AINS Hospital Website Management System

23CSJ404: INTRODUCTION TO DATABASE SYSTEMS

PROJECT REPORT

Submitted By

A V Kiran (TKM23CS056)

Aadithya P K (TKM23CS002)

Abhinav R (TKM23CS007)

Sidharth Hariharan (TKM23CS132)

Aravind Kumar V (LTKM23CS149)

to

TKM College of Engineering (Govt. Aided and Autonomous)

in partial fulfillment of the requirements for the award of Bachelor of Technology

in Computer Science and Engineering

Department of Computer Science and Engineering

T.K.M College of Engineering, Kollam

March 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

TKM COLLEGE OF ENGINEERING, KOLLAM

CERTIFICATE

This is to certify that the report entitled **'AINS Hospital Website Management System'** submitted by **A V Kiran(TKM23CS056), Aadithya P K(TKM23CS002), Abhinav R (TKM23CS007), Sidharth Hariharan(TKM23CS132), Aravind Kumar V(LTKM23CS149)** to TKM College of Engineering (Govt. Aided and Autonomous affiliated to the APJ Abdul Kalam Technological University) in partial fulfillment of the B.Tech. degree in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or purpose.

Project Coordinator **Head of the Department**

DECLARATION

We hereby declare that the project report '**AINS Hospital Website Management System'**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Prof. Sherin M Wilