointments can lead to overbooking, underutilization of resources, and long waiting times for patients. The ERP system facilitates efficient appointment scheduling and management by providing a centralized booking system that optimizes time slots and resource allocation.
- **Complex Prescription and Medication Management:** Handling prescriptions, medication stocks, and patient drug information manually can lead to errors, delays, and miscommunication between healthcare providers and pharmacies. The ERP system automates the prescription management process, ensuring accurate and timely delivery of medication information.
- **Poor Communication and Coordination:** Effective communication between different departments, such as doctors, nurses, pharmacists, and administrative staff, is crucial for providing high-quality patient care. The ERP system enhances communication and coordination through integrated messaging, task management, and notification systems.
- **Security and Compliance Concerns:** Hospitals are responsible for protecting sensitive patient information and ensuring compliance with healthcare regulations. The ERP system incorporates robust security measures, access controls, and audit trails to safeguard data and support compliance.
- **Resource Management and Optimization:** Hospitals must manage a wide range of resources, including staff, equipment, and facilities. Inefficient management can lead to resource wastage, increased costs, and reduced quality of care. The ERP system provides tools for monitoring and optimizing the utilization of resources, improving operational efficiency.
- **Manual and Redundant Administrative Processes:** Many administrative tasks, such as billing, inventory management, and human resource management, are often handled manually, leading to errors and inefficiencies. The ERP system automates some of these processes, reducing manual workload, minimizing errors, and streamlining administrative functions.
- **Difficulty in Tracking and Analyzing Data:** Hospitals generate vast amounts of data, but without the proper tools, it is challenging to analyze this data to gain insights into performance and patient outcomes. The ERP system includes analytics and reporting capabilities that enable hospital administrators to track key performance indicators (KPIs) and make data-driven decisions.
- **By addressing these issues, the hospital ERP system helps to improve patient care, streamline operations, and reduce costs, ultimately enhancing the overall efficiency and effectiveness of healthcare delivery.**

1.3 Objectives of the Hospital ERP System

The primary goal of the Hospital ERP (Enterprise Resource Planning) system is to streamline and integrate various hospital operations, improving efficiency and patient care. Below are the main objectives of the system:

1. Centralized Data Management:

One of the core objectives is to provide a centralized platform where all patient, staff, and hospital data can be stored and managed. This eliminates the need for multiple disjointed systems and manual record-keeping, reducing errors and redundancies. It ensures that all departments, such as administration and pharmacy have real-time access to accurate information, enabling better coordination and decision-making.

2. Improved Patient Care:

The ERP system aims to enhance the overall quality of patient care. By providing features such as appointment scheduling, and prescription management, it allows healthcare professionals to access and share patient information seamlessly. This leads to more accurate diagnoses, personalized treatment plans, and quicker response times. Additionally, integrated communication channels facilitate better coordination between doctors, nurses, and administrative staff.

3. Efficient Resource Management:

Efficient utilization of resources, including medical staff, equipment, and facilities, is a key objective. The system enables effective scheduling of doctors and allocation of medical equipment. This optimizes resource usage, reduces wait times, and minimizes operational costs. For example, the system can manage doctor schedules and appointment slots to avoid overbooking and ensure that all patients are attended to promptly.

5. Data Security:

Hospitals are required to comply with various regulations concerning patient data privacy and security. The ERP system is designed to ensure compliance with these standards by incorporating robust security measures, such as role-based access controls and data encryption. It also provides audit trails and reporting capabilities that facilitate compliance audits and help identify and address security vulnerabilities.

6. Scalability and Adaptability:

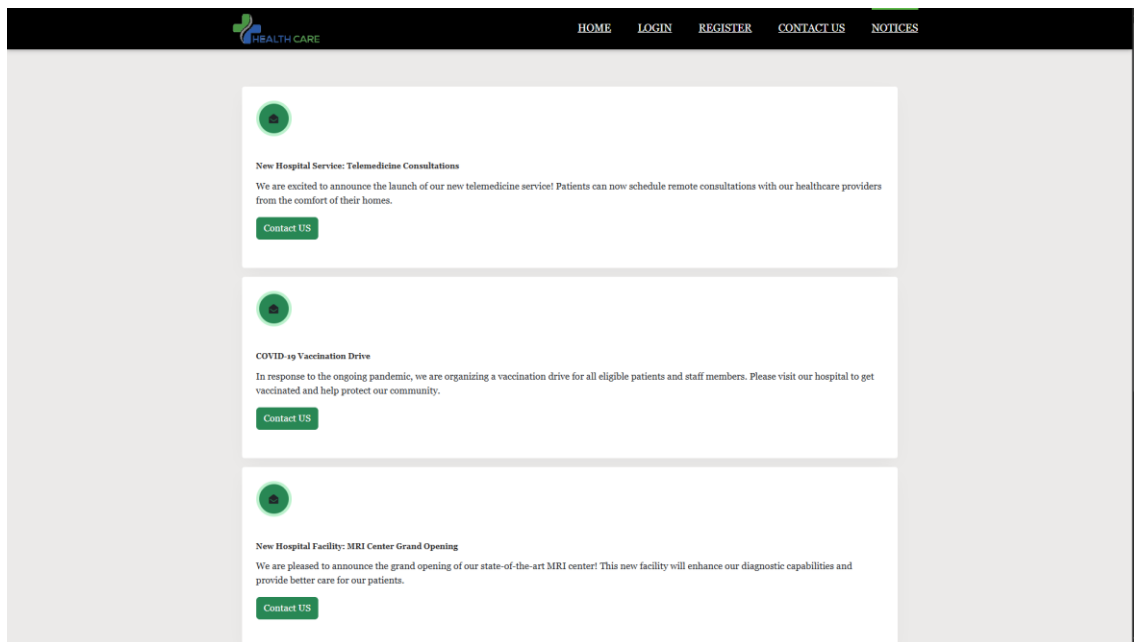
A long-term goal of the system is to support the hospital's growth and adapt to changing needs. The modular design of the ERP system allows for the easy addition of new functionalities and integration with other healthcare technologies. Whether it's expanding to support additional departments or incorporating telemedicine features, the system can evolve alongside the hospital's requirements.

2. Features Overview

2.1 Notices

Overview:

The **Notices** feature allows users to stay informed about important updates and announcements from the hospital. This section is populated with notices from the database, ensuring that staff and patients are kept up to date with the latest services and developments.



Functionality:

- **Viewing Notices:**
 - Users can navigate to the **Notices** section, where they are presented with a list of news articles or notices pulled from the database.
 - Each notice includes:
 - **Title:** A concise headline summarizing the announcement.
 - **Description:** A brief explanation of the notice, providing essential details.
- **Example Notice:**
 - **New Hospital Service: Telemedicine Consultations**
 - *We are excited to announce the launch of our new telemedicine service! Patients can now schedule remote consultations with our healthcare providers from the comfort of their homes.*

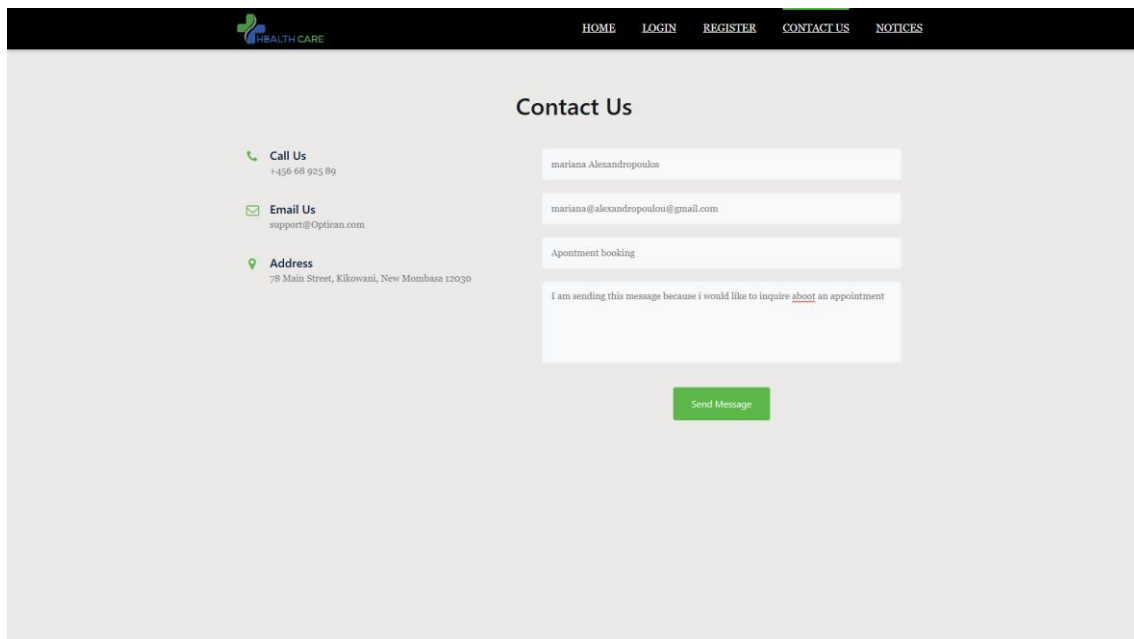
Benefits:

- **Timely Updates:** Ensures that users are always aware of new hospital services, important news, and policy changes.

2.2. Contact message

Overview:

The Contact Message feature provides users with an easy way to communicate with the hospital administration or specific departments. When users visit the “Contact Us” page, they can submit their name, email, subject, and message through a simple form. This information is then securely saved in the database for the hospital staff to review and respond to.



Functionality:

- **User Input:** The form requires the following fields:
 - **Name:** The user's name, which helps identify the sender.
 - **Email:** The user's email address, enabling the hospital staff to respond to the message.
 - **Subject:** A brief title or topic for the message, allowing staff to quickly understand the nature of the inquiry.
 - **Message:** The main content of the inquiry or feedback.
- **Data Handling:** Upon submission, the form data is stored in the database. This information is securely saved and linked to a unique entry for easy tracking and management.

Benefits:

- **Improved Communication:** Enables users to communicate directly with the hospital in a structured and organized manner.

2.3 Registration

Overview:

The Registration feature enables both patients and hospital staff (employees) to register on the system, creating individual profiles with their personal and professional details. The system captures essential information, ensuring role-based access and tailored functionality for each user type.

Functionality:

- **User Input:** The registration form includes the following fields for all users:
 - **First Name and Last Name:** For identifying the user.
 - **Email:** Used as the unique identifier for login and for communication purposes.
 - **Phone Number:** Provides an additional contact method.
 - **Password & Confirm Password:** Ensures secure access to the system by requiring users to set a password and confirm it.
 - **Gender Selection:** Users can choose between male or female.

For Employees: An additional field allows the user to select their role from the following options:

- **Doctor, Pharmacist, Secretary, Admin, Nurse, Technician.**
- **Data Handling:**
 - **Role-based Registration:** Based on the role selected, the user will be assigned specific permissions and access levels within the system.
 - **Validation and Security:** The system verifies the data entered (e.g., matching password fields, valid email, etc.) and stores the details securely in the database.

- **Registration Workflow:**
 - **Patients:** After registering, patients can access features like appointment booking, prescription management, and profile updates.
 - **Employees:** Employees will have role-based access, allowing doctors, nurses, and other staff to manage their schedules, patient interactions, and other relevant operations.
- **Confirmation:** Upon successful registration, users receive a confirmation email with instructions on how to log in and verify their account.

Benefits:

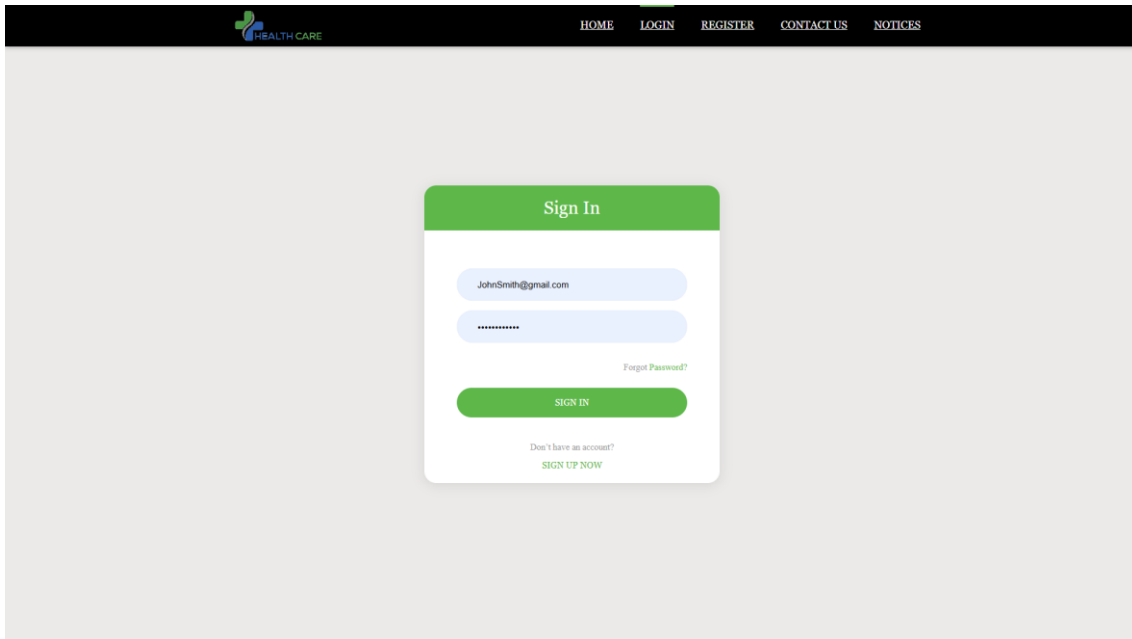
- **Flexible User Management:** Supports multiple user types with different roles and access levels, enhancing security and functionality.
- **Streamlined Onboarding:** Simplifies the registration process for both patients and employees, making it user-friendly and efficient.
- **Secure Data Entry:** Ensures the integrity of user data through form validation and secure storage protocols.

This registration feature makes the onboarding process easy and intuitive while ensuring that the system provides the right tools and access depending on whether the user is a patient or an employee.

2.4 Sign In

Overview:

The Sign In feature allows registered users to securely access their accounts in the system using their email and password. It includes a "Forgot Password" option to help users recover access in case they forget their credentials.



Functionality:

- **User Input:** The sign-in form includes the following fields:
 - **Email:** The unique identifier for the user, used to log into the system.
 - **Password:** The secure password associated with the user's account.
- **Sign In Process:**
 - Users enter their email and password and click the "Sign In" button.
 - The system validates the credentials against stored data. If the information matches, users gain access to their respective dashboards and functionalities.
- **Forgot Password:**
 - If users cannot remember their password, they can click the "Forgot Password?" link.
 - This action redirects them to a recovery page where they can enter their registered email address.
 - Upon submission, the system generates a password reset token and sends it to the provided email address, allowing users to securely create a new password.

Benefits:

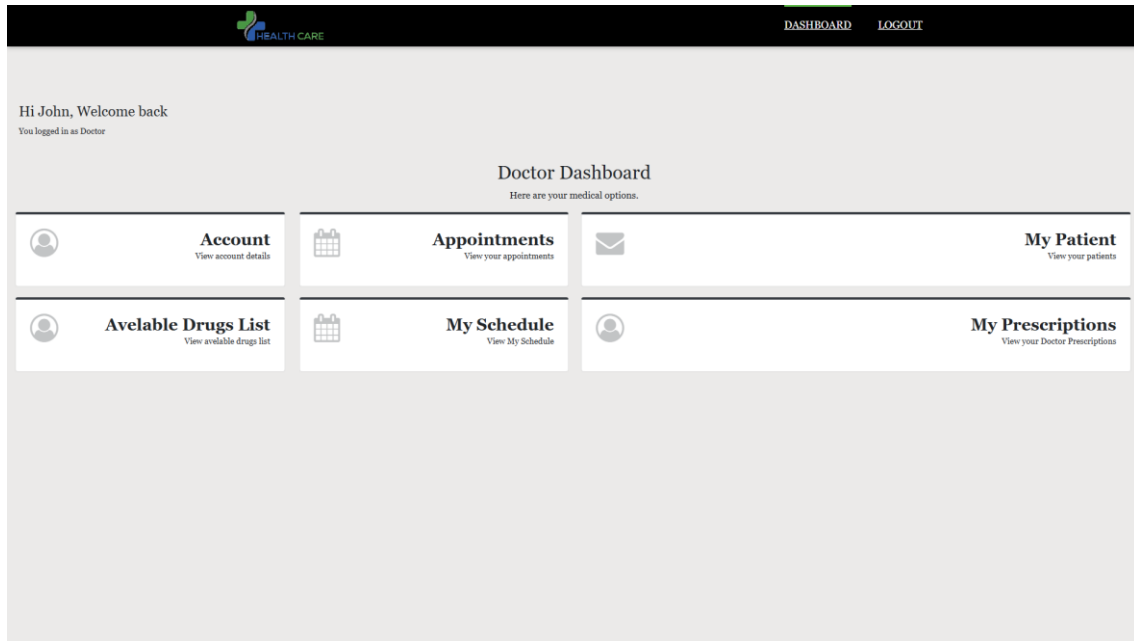
- **Secure Access:** Ensures that only authorized users can access their accounts, maintaining data security.

- **User-Friendly Recovery:** The "Forgot Password" feature simplifies account recovery, enhancing user experience and reducing frustration.

2.5 Dashboard Overview

Overview:

The Dashboard serves as the central hub for users upon logging into the system. It provides a user-friendly interface tailored to the specific needs and functionalities available to different user types, ensuring easy navigation and access to important information.



Functionality:

- **General Access:** All users see a personal account section where they can view and manage their profile information.
- **Doctor Dashboard:** Doctors have access to the following sections:
 - **Appointments:** View and manage their scheduled appointments with patients.
 - **My Patients:** Access patient profiles for those they are responsible for.
 - **Available Drugs List:** View a list of medications available for prescribing.
 - **My Schedule:** Check their personal schedule to keep track of working hours and appointments.
 - **My Prescriptions:** Review prescriptions they have issued to patients.
- **Patient Dashboard:** Patients have access to the following sections:
 - **Account:** Manage personal details and preferences.
 - **Appointments:** View upcoming appointments and book new ones.
 - **My Prescriptions:** Access their current prescriptions.
 - **Search for Appointments:** A tool to find available appointments with doctors.
- **Pharmacist Dashboard:** Pharmacists have a tailored interface with access to:
 - **Available Drug List:** View all medications available in the pharmacy.
 - **Search for Prescription:** Search for patient prescriptions to fulfill medication orders.

Benefits:

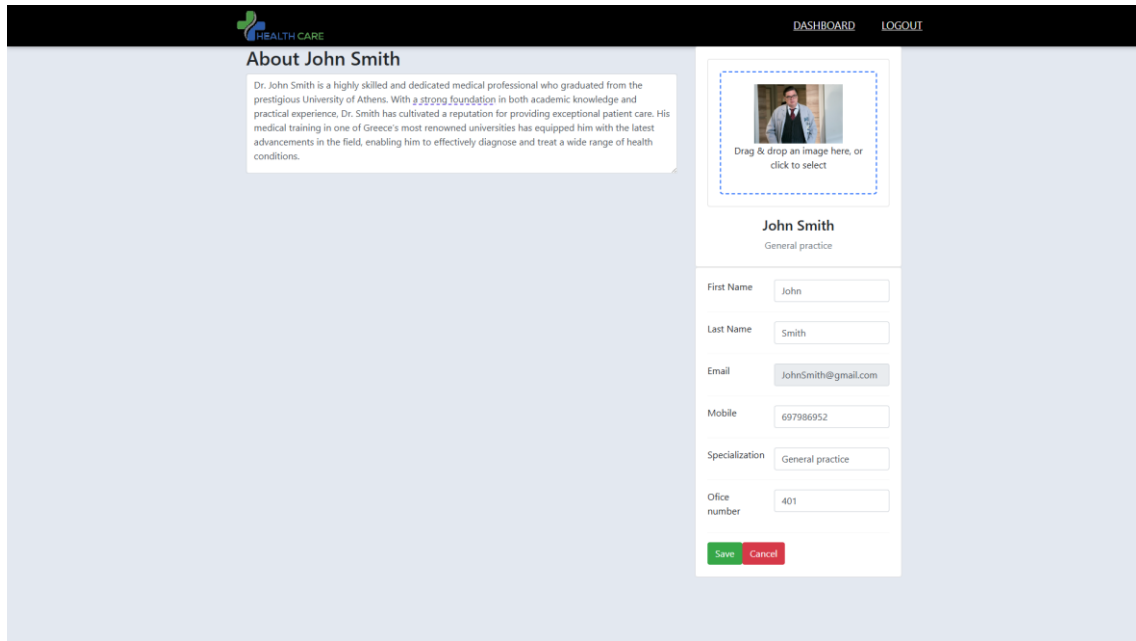
- **Personalized Experience:** Each user type has a customized dashboard that displays relevant information and actions, enhancing usability.
- **Centralized Information Access:** Users can quickly find and manage essential tasks and data, improving efficiency.

The Dashboard Overview ensures that users can easily navigate their responsibilities and access the tools they need to provide or receive care effectively.

2.6 Profile Page

Overview:

The Profile Page allows users to view and manage their personal information within the system. Each user has a dedicated profile that displays essential details, ensuring they can keep their information up to date and relevant.



Functionality:

- **Profile Information Displayed:**
 - **First Name and Last Name:** Basic identification of the user.
 - **Phone Number:** Contact information for communication.
 - **Email:** The user's registered email address (non-editable for security).
 - **Specialization:** For doctors, this indicates their area of expertise.
 - **Office Number:** The contact number for their office or practice.
 - **Profile Picture:** A visual representation that can be updated by the user.
- **Editing Capabilities:**
 - Users can click the **Edit** button to modify their profile information.
 - They can update the following fields:
 - First Name
 - Last Name
 - Phone Number
 - Specialization
 - Office Number
 - Profile Picture

- After making changes, users must click the **Save** button to update their profile.
- **Validation:** The system ensures that all changes comply with data integrity rules (e.g., valid phone numbers) before saving.

Benefits:

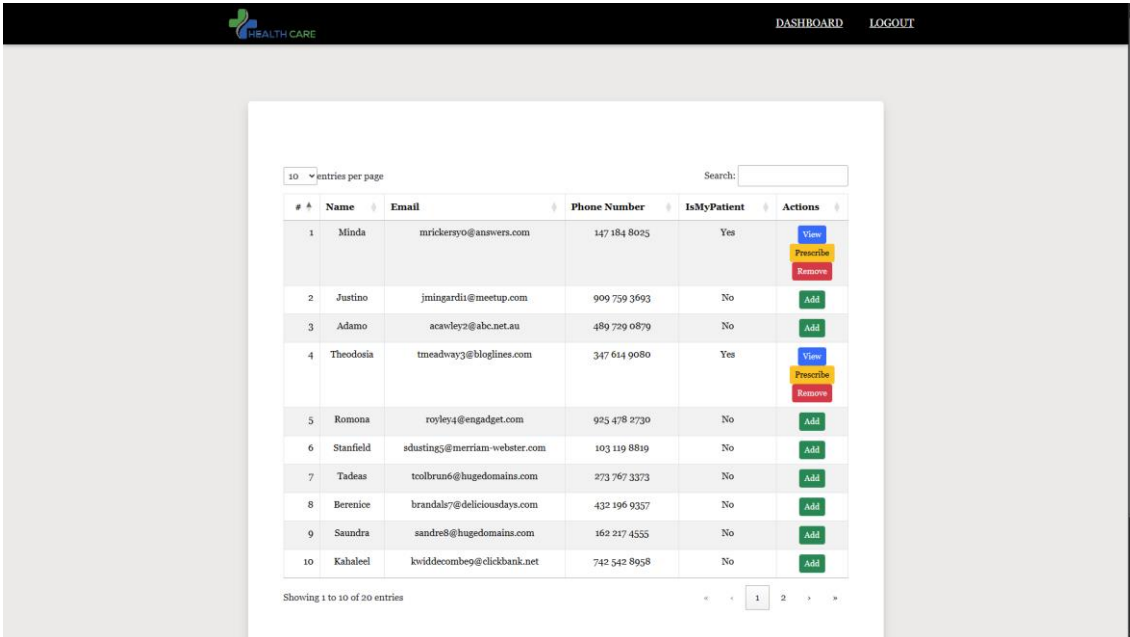
- **User Empowerment:** Enables users to take control of their personal information, ensuring accuracy and relevancy.
- **Easy Updates:** The straightforward edit-and-save process makes it simple for users to maintain their profiles.
- **Professional Presentation:** Ensures that accurate information is available to patients and colleagues, enhancing professionalism and trust.

The Profile Page feature provides a streamlined approach for users to manage their essential information, facilitating effective communication and professional representation within the system.

2.7 Patient Management

Overview:

The Patient Management feature provides doctors with an efficient way to access and manage patient information. Through the "My Patients" section, doctors can view a list of all patients in the system, with options to filter, add, or manage patients based on their needs.



#	Name	Email	Phone Number	IsMyPatient	Actions
1	Minda	mrickery0@answers.com	147 184 8025	Yes	View Prescribe Remove
2	Justino	jmingardi@meetup.com	909 759 3693	No	Add
3	Adamo	acawley2@abc.net.au	489 729 0879	No	Add
4	Theodosia	tmeadway3@bloglines.com	347 614 9080	Yes	View Prescribe Remove
5	Romona	royley4@engadget.com	925 478 2730	No	Add
6	Stanfield	sdustings@merriam-webster.com	103 119 8819	No	Add
7	Tadeas	toolbrum6@hugedomains.com	273 767 3373	No	Add
8	Berenice	brandale7@deliciousdays.com	432 196 9357	No	Add
9	Saundra	sandre8@hugedomains.com	162 217 4555	No	Add
10	Kahaleel	kwiddecombe9@clickbank.net	742 542 8958	No	Add

Showing 1 to 10 of 20 entries

Functionality:

- Patient List:** Doctors can see a table that lists all patients in the system. The table includes:
 - Patient name
 - Key information (e.g., date of birth, contact details)
 - Role-based actions based on whether the patient is assigned to the doctor or not.
- Filtering and Pagination:**
 - Filter by Keywords:** Doctors can search for specific patients using keywords, such as name, or other relevant data.
 - Pagination:** The system allows doctors to choose how many patients they want to display per page, making it easier to navigate larger lists.
- Patient Actions:**
 - Add Patient:** For patients not currently assigned to the doctor, the "Add Patient" button is displayed next to their entry in the list. Clicking this button assigns the patient to the doctor.
 - View:** For patients already assigned to the doctor, the "View" button is available, allowing the doctor to see detailed patient information, including medical history and treatment plans.
 - Prescribe:** The "Prescribe" button allows doctors to directly create and manage prescriptions for their assigned patients.

- **Delete:** The "Delete" button removes the patient from the doctor's list but does not delete the patient's data from the system.

Benefits:

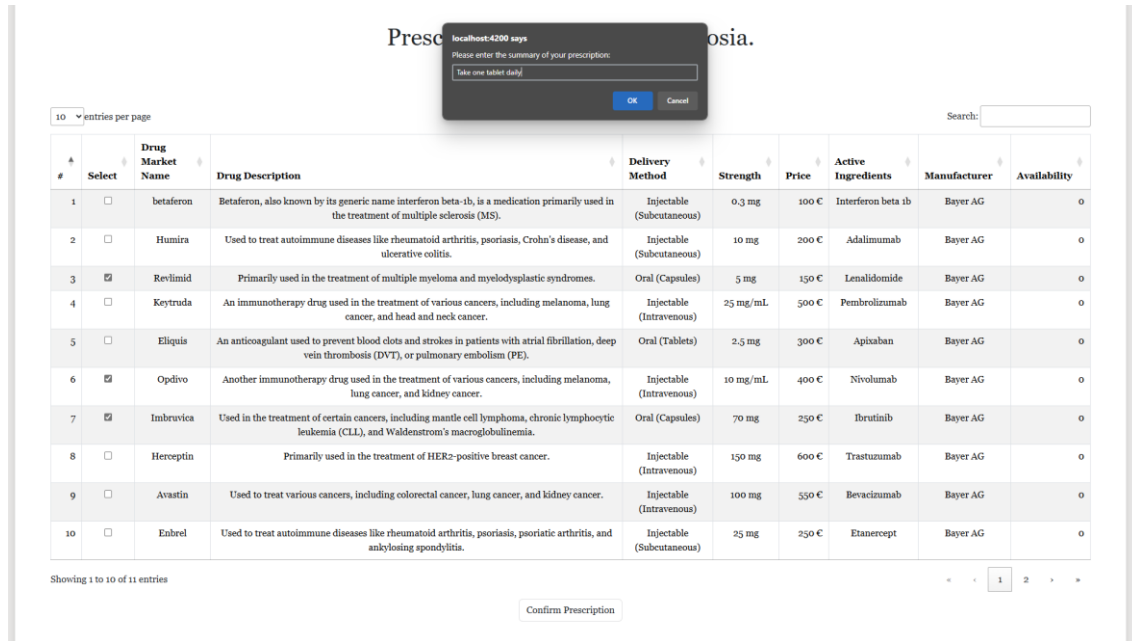
- **Efficient Patient Access:** Provides doctors with a clear overview of all patients, improving workflow and access to key data.
- **Customizable Display:** The filtering and pagination features allow doctors to tailor how they view and manage patient information, making the system scalable for both small and large practices.
- **Role-Based Management:** Ensures that doctors can quickly add new patients, view detailed profiles, prescribe medications, or remove patients from their list, making the system flexible and user-friendly.

The Patient Management feature ensures doctors can effectively organize and interact with their patient lists, helping to streamline patient care and administrative tasks.

2.8 Prescription

Overview:

The Prescription feature allows doctors to prescribe medication to their assigned patients through an intuitive, step-by-step process. This ensures that prescriptions are handled accurately and efficiently, with confirmation messages and prompts guiding the doctor through each step.



Functionality:

- **Initiating a Prescription:**
 - When a doctor clicks the **Prescribe** button next to a patient in their list, a confirmation message appears asking, “Are you sure you want to prescribe medication to [Patient's Name]?”
 - If the doctor clicks **Yes**, they are redirected to a new page to begin the prescription process.
- **Selecting Medications:**
 - On the prescription page, doctors see a table displaying the **Available Drugs**. This table includes key information for each drug, such as:
 - Drug name
 - Dosage form (tablet, capsule, etc.)
 - Strength
 - Manufacturer
 - Doctors select the drugs they wish to prescribe by checking the checkbox next to each relevant drug.
- **Finalizing the Prescription:**
 - After selecting the desired drugs, the doctor clicks **Prescribe**.

- A message prompt appears asking the doctor to add instructions or a description (e.g., "Take one tablet after dinner").
- Once the instructions are added, the doctor clicks **OK**.
- **Confirmation:**
 - The system displays a final message confirming whether the prescription was saved successfully or not. If the process is successful, the doctor is notified and the prescription is saved to the patient's record for future reference.

Benefits:

- **Streamlined Workflow:** The step-by-step process, with confirmation prompts at each stage, ensures that doctors can easily prescribe medication without missing any important details.
- **Detailed Information:** The drug table provides doctors with all necessary information, helping them make informed decisions about which medications to prescribe.
- **Custom Instructions:** The ability to add specific instructions ensures that patients receive detailed guidance on how to take their medications.

The Prescription feature simplifies the process of prescribing medication while ensuring that it remains secure and accurate, enhancing both doctor efficiency and patient care.

2.9 Schedule and Appointments

Overview:

The Schedule and Appointments feature provides doctors with the ability to manage their availability, while patients can search for and book appointments. The system ensures a seamless interaction between doctor schedules and patient bookings.

#	Date	Time	Day	Doctor Name	Doctor Specialization
1	Sep 28, 2024	08:00:00	SATURDAY	mariana alexandropoulou	cardiologist
2	Sep 28, 2024	08:30:00	SATURDAY	mariana alexandropoulou	cardiologist
3	Sep 28, 2024	09:00:00	SATURDAY	mariana alexandropoulou	cardiologist
4	Sep 28, 2024	09:30:00	SATURDAY	mariana alexandropoulou	cardiologist
5	Sep 28, 2024	10:00:00	SATURDAY	mariana alexandropoulou	cardiologist
6	Sep 28, 2024	10:30:00	SATURDAY	mariana alexandropoulou	cardiologist
7	Sep 29, 2024	08:00:00	SUNDAY	mariana alexandropoulou	cardiologist
8	Sep 29, 2024	08:30:00	SUNDAY	mariana alexandropoulou	cardiologist
9	Sep 29, 2024	09:00:00	SUNDAY	mariana alexandropoulou	cardiologist
10	Sep 29, 2024	09:30:00	SUNDAY	mariana alexandropoulou	cardiologist

Functionality:

- **Doctor’s Schedule Management:**
 - In the **My Schedule** section, doctors view a table showing days of the week (Monday, Tuesday, etc.) and times (e.g., 7:00, 7:30, etc.). Each cell in the table indicates whether the doctor is **Available** or **Unavailable** during that time slot.
 - Doctors can click on individual cells to toggle their availability between "Available" and "Unavailable."
 - After adjusting their availability, doctors click **Save** to store their schedule.
 - The backend then automatically generates empty appointment slots based on the doctor's availability for the next two months.
- **Patient Appointment Search and Booking:**
 - In the **Search for Appointments** section, patients can search for available appointments based on:
 - **Specialization:** Filter by keywords like "cardiologist" or other criteria.
 - **Date and Time:** Filter based on the desired appointment day and time.
 - **Pagination:** Patients can choose how many appointments to display per page.
 - **Sorting:** Appointments can be ordered by date, time, or other filters.
 - Once patients find a suitable appointment, they click on it, and a confirmation message appears: “Are you sure you want to book this appointment?”

- If they click **Yes**, the appointment is saved, and the patient's booking is finalized.
- **Appointment Lists:**
 - Both doctors and patients can view their respective lists of upcoming appointments in the **My Appointments** section.
 - Doctors can see all their scheduled patient appointments.
 - Patients can see their booked appointments, including the date, time, and doctor they are scheduled to see.

Benefits:

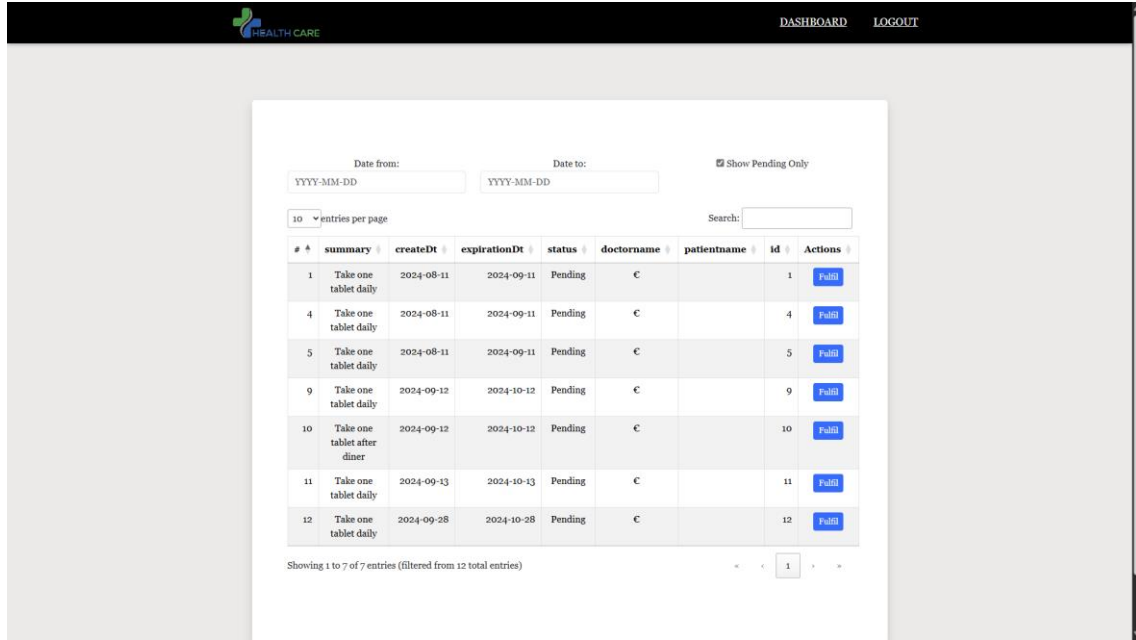
- **Efficient Scheduling:** Doctors can easily manage and update their availability, ensuring their schedules are always up to date.
- **User-Friendly Search:** Patients have multiple filters and sorting options to find appointments that fit their needs.
- **Seamless Booking:** The booking confirmation ensures patients don't accidentally schedule unwanted appointments, enhancing the overall booking experience.
- **Appointment Management:** Both doctors and patients can keep track of their appointments, improving communication and organization.

The Schedule and Appointments feature ensures a smooth and efficient process for doctors to manage their time and for patients to find and book appointments, improving the overall functionality of the system.

2.10 Pharmacy

Overview:

The Pharmacy feature enables pharmacists to efficiently manage and fulfill prescriptions through an intuitive interface. They can search for prescriptions, filter based on fulfillment status, and mark prescriptions as fulfilled, streamlining the medication distribution process.



Functionality:

- **Prescription Search:**
 - Pharmacists can access the **Search for Prescriptions** section where they see a list of prescriptions that need to be fulfilled.
 - Similar to other search functionalities in the system, pharmacists can:
 - **Filter by Keywords:** Search using keywords, such as drug names or patient details.
 - **Pagination:** Control how many prescriptions are shown per page.
 - **Filter by Fulfillment Status:** Choose to display only **Fulfilled** or **Not Fulfilled** prescriptions, or view all.
- **Prescription Fulfillment:**
 - Once a pharmacist identifies a prescription to fulfill, they click on it to proceed.
 - A confirmation message appears: “Are you sure you want to fulfill this prescription?”
 - If they click **OK**, the prescription is marked as fulfilled in the system.
 - The system updates the prescription status, indicating that the medication has been dispensed to the patient.

Benefits:

- **Streamlined Workflow:** The ability to filter and search through prescriptions ensures that pharmacists can quickly find and fulfill pending requests.
- **Fulfillment Tracking:** The system tracks which prescriptions have been fulfilled and which are pending, providing clear insights into the workflow.
- **Efficient Medication Management:** The process makes it easy for pharmacists to handle large volumes of prescriptions while maintaining clear records.

The Pharmacy feature enhances the efficiency and accuracy of prescription fulfillment, ensuring that pharmacists can manage their tasks effectively while providing patients with timely access to their prescribed medications.

3: Database Structure

3.1 Introduction

The **Hospital ERP System** relies on a well-structured relational database to manage various operations, including user registration, appointments, prescriptions, drug management, and internal communications. The database plays a pivotal role in integrating these functions, enabling a seamless and secure flow of information across different roles like doctors, pharmacists, patients, and administrators. The database is built using **MySQL**, chosen for its robustness, flexibility, and support for complex relationships and constraints. It leverages features such as **foreign keys**, **triggers**, and **data integrity checks** to ensure consistency and prevent errors. Every aspect of the hospital's day-to-day operations is closely tied to the data stored here, including critical functions like prescription fulfillment and appointment scheduling.

Key Components of the Database:

1. **User and Role Management:** The system stores detailed information on users, including doctors, patients, and administrative staff. Each user is assigned a role that dictates their access and privileges, such as booking appointments, managing prescriptions, or updating schedules.
2. **Appointment Scheduling:** Doctors' availability is managed through a time-slot-based scheduling system. This component ensures that patients can easily search for and book appointments, while doctors have control over their availability.
3. **Prescription Management:** The system tracks prescriptions, linking doctors to patients and medications. It also allows pharmacists to view and fulfill prescriptions, ensuring efficient medical care.
4. **Pharmacy and Drug Inventory:** The drug database manages details about available medications, including manufacturers and dosage information. Pharmacists can check prescription status and update the availability of drugs in real time.
5. **System Notices:** The system facilitates internal communication by storing notices and updates from the hospital administration, ensuring that users stay informed about new services, changes, or important announcements.

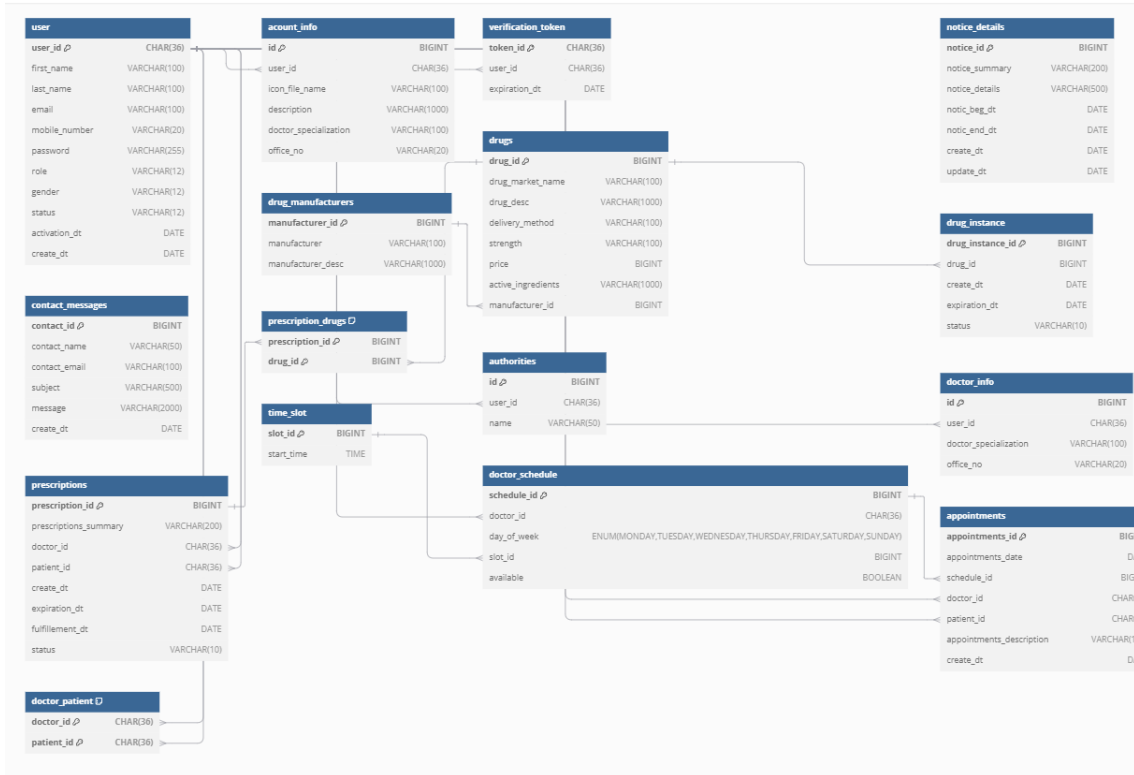
Key Design Elements:

- **Triggers and Automation:** The database is designed with triggers to automate processes like generating unique user IDs (UUIDs), creating doctor schedules, and assigning roles. This reduces the risk of manual errors and increases efficiency.
- **Data Integrity and Validation:** Constraints such as CHECK statements are used to enforce the validity of fields like roles, gender, and prescription status. This ensures that invalid data cannot be inserted, maintaining the integrity of the system.
- **Referential Integrity:** The use of foreign keys ensures relationships between tables are preserved, allowing for seamless cascading operations like deleting a user and all associated data, such as schedules and appointments.
- **Scalability:** The database is designed to handle an increasing number of users, patients, and transactions, ensuring that the hospital ERP system can scale effectively as the institution grows.

This chapter will provide a detailed overview of the database schema, explaining how each component works together to support the hospital's operations. By doing so, we highlight the database's central role in maintaining the integrity, efficiency, and scalability of the ERP system.

Database Schema Overview

The following diagram illustrates the Hospital ERP System's database schema, showing the relationships between key tables such as user, prescriptions, appointments, and drugs.



4.2 Detailed View of the Database Schema

The Hospital ERP database schema is structured to provide a seamless workflow for user management, scheduling, prescription handling, drug inventory, and other critical operations. The design emphasizes data integrity, modularity, and scalability, ensuring that different roles (e.g., Doctors, Patients, Pharmacists, Admins) can interact smoothly. Below is a detailed exploration of the schema's tables, their roles, and relationships.

1. user

The user table serves as the cornerstone of the database, storing core information for all individuals interacting with the hospital system, including employees and patients. This table manages all personal details and supports diverse roles through flexible field design.

Fields:

- `user_id`: This unique identifier (UUID) ensures that each user in the system can be referenced without ambiguity.
- `first_name`, `last_name`: The user's full name, aiding in easy identification.
- `email`: A unique email address serves as a primary contact method and login credential.
- `mobile_number`: The user's phone number, also unique, used for contact and possible two-factor authentication.
- `password`: Encrypted to ensure secure storage, ensuring compliance with best practices in data protection.
- `role`: Defines the function of the user within the system, ranging from patients to specific employee roles like Doctor, Admin, or Pharmacist. This field is crucial for access control and permissions.
- `gender`: Captures the gender of the user (Male or Female), which may be used in health-related processes.
- `status`: Reflects the current state of the account (Active, Pending, or Inactive), allowing the system to disable or activate accounts dynamically.
- `activation_dt`, `create_dt`: Timestamps used to track when the user was created and activated, which is useful for reporting and audit trails.

Relationships:

This table links to other tables such as appointments, prescriptions, and `doctor_patient`, serving as a hub for user-related data across the system.

2. account_info

The account_info table holds additional profile information, particularly useful for doctors or other healthcare providers. It stores attributes that extend beyond the basic user table, giving a more detailed profile.

Fields:

- user_id: A foreign key that links directly to the user table, ensuring that the additional profile is always tied to a valid user.
- icon_file_name: Stores the filename for the user's profile picture, enhancing the user experience by allowing visual identification of doctors.
- description: A brief narrative about the doctor's expertise or experience. This could be used in patient-facing profiles or internal records.
- doctor_specialization: Defines the medical specialty of the doctor (e.g., Cardiologist, Neurologist), which is critical for both appointment scheduling and patient assignment.
- office_no: The office number where the doctor is stationed within the hospital, providing location details for easy patient navigation.

Relationships:

The account_info table is linked to doctors and is primarily used in conjunction with other tables like doctor_schedule and appointments.

3. verification_token

This table facilitates account verification processes, especially during the user registration phase. It stores tokens sent to users via email or other methods to confirm their identity.

Fields:

- token_id: A unique identifier (UUID) for the token.
- user_id: A foreign key that ties the token to a specific user in the user table.
- expiration_dt: The date and time when the token expires. This helps ensure that verification tokens cannot be reused after a certain period, adding security.

Usage:

The verification token is used in workflows like new user sign-ups and password recovery to validate user actions before granting access.

4. notice_details

The notice_details table is designed to manage important announcements and notices within the system, such as hospital-wide news or system updates. These notices are typically displayed to users upon login or in designated areas of the system.

Fields:

- notice_summary: A short, attention-grabbing headline or title summarizing the notice.
- notice_details: Full content of the notice, which provides detailed information about the message.
- notic_beg_dt, notic_end_dt: Defines the start and end date for the visibility of the notice. After the notic_end_dt, the notice will no longer be shown to users.
- create_dt, update_dt: Timestamps indicating when the notice was created and last updated, useful for tracking changes.

Usage:

This table ensures that users stay informed of hospital policies, upcoming system maintenance, or general news.

5. contact_messages

This table collects messages sent via the contact form, typically used for user inquiries, feedback, or other communications directed towards hospital administrators.

Fields:

- contact_name: The name of the person sending the message.
- contact_email: The email address of the sender.
- subject, message: The subject line and full body of the message, respectively, which describe the inquiry or feedback.
- create_dt: Timestamp indicating when the message was sent.

Usage:

contact_messages serves as a repository for user feedback and inquiries, enabling efficient response management by administrators.

6. drug_manufacturers

The drug_manufacturers table stores information about pharmaceutical companies that produce drugs used within the hospital.

Fields:

- manufacturer_id: A unique identifier for each manufacturer.
- manufacturer: The official name of the drug manufacturer.
- manufacturer_desc: A description of the manufacturer, including background information or specialties.

Usage:

This table is referenced by the drugs table, ensuring that all drug records are tied to a valid manufacturer.

7. drugs

This table stores a comprehensive list of all drugs available within the hospital. It plays a crucial role in the prescription system and inventory management.

Fields:

- `drug_id`: A unique identifier for each drug.
- `drug_market_name`: The commercial name of the drug, which is used when prescribing or stocking the drug.
- `drug_desc`: A detailed description of the drug, including its usage, effects, and side effects.
- `delivery_method`: Specifies how the drug should be administered (e.g., oral, intravenous).
- `strength`: The dosage or potency of the drug (e.g., 500 mg).
- `price`: The cost of the drug, used for billing and inventory purposes.
- `active_ingredients`: Lists the active ingredients in the drug, which are essential for medical decision-making.
- `manufacturer_id`: A foreign key linking to the `drug_manufacturers` table.

Usage:

`drugs` is central to the prescription system and is referenced by the `prescription_drugs` and `drug_instance` tables.

8. drug_instance

The `drug_instance` table tracks individual lots or batches of drugs within the hospital's inventory. Each instance represents a specific stock of a drug, ensuring that expiration dates and availability are carefully monitored.

Fields:

- `drug_instance_id`: A unique identifier for each drug instance.
- `drug_id`: Foreign key linking to the `drugs` table.
- `create_dt`, `expiration_dt`: Timestamps tracking when the drug instance was created and when it expires.
- `status`: Indicates whether the drug instance is currently available (Available, Unavailable, Pending).

Usage:

This table ensures precise tracking of drug stocks, helping pharmacists and administrators manage inventory and avoid prescribing expired drugs.

9. prescriptions

This table handles the core functionality of the prescription management system. Doctors issue prescriptions that are tracked here, and patients can fulfill them through the hospital's pharmacy.

Fields:

- prescription_id: A unique identifier for each prescription.
- prescriptions_summary: A short summary of the prescription, usually a brief description of what it entails.
- doctor_id, patient_id: Foreign keys linking to the user table, representing the doctor who issued the prescription and the patient receiving it.
- create_dt, expiration_dt, fulfillment_dt: Timestamps representing when the prescription was issued, when it expires, and when it is fulfilled by the pharmacy.
- status: Tracks the prescription's lifecycle, such as Pending, Expired, or Fulfilled.

Usage:

The prescriptions table is central to the drug dispensing process, with relationships to both the prescription_drugs and drugs tables.

10. prescription_drugs

This table defines the relationship between a prescription and the drugs that have been prescribed. It allows multiple drugs to be associated with a single prescription.

Fields:

- prescription_id: Foreign key linking to the prescriptions table.
- drug_id: Foreign key linking to the drugs table.

Usage:

By linking prescriptions to drugs, this table supports multi-drug prescriptions and ensures that each drug prescribed is properly tracked.

11. authorities

The authorities table manages user permissions and roles, such as Administrator, Doctor, or Patient. It ensures that users are assigned the appropriate access rights based on their role in the system.

Fields:

- id: A unique identifier for each authority record.
- user_id: Foreign key linking to the user table, specifying which user the authority is assigned to.
- name: The name of the authority or role (e.g., ROLE_ADMIN, ROLE_DOCTOR, ROLE_PATIENT).

Usage:

The authorities table provides a granular way to control user access to various parts of the system, ensuring that only authorized individuals can access certain features or data.

12. doctor_patient

This table maintains the relationship between doctors and their assigned patients, tracking who is under whose care.

Fields:

- doctor_id, patient_id: Foreign keys linking to the user table, identifying the doctor and patient involved.

Usage:

The doctor_patient table ensures that patient assignments are clearly defined and easily retrievable, aiding in appointment scheduling and patient management.

13. time_slot

The time_slot table stores predefined time slots used for scheduling appointments. Each slot represents a specific period during which doctors are available for appointments.

Fields:

- slot_id: A unique identifier for each time slot.
- start_time: The time at which the slot begins.

Usage:

This table allows the system to break down each doctor's availability into specific intervals, facilitating smooth appointment scheduling.

14. doctor_schedule

The doctor_schedule table defines the availability of each doctor for specific days of the week and time slots.

Fields:

- schedule_id: A unique identifier for each schedule entry.
- doctor_id: Foreign key linking to the user table, representing the doctor.
- day_of_week: Indicates the day of the week (e.g., Monday, Tuesday) when the doctor is available.
- slot_id: Foreign key linking to the time_slot table, specifying the available time slot.
- available: Boolean field indicating whether the doctor is available in the specified time slot.

Usage:

The doctor_schedule table works closely with the appointments table to ensure that doctors are only scheduled for appointments during their designated time slots.

15. appointments

This table records all appointments between doctors and patients. Each appointment is tied to a specific doctor, patient, and time slot.

Fields:

- appointments_id: A unique identifier for each appointment.
- appointments_date: The date of the appointment.
- schedule_id: Foreign key linking to the doctor_schedule table, specifying when the appointment will take place.
- doctor_id, patient_id: Foreign keys linking to the user table, representing the doctor and patient involved.
- appointments_description: Optional field for adding a description or notes about the appointment.
- create_dt: Timestamp for when the appointment was created.

Usage:

appointments tracks the scheduling and fulfillment of all doctor-patient interactions, providing a detailed record of healthcare delivery.

3.3 Functions and Procedures

In order to enhance the performance, maintain data integrity, and automate various processes within the **Hospital ERP System**, a set of database functions, triggers, and stored procedures has been developed. These features work behind the scenes to handle critical tasks such as data validation, user account management, appointment scheduling, and prescription tracking. Below is a detailed view of the key functions and procedures that support the hospital's daily operations.

1. Triggers

Triggers are used in the database to automate actions that must take place when certain events occur, such as inserting new records or updating data. The use of triggers in the hospital database helps maintain data integrity and reduces the chances of manual errors. Here are some of the key triggers implemented:

1.1 before_insert_user

Purpose: This trigger is executed before a new record is inserted into the user table. It automatically assigns a unique identifier (UUID) to a new user if the user_id field is not provided.

Use Case: This is crucial to ensure that each user gets a unique, non-repetitive identifier that links them to various system components (e.g., appointments, prescriptions).

Code snippets:

```
CREATE TRIGGER before_insert_user
BEFORE INSERT ON `user`
FOR EACH ROW
BEGIN
IF NEW.user_id IS NULL THEN
SET NEW.user_id = UUID();
END IF;
END;
```

1.2 create_doctor_schedule

Purpose: This trigger is executed after a new doctor is added to the user table. It automatically creates a schedule for the doctor by populating the doctor_schedule table with time slots for each day of the week, initially marking all slots as available or unavailable.

Use Case: This simplifies the onboarding of new doctors, ensuring that each doctor has an initial schedule that can be later modified as needed.

Details:

- Uses a cursor to loop through all predefined time slots from the time_slot table.
- Inserts entries into the doctor_schedule table for all days of the week, setting certain slots as available by default (e.g., morning hours).

Code snippets:

```
CREATE TRIGGER create_doctor_schedule
AFTER INSERT ON `user`
FOR EACH ROW
BEGIN
    DECLARE done BIGINT DEFAULT FALSE;
    DECLARE cur_time_slot_id INT;
    DECLARE day VARCHAR(10);
    DECLARE time_slot_time TIME;

    DECLARE cur CURSOR FOR SELECT slot_id, start_time FROM time_slot;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    IF NEW.role = 'Doctor' THEN
        OPEN cur;
        read_loop: LOOP
            FETCH cur INTO cur_time_slot_id, time_slot_time;
            IF done THEN
                LEAVE read_loop;
            END IF;

            IF time_slot_time IN ('08:00:00', '08:30:00', '09:00:00') THEN
                INSERT INTO doctor_schedule (doctor_id, day_of_week, slot_id,
                available)
                VALUES
                    (NEW.user_id, 'MONDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'TUESDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'WEDNESDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'THURSDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'FRIDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'SATURDAY', cur_time_slot_id, TRUE),
                    (NEW.user_id, 'SUNDAY', cur_time_slot_id, TRUE);
            ELSE
                INSERT INTO doctor_schedule (doctor_id, day_of_week, slot_id,
                available)
                VALUES
                    (NEW.user_id, 'MONDAY', cur_time_slot_id, FALSE),
```

```
        (NEW.user_id, 'TUESDAY', cur_time_slot_id, FALSE),
    (NEW.user_id, 'WEDNESDAY', cur_time_slot_id, FALSE),
    (NEW.user_id, 'THURSDAY', cur_time_slot_id, FALSE),
    (NEW.user_id, 'FRIDAY', cur_time_slot_id, FALSE),
    (NEW.user_id, 'SATURDAY', cur_time_slot_id, FALSE),
    (NEW.user_id, 'SUNDAY', cur_time_slot_id, FALSE);
END IF;
END LOOP;
CLOSE cur;
END IF;
END;
```


1.3 create_user_authorities

Purpose: This trigger assigns roles and permissions to users after they are added to the user table. Depending on the role (e.g., Admin, Doctor, Patient), appropriate authorities are inserted into the authorities table.

Use Case: This helps maintain role-based access control by ensuring that users are given the correct permissions immediately upon registration.

Details: Each user role receives default permissions:

- Admin receives both ROLE_USER and ROLE_ADMIN.
- Doctor receives both ROLE_USER and ROLE_DOCTOR, etc.

Code snippets:

```
CREATE TRIGGER create_user_authorities
AFTER INSERT ON `user`
FOR EACH ROW
BEGIN
    IF NEW.role = 'Admin' THEN
        INSERT INTO `authorities` (`user_id`, `name`)
        VALUES (NEW.user_id, 'ROLE_USER'), (NEW.user_id, 'ROLE_ADMIN');
    END IF;
    -- Other roles handled similarly
END;
```

These functions and triggers play a crucial role in maintaining the integrity, reliability, and efficiency of the Hospital ERP System, streamlining various processes, ensuring data is accurate and up-to-date, and improving overall productivity.

4: Backend System Design

4.1 Introduction

The backend architecture of the **Hospital ERP System** is the core engine that powers all the functionalities, ensuring that data flows smoothly between users, services, and the database. This backend was developed using the **Java Spring Framework** and follows a well-structured, layered approach that allows for modular, maintainable, and scalable development.

The backend is designed to manage complex workflows involving patients, doctors, pharmacists, and administrators, providing a secure and reliable environment for managing appointments, prescriptions, drug inventories, and user interactions. This section outlines the backend's architecture, the technologies used, and how each component contributes to achieving the business goals of the Hospital ERP system.

Purpose

The main purpose of the backend is to:

1. **Centralize Data Management:** Facilitate the storage, retrieval, and modification of data in a secure and efficient way.
2. **Provide Business Logic:** Execute the core business operations, such as appointment booking, prescription handling, user role management, and communication between departments.
3. **Ensure Secure Access:** Protect sensitive healthcare data with robust authentication and authorization mechanisms.
4. **Integrate System Components:** Offer a unified platform where different hospital functions (like scheduling, drug management, and communication) are integrated, promoting streamlined operations and reducing the need for manual intervention.

Technologies Used

- **Java Spring Boot:**
 - The backbone of the application, **Spring Boot** provides an easy-to-develop framework with built-in tools for dependency management, application configuration, and a simplified setup for creating RESTful APIs.
 - **REST APIs:** The backend exposes well-defined endpoints that facilitate interaction between the frontend and the backend, allowing seamless operations for end users.
 - **Dependency Injection:** Spring's dependency injection mechanism makes it easy to manage different components, keeping the code modular and reducing coupling between layers.
 - **Layered Architecture:** The backend follows a layered structure, segregating different functionalities into Controller, Service, and Repository layers. This separation of concerns ensures that each layer performs its own specific responsibilities, making the application easier to maintain and expand.
- **Spring Security:**
 - **Authentication and Authorization:** **Spring Security** is responsible for authenticating users and authorizing access to different parts of the system based on their roles (e.g., Doctor, Patient, Admin).
 - **Role-Based Access Control (RBAC):** Users are assigned specific roles with restricted access to the features relevant to their responsibilities, ensuring sensitive medical data is protected.

- **JWT (JSON Web Tokens):** Used to create a stateless session system that ensures secure communication between clients and the backend. Each token carries encrypted information about the user and their roles, facilitating both verification and role-based decision-making during each request.
- **Hibernate/JPA (Java Persistence API):**
 - **ORM for Database Interaction:** **Hibernate** provides an **Object-Relational Mapping (ORM)** solution that makes it easier to interact with the database by converting database records to Java objects.
 - **Abstraction and Efficiency:** By using JPA, the backend abstracts database operations, allowing developers to focus on business logic instead of raw SQL queries. This abstraction provides more readable, maintainable code while also ensuring data integrity.
 - **Entity-Relationship Mapping:** Each database entity (such as users, appointments, drugs) is mapped to a corresponding Java class. Relationships between entities, such as those between doctors and patients or between prescriptions and drugs, are represented using appropriate annotations like `@OneToMany` or `@ManyToOne`.
 - **Database Transactions:** Supports transaction management, ensuring that multi-step processes (such as booking an appointment or issuing a prescription) are executed completely or rolled back in case of any errors, preventing partial data changes.

Together, these technologies form a powerful, secure, and scalable backend capable of handling the diverse and intricate workflows of a hospital. By leveraging Spring Boot for rapid application development, Spring Security for robust authentication, and Hibernate/JPA for efficient database interactions, the Hospital ERP System backend supports a rich set of functionalities that deliver a seamless experience to healthcare providers, patients, and administrators. This integration of multiple technologies ensures a reliable, maintainable, and performance-optimized backend architecture, which forms the foundation for a highly effective healthcare management system.

4.2 Architecture Overview

The backend of the **Hospital ERP System** is designed with a **layered architecture** to separate concerns, enhance maintainability, and promote scalability. This architectural pattern allows each component to perform a specific role, ensuring clear boundaries between functionalities and making it easy to expand or modify the system as needed.

Layers of the Backend System

The backend architecture can be divided into the following main layers:

1. **Controller Layer (API Layer)**
2. **Service Layer (Business Logic Layer)**
3. **Repository Layer (Data Access Layer)**
4. **Security Layer (Access Control and Authentication)**

1. Controller Layer (API Layer)

Overview:

- The **Controller Layer** serves as the entry point for all incoming HTTP requests. It is responsible for handling user input, processing API requests, and returning responses.
- This layer is implemented using **Spring MVC**, part of the **Spring Boot** framework, which allows the backend to expose **RESTful APIs**.

Responsibilities:

- **Request Mapping:** Controllers map incoming HTTP requests to appropriate methods using annotations such as `@GetMapping`, `@PostMapping`, etc.
- **Validation:** Basic validation of incoming request parameters to ensure that the data received is correct.
- **Delegation:** Delegates the actual processing logic to the **Service Layer**. Controllers do not contain business logic; instead, they pass requests to services.

Example Controllers:

- **UserController:** Manages user-related endpoints, such as registration, login, and profile updates.
- **AppointmentController:** Handles endpoints for creating, modifying, and viewing appointments.
- **PrescriptionController:** Allows doctors to create prescriptions and pharmacists to view and fulfill them.

2. Service Layer (Business Logic Layer)

Overview:

- The **Service Layer** contains the core business logic of the application. It is where data from the **Repository Layer** is processed and where the rules and workflows of the hospital system are implemented.

Responsibilities:

- **Business Rules Execution:** Contains all the business rules, such as appointment booking restrictions, prescription fulfillment, and inventory management.
- **Data Transformation:** Transforms data as required by the application, such as converting an appointment entity to a response DTO for API consumers.

- **Interaction with Repositories:** Interacts with the **Repository Layer** to read and write data, utilizing CRUD operations for different entities.

Example Services:

- **UserService:** Handles user registration, assigning roles, updating profiles, and managing user-related workflows.
- **AppointmentService:** Manages doctor availability, patient booking, and the updating of appointment details.
- **PrescriptionService:** Facilitates the creation, management, and fulfillment of prescriptions, linking them with available drugs.
- **DrugService:** Manages drug inventory, availability, and interactions with prescriptions.

3. Repository Layer (Data Access Layer)**Overview:**

- The **Repository Layer** is responsible for direct database access and interacting with **MySQL** through **Spring Data JPA**. It abstracts the complexities of querying the database and provides a simplified interface for data manipulation.

Responsibilities:

- **CRUD Operations:** Provides basic operations for creating, reading, updating, and deleting database records.
- **Custom Queries:** Supports writing complex, custom queries using JPA methods or native SQL when required.
- **Data Handling:** Ensures efficient interaction with the underlying database, leveraging JPA and Hibernate for seamless integration between Java objects and relational data.

Example Repositories:

- **UserRepository:** Interfaces with the user table for operations such as fetching user details based on their email, role, or unique ID.
- **AppointmentRepository:** Handles data operations related to the appointments table, such as finding appointments by doctor or patient.
- **DrugRepository:** Manages the retrieval and storage of drug-related data, including inventory and drug manufacturers.

4. Security Layer (Access Control and Authentication)**Overview:**

- The **Security Layer** is responsible for protecting the system from unauthorized access, managing user sessions, and ensuring that data is accessed only by authorized users. This layer is implemented using **Spring Security**.

Responsibilities:

- **Authentication:** Verifies user identity, typically by comparing credentials (email/password) with the data stored in the system.
- **Authorization:** Assigns roles to users (ROLE_DOCTOR, ROLE_PATIENT, ROLE_ADMIN) and restricts access to resources based on these roles.
- **Token Management:** Uses **JWT (JSON Web Tokens)** for stateless authentication, enabling secure and scalable session management.

- **CSRF Protection:** Utilizes filters like **CsrfCookieFilter** to protect against Cross-Site Request Forgery attacks.

Components:

- **SecurityConfig:** Configures Spring Security settings, such as which endpoints are publicly accessible and which require authentication.
- **JwtAuthenticationFilter:** Verifies JWT tokens for each request, extracting the user information and roles for proper authorization.
- **CustomAuthenticationProvider:** Implements custom user authentication, aligning the backend security with the system's business-specific needs.

Overall Flow of Data Through Layers:

1. **User Interaction (Controller Layer):**
 - A user initiates an action, such as booking an appointment, by making an HTTP request.
2. **Processing Logic (Service Layer):**
 - The **Controller** receives the request and passes it to the **Service Layer**.
 - The **Service** validates business rules (e.g., checking if the doctor is available during the requested time slot).
3. **Database Interaction (Repository Layer):**
 - If valid, the **Service** layer interacts with the **Repository Layer** to create or update data in the database.
4. **Security Checks (Security Layer):**
 - Throughout this process, security components ensure that the request is authorized, and users are allowed to perform the action.

Scalability and Maintainability:

The **layered architecture** allows each layer to evolve independently, making it easier to scale the system, add new features, or update existing components without affecting the overall stability.

Controller, **Service**, and **Repository** layers maintain clear separation, allowing for improved testing and debugging capabilities. This modularity is crucial in the healthcare domain, where system reliability and security are paramount.

Summary

The backend architecture of the **Hospital ERP System** is designed to handle complex workflows while remaining flexible and scalable. Each layer plays a distinct role—**Controllers** act as intermediaries between users and services, **Services** handle business logic, **Repositories** provide data access, and **Security** ensures secure operations. By structuring the backend in this way, the system ensures a secure, maintainable, and efficient foundation to deliver a reliable and comprehensive healthcare management solution.

4.3 RESTful API Design

The **Hospital ERP System** backend exposes a set of **RESTful APIs** to handle various interactions between users (patients, doctors, pharmacists, and admins) and the system. These APIs are organized by functionality and are implemented using the **Java Spring Boot** framework. This section provides an overview of the major APIs, highlighting their endpoints, HTTP methods, functionalities, and security configurations.

Overview of API Structure

The APIs are organized based on the application's primary features and entities, such as user management, appointments, prescriptions, contact inquiries, schedules, and drug management. The APIs are designed to adhere to **REST principles**, making them easy to understand and use, while providing a clear structure for clients (e.g., web or mobile apps) to interact with the system.

Security and Access Control

- **Authentication and Authorization:** The APIs are protected using **Spring Security**. Users must authenticate themselves using credentials (username and password), and roles are assigned to enforce access control.
- **JWT Tokens:** Authentication is performed using **JWT tokens**, ensuring stateless sessions for secure communication.
- **Role-Based Permissions:** APIs are restricted based on roles, such as ADMIN, DOCTOR, PATIENT, PHARMACIST, etc., which dictate who can access specific endpoints.

The following is an overview of the major API groups, their functionalities, and how they contribute to the system's overall goals:

1. User Management APIs

Purpose: Handle user registration, login, and profile management for all users (doctors, patients, and hospital staff).

- **Endpoints:**
 - **POST /register:** Register a new user. Supports roles like Patient, Doctor, Pharmacist, etc. This endpoint also triggers an email verification process.
 - **GET /user:** Retrieve user details after successful login. Available for authenticated users.
 - **GET /myAccount:** Retrieve account details for the currently logged-in user.
 - **POST /resetEmailRequest:** Request an email to reset the user password.
 - **POST /changePassword:** Change the user's password by validating a verification token.
 - **POST /updateUserAccount:** Update user profile information such as contact details, specialization, or profile picture.
 - **GET /confirmEmail:** Confirm user registration via a verification token sent to the registered email.
- **Role Permissions:**
 - Most endpoints are open to all users for registration purposes (/register, /confirmEmail). Others require specific roles like USER, DOCTOR, or ADMIN to access profile-related data.

2. Appointment Management APIs

Purpose: Allow patients to book appointments, doctors to manage schedules, and both doctors and patients to view scheduled appointments.

- **Endpoints:**
 - **GET /PatientAppointment:** Get a list of appointments for the logged-in patient.
 - **GET /DoctorAppointment:** Get a list of appointments for the specified doctor.
 - **GET /AppointmentById:** Retrieve appointment details by appointment ID.
 - **GET /AppointmentByCriteria:** Search for appointments based on various criteria (e.g., doctor specialization, available slots).
 - **POST /updateAppointment:** Update the details of an existing appointment (e.g., changing appointment date or doctor).
- **Role Permissions:**
 - Patients and doctors can view and modify their appointments, while admins can have broader access for management purposes.
 - Specific endpoints such as /DoctorAppointment are restricted to roles like DOCTOR to ensure privacy and control.

3. Prescription Management APIs

Purpose: Facilitate the process of prescribing, viewing, and fulfilling prescriptions by doctors and pharmacists.

- **Endpoints:**
 - **GET /myPatientPrescription:** View all prescriptions for a specific patient. Available to doctors treating that patient.
 - **GET /myDoctorPrescription:** View all prescriptions issued by a specific doctor.
 - **GET /PrescriptionById:** Retrieve prescription details by prescription ID.
 - **POST /createPrescription:** Allows doctors to create a new prescription for a patient, including selecting drugs.
 - **POST /updatePrescription:** Allows modifications to an existing prescription (e.g., adding or changing prescribed drugs).
 - **GET /SearchForPrescription:** Allows pharmacists to search for prescriptions that need to be fulfilled.
- **Role Permissions:**
 - Prescriptions can only be created by doctors (ROLE_DOCTOR), while fulfillment actions are restricted to pharmacists (ROLE_PHARMACIST). Patients have read-only access to their own prescriptions.

4. Doctor Schedule Management APIs

Purpose: Allow doctors to manage their schedules, view availability, and allow patients to see available time slots for booking appointments.

- **Endpoints:**
 - **GET /DoctorsSchedule:** View the schedule for a specific doctor.
 - **GET /Schedule:** Retrieve details of a specific time slot.
 - **POST /UpdateSchedule:** Update the availability of an existing schedule entry for a doctor.
 - **POST /CreateNewSchedules:** Create new schedule entries for a doctor, specifying available time slots.
- **Role Permissions:**
 - Access to these endpoints is limited to users with DOCTOR or ADMIN roles to ensure only authorized personnel can manage schedules.

5. Drug and Inventory Management APIs

Purpose: Manage drug information, inventory levels, and support prescription fulfillment by pharmacists.

- **Endpoints:**
 - **GET /myAvailableDrugs:** Retrieve details of all available drugs, including quantities. Available to doctors and pharmacists.
 - **GET /myDrugs:** Retrieve a list of all drugs with instance count.
 - **GET /SearchForPrescription:** Used by pharmacists to look for prescriptions that are yet to be fulfilled.
- **Role Permissions:**
 - Only pharmacists (ROLE_PHARMACIST) can perform actions related to prescription fulfillment, while doctors can access drug availability to create prescriptions.

6. Contact and Notice APIs

Purpose: Facilitate user inquiries, feedback, and display hospital notices to all users.

- **Endpoints:**
 - **POST /contact:** Submit a contact form with a message to the hospital administration. Open to all users.
 - **GET /notices:** Retrieve all current notices published by the hospital. Available to all users.
- **Role Permissions:**
 - Both endpoints are publicly accessible to ensure that all users can submit inquiries and stay updated on hospital news.

7. File Management APIs

Purpose: Provide a mechanism for uploading and downloading files, such as medical records or documents.

- **Endpoints:**
 - **POST /upload:** Allows users to upload files, such as medical documents. The file is stored with a unique identifier for future retrieval.
 - **GET /getFiles:** Lists all available files that have been uploaded to the server.
 - **GET /download/{path}:** Allows users to download files by specifying the file path.
- **Role Permissions:**
 - File management actions can be restricted based on user type. Patients may only upload personal records, while doctors can access patient records they are authorized to view.

Summary

The **RESTful APIs** of the Hospital ERP system are built around key functional domains, including user management, appointment scheduling, prescription handling, and drug management. These APIs follow a layered approach with **controllers** delegating business logic to **services** and data persistence handled through **repositories**. Security is enforced using **Spring Security** to ensure that all interactions are authenticated and authorized, protecting sensitive patient data and ensuring that users can only access features relevant to their roles.

This comprehensive API structure supports an efficient healthcare workflow, providing doctors, patients, pharmacists, and administrative staff with the tools they need to manage hospital activities seamlessly and securely.

5 Frontend System Design

5.1 Introduction

The **frontend** of the **Hospital ERP System** is a comprehensive web-based interface designed to offer an intuitive and responsive user experience for doctors, patients, pharmacists, and hospital administrators. The frontend is built using **Angular 14**, leveraging modern web technologies to create a seamless, interactive application that integrates with the backend through **RESTful APIs**. The primary goal is to provide a user-friendly experience, ensuring that users can easily manage appointments, prescriptions, profiles, and other healthcare-related operations.

Key functionalities of the frontend include user authentication, appointment scheduling, prescription management, profile handling, drug availability, and internal communications. By using **Bootstrap** and **Angular Material**, the interface is designed to be responsive, providing a consistent experience across different devices.

5.2 Frontend Architecture Overview

The **frontend architecture** of the Hospital ERP system follows a **modular component-based structure** with a focus on separation of concerns. Each module and component has a specific responsibility, promoting code reuse, easier maintenance, and scalability.

Key Features of the Architecture:

1. **Modular Component Structure:**

The application is built with Angular's **component-based architecture**, which allows features to be encapsulated into individual, reusable components. Each feature, such as user authentication, appointments, prescriptions, etc., is implemented as a separate component.

2. **Routing:**

Angular's Router is used to manage navigation between different components, enabling a single-page application (SPA) experience. Users can move seamlessly between pages without the need for full page reloads, enhancing the performance and usability of the application.

Routing also employs route guards to ensure only authenticated users can access specific routes, enforcing role-based access to various parts of the application.

3. **Services and Dependency Injection:**

The architecture utilizes Angular services to handle data exchange between components and the backend via HTTP requests. Services ensure consistent access to business logic and data, promoting a centralized approach to handling HTTP requests and responses.

Dependency injection is extensively used to manage the dependencies between components and services, making the application more modular and easier to maintain.

4. **State Management:**

State is managed across components using services to hold and share data, such as logged-in user details, appointment information, and drug availability. This provides consistency and reduces redundancy in accessing data.

5. **Responsive Design:**

The application leverages Bootstrap 5 to ensure responsiveness, adapting to different screen sizes and resolutions. Angular Material is also used to incorporate UI elements like forms, buttons, and cards, making the user interface interactive and visually appealing.

6. **Role-Based UI:**

Different components are shown or hidden based on the user's role (e.g., doctor, patient, pharmacist). This is achieved using route guards, conditional rendering, and services that manage user roles, ensuring that users see only what is relevant to their responsibilities.

5.3 Components Overview

The **frontend** application is composed of several components, each serving a specific function within the Hospital ERP system. These components are categorized based on their purpose and user interaction. Below is an overview of the major components:

1 Authentication System Components

- **RegisterComponent:**
Handles the registration of new users, including patients, doctors, and other hospital staff. It collects necessary user information and sends it to the backend for processing.
- **LoginComponent:**
Manages user authentication by allowing users to enter their credentials and receive a token for further interactions with the system.
- **LogoutComponent:**
Logs the user out of the system, clears session data, and redirects the user to the login page.
- **ResetPasswordComponent:**
Allows users to initiate a password reset by entering their email address. A reset link is sent to their email, which they can use to set a new password.
- **ChangePasswordComponent:**
Allows users to change their password after authenticating with a valid token, which is typically received through the reset password process.

2 Dashboard and Home Components

- **HomeComponent:**
Acts as the landing page for all users. It provides basic information about the hospital, services offered, and highlights key features of the ERP system.
- **DashboardComponent:**
Displays an overview of the user's key information. The dashboard content is role-based:
 - **Doctors** see their schedule, appointments, and patient information.
 - **Patients** see upcoming appointments, recent prescriptions, and notices.
 - **Pharmacists** see pending prescriptions for fulfillment.

3 User Management Components

- **AccountComponent:**
Allows users to view and update their profile details, including contact information, gender, role-specific data, and profile picture. It offers an editable form to modify fields except for email, which remains constant.
- **ProfilePageComponent:**
Provides a more detailed version of a user profile. For doctors, it displays specialization, office information, and contact details, enhancing transparency for patients accessing this information.

4 Appointment Management Components

- **AppointmentsComponent:**
Displays a list of appointments for the current user, whether a patient or a doctor. Patients see their booked appointments, while doctors see all upcoming consultations.
- **CreateAppointmentComponent:**
Allows patients to create a new appointment. Patients after selecting an available appointment and confirm their booking.
- **SearchForAppointmentsComponent:**
Provides a search feature for patients to find and book available slots with doctors. Filters include specialization, date, and time.

5 Drug Management Components

- **DrugsListComponent:**
Displays all available drugs and their details, accessible to authorized roles such as doctors and pharmacists.
- **AvailableDrugsListComponent:**
A more specific component used by doctors and pharmacists to view drugs that are currently available for prescription, ensuring that prescribed medications are in stock.

6 Prescription Management Components

- **PatientListComponent:**
Shows a list of all patients assigned to the logged-in doctor or not. Doctors can select a patient to either view or issue new prescriptions or remove/add from there patients.
- **PrescriptionComponent:**
Allows doctors to create new prescriptions for selected patients. It includes fields to select drugs, specify dosages, and add any additional instructions.
- **ViewPrescriptionComponent:**
Allows both doctors and patients to view the details of issued prescriptions. It includes information like drug name, and any instructions provided by the doctor.
- **PrescriptionSearchComponent:**
Enables pharmacists to search and view prescriptions that need to be fulfilled. It helps them track pending and fulfilled prescriptions efficiently.

7 Contact and Notices Components

- **ContactComponent:**
Offers a contact form for users to send inquiries or provide feedback to the hospital administration. It includes fields for the user's name, email, subject, and message content.
- **NoticesComponent:**
Displays important notices issued by the hospital, such as new services, policy updates, or health advisories. It is accessible to all users to ensure hospital-wide communication.

8 Doctor Schedule Management Components

- **MyScheduleComponent:**

Allows doctors to manage their weekly availability. They can set available time slots for patients to book, ensuring an organized appointment schedule.

9 Registration and User Onboarding Components

- **NewRegistrationPageComponent:**
- A specialized registration component designed to handle the onboarding of new hospital employees, including doctors and pharmacists, in addition to patients. It includes role-specific fields such as specialization for doctors.

Summary

The frontend of the Hospital ERP System offers a comprehensive, user-friendly interface for all stakeholders, built on Angular 14 with modular components. By breaking down functionalities into reusable components, the system can be easily maintained and extended. Each component addresses a specific aspect of the user experience, whether it is managing user accounts, scheduling appointments, handling prescriptions, or interacting with the hospital administration. This modular approach, combined with effective routing and services for data management, forms a responsive and scalable frontend application that ensures smooth healthcare operations and enhances user satisfaction.

6 Security

The **Hospital ERP System** includes a comprehensive and robust security framework to protect sensitive healthcare data, ensure user privacy, and enforce role-based access to application resources. Security in the system is primarily implemented using **Spring Security**, a powerful and customizable security framework that integrates with the **Java Spring Boot** backend.

The security configuration ensures the following key aspects:

1. Authentication and Authorization:

Authentication is handled using Basic Authentication and Form Login mechanisms. Users are required to provide valid credentials to gain access to the system. These credentials are verified against the database.

Authorization controls what resources users can access based on their roles. Users are assigned roles such as ADMIN, DOCTOR, PATIENT, or PHARMACIST. Role-based restrictions ensure that sensitive operations are accessible only to authorized personnel. For example:

- **Doctors** can access schedules, manage appointments, and issue prescriptions.
- **Patients** can view and manage their appointments and prescriptions.
- **Pharmacists** can access prescription details for fulfillment.
- **Admins** can have broader access to manage different entities within the hospital ERP system.

2. Spring Security Configuration:

The **ProjectSecurityConfig** class defines the security policies for the application. Key components of this configuration include:

- **Security Filter Chain** (`SecurityFilterChain`): Configures the security settings of the HTTP requests.
- **Session Management**: The application uses **session creation policies** (`SessionCreationPolicy.ALWAYS`), which helps in maintaining user sessions across multiple requests.
- **CSRF Protection**: The **Cross-Site Request Forgery (CSRF)** protection is managed using a `CookieCsrfTokenRepository`. Some endpoints are explicitly exempted from CSRF protection to facilitate operations like registration, contact messages, file upload/download, and password reset (`ignoringRequestMatchers`).
- **CORS Configuration**: **Cross-Origin Resource Sharing (CORS)** settings are defined to allow requests from the frontend running on `http://localhost:4200`. This configuration specifies allowed origins, methods, and headers to ensure secure communication between the frontend and backend.

3. CSRF Token Handling:

- CSRF tokens are used to protect against **cross-site request forgery** attacks. The **CsrfCookieFilter** is added to ensure that CSRF tokens are available and used correctly during user interactions.
- The **CsrfTokenRequestAttributeHandler** sets the CSRF token attribute name to `_csrf` for consistency across the application.

4. Role-Based Endpoint Protection:

The `authorizeHttpRequests()` configuration enforces role-based access for each endpoint:

- I. **Public Endpoints** (permitAll()):
 - a. Endpoints like /register, /login, /contact, /notices, /confirmEmail, /resetEmailRequest, and /changePassword are available to all users without authentication, enabling new users to register, reset passwords, or contact the hospital without needing to be logged in.
- II. **Authenticated Endpoints** (authenticated()):
 - a. Endpoints such as /myAccount, /PatientAppointment, and /myPatientPrescription are accessible to users who are authenticated. These endpoints allow authenticated users to manage their accounts, view their appointments, and manage prescriptions.
- III. **Role-Specific Endpoints** (hasAnyRole()):
 - a. **Users** with roles like ADMIN, USER, DOCTOR, or PATIENT can access endpoints related to general actions, such as viewing their appointments or prescriptions.
 - b. **Doctors** (ROLE_DOCTOR) can access endpoints like /DoctorsSchedule, /UpdateAppointment, and /createPrescription for managing schedules, updating appointments, and issuing prescriptions.
 - c. **Pharmacists** (ROLE_PHARMACIST) have access to endpoints for searching and fulfilling prescriptions (/SearchForPrescription, /updatePrescription).
 - d. **Patients** (ROLE_PATIENT) can book and manage their appointments via endpoints like /PatientAppointment, /updateAppointment.
5. **Password Encoding:**
 - a. Passwords are encoded using **BCrypt** (BCryptPasswordEncoder) before being stored in the database. BCrypt hashing ensures that even if the database is compromised, passwords remain secure and cannot be easily decrypted.
6. **Session and Cookie Management:**
 - a. The security configuration ensures sessions are managed efficiently with SessionCreationPolicy.ALWAYS, allowing the system to maintain session information across multiple requests for authenticated users.
 - b. Cookies are used for storing CSRF tokens (CookieCsrfTokenRepository.withHttpOnlyFalse()), which helps mitigate CSRF attacks while allowing the client-side JavaScript to access the CSRF token for subsequent requests.
7. **HTTP Security Customization:**
 - a. **Form Login and HTTP Basic Authentication:**

Both **form-based login** and **HTTP Basic Authentication** are enabled by default (formLogin(Customizer.withDefaults()) and httpBasic(Customizer.withDefaults())).

Form-based login provides a simple and user-friendly interface for authentication, while HTTP Basic Authentication is useful for testing purposes or for machine-to-machine communication.
 - b. **CORS:**

The CORS policy allows requests from the frontend hosted at http://localhost:4200, including credentials (e.g., session cookies) to ensure a smooth flow of data between the Angular frontend and the backend.

All HTTP methods are allowed, providing flexibility for the frontend to perform CRUD operations as needed.

8. **Filters and Handlers:**

- a. **CsrfCookieFilter:** A custom filter that ensures CSRF tokens are included in the HTTP response as cookies, making it easier for client-side frameworks like Angular to use these tokens in subsequent requests.
- b. **BasicAuthenticationFilter:** The custom CsrfCookieFilter is added after BasicAuthenticationFilter to make sure authentication processes are completed before handling CSRF tokens.

Summary

The **security configuration** of the Hospital ERP System is designed to safeguard sensitive healthcare information, ensure compliance with best practices in authentication and authorization, and protect against common security vulnerabilities such as **CSRF** attacks. The use of **Spring Security**, **JWT tokens**, and **role-based access control** ensures that only authorized users can access specific resources, thus maintaining data integrity and privacy. Additionally, **password encryption**, **session management**, and proper **CORS policies** contribute to a secure and efficient application that adheres to industry standards, providing a safe environment for both patients and healthcare professionals.

7 Conclusion and Future Improvements

7.1 Conclusion

In conclusion, the hospital ERP system developed here represents a comprehensive solution to the complex challenges of managing hospital operations. By integrating core functions like appointment scheduling, prescription management, and secure user authentication, the system significantly enhances the efficiency and coordination of healthcare processes.

One of the key strengths of this system is its foundation on Spring, a widely adopted framework in the software industry, known for its versatility and scalability. Spring is used by leading companies around the world due to its modular architecture, dependency injection, and powerful security features, making it ideal for building enterprise-grade applications. The use of this framework ensures that the system is not only secure and maintainable but also future-proof, with a vast ecosystem of tools and libraries supporting ongoing development.

Through streamlined workflows and improved data management, the ERP system enables hospital staff to focus on delivering high-quality patient care while minimizing administrative overhead. It provides a scalable solution capable of growing with the hospital's needs, and its robust, industry-standard architecture ensures reliability and long-term viability. Overall, this system stands as a vital technological advancement for healthcare institutions aiming to optimize their operations and enhance patient care.

7.2 Future Improvements

While the current version of the hospital ERP system meets the core needs of hospital management, several enhancements can be made to further optimize performance, enhance patient care, and improve overall hospital operations. Below are some key areas for future improvements:

1. **Billing and Invoicing**
The addition of a billing and invoicing module would automate the generation of invoices for medical services, simplifying financial management for both patients and the hospital. Integration with payment gateways could further streamline the process by enabling digital payment options.
2. **Medical Records Management**
An electronic medical records module could be integrated to store comprehensive patient medical histories. This would allow doctors and staff to access, update, and share patient information securely, improving continuity of care and compliance with healthcare regulations.
3. **Lab Test Management**
A laboratory test management feature could automate the ordering, tracking, and results reporting of lab tests. This would help ensure timely communication between doctors, lab technicians, and patients, improving efficiency in the diagnostic process.
4. **Ward and Bed Allocation**
The addition of ward and bed allocation management would assist hospital staff in tracking room availability, patient admissions, and bed assignments in real-time, helping optimize resource utilization, especially during high-occupancy periods.
5. **Insurance and Claims Processing**
A module for handling insurance and claims would streamline the process of filing and tracking insurance claims, reducing administrative overhead. Integration with insurance providers would help ensure that claims are processed efficiently and that patients have a smooth billing experience.
6. **Reporting and Analytics**
Adding a reporting module to provide deeper insights into hospital performance, including patient care outcomes, financial metrics, and staff productivity, could help hospital administrators make data-driven decisions. Custom dashboards could allow each department to monitor key performance indicators.
7. **Staff Shift Scheduling**
Implementing a staff shift scheduling feature would optimize the allocation of human resources, ensuring adequate staffing levels across departments. Automated shift planning could help avoid scheduling conflicts and reduce administrative effort.
8. **Emergency Department Management**
A dedicated emergency department management module could improve response times and patient handling during critical situations. This feature would allow hospital staff to prioritize emergency cases, manage triage, and ensure efficient use of emergency department resources.
9. **Telemedicine Integration**
As telehealth becomes increasingly vital, integrating a telemedicine module would allow doctors to conduct virtual consultations, reducing the need for in-person visits while ensuring that patients receive care, especially in remote areas.
10. **SMS/Email Notifications for Patients and Staff**
Adding an automated notification system for appointment reminders, lab results, and

prescription updates could improve communication between patients and healthcare providers. SMS or email notifications could help reduce no-shows and improve patient engagement.

11. **Medical Device Integration**
Integration with medical devices such as patient monitors, ventilators, or diagnostic equipment would allow real-time data capture directly into the hospital ERP system. This would enable doctors to monitor patient vitals continuously and make informed decisions quickly.
12. **Financial Accounting and Auditing**
Adding financial accounting and auditing features would support the management of hospital finances, including budgets, payroll, and tax reporting. Automated audits could help ensure compliance with legal regulations and streamline financial oversight.
13. **Document Management (for scanned records and files)**
A document management system that supports the digitization and secure storage of paper-based records, such as consent forms, lab reports, and prescriptions, could help hospitals transition toward a paperless environment, improving data retrieval and compliance with regulatory requirements.
14. **Multi-location Support**
For hospital networks or institutions with multiple locations, introducing multi-location support would allow administrators to manage operations across different sites from a single system. This could include centralized patient records, financial management, and resource allocation.
15. **Mobile Application Development**
A mobile application would provide convenient access for hospital staff and patients. Doctors could manage appointments, access records, and write prescriptions from their mobile devices, while patients could book appointments and receive health updates via the app.
16. **AI and Machine Learning for Predictive Analytics**
Leveraging AI and machine learning could improve predictive analytics for patient admissions, resource allocation, and even early diagnosis based on patient data. These technologies could help healthcare providers identify trends, forecast patient demand, and optimize care.
17. **Integration with Wearable Devices**
Integrating data from wearable health devices such as smartwatches and fitness trackers could provide doctors with real-time data on patient vitals, aiding in long-term patient monitoring and early detection of health issues.

These future improvements aim to enhance the overall functionality, flexibility, and scalability of the hospital ERP system. Each proposed enhancement would contribute to streamlining hospital operations, improving patient care, and supporting the hospital's long-term strategic goals.

Bibliography

1. A Web-Based Application for Innovative Hospital Appointment Scheduling Using Neural Network. <https://doi.org/10.1109/IISA.2018.8633605>
2. Information Technologies for the Healthcare Delivery System. <https://scholarworks.rit.edu/theses/8573>
3. Security and Privacy Analysis of Mobile Health Applications: The Alarming State of Practice. <https://doi.org/10.1109/ACCESS.2018.2805344>
4. Computer-aided diagnosis and access on peoples' health status using web technology and mobile devices. https://doi.org/10.1007/978-3-642-04798-5_24
5. Blockchain Mutability: Challenges and Proposed Solutions. <https://doi.org/10.1109/TETC.2020.2978428>
6. Immutability and Decentralized Storage: An Analysis of Emerging Threats. <https://doi.org/10.1109/ACCESS.2019.2963248>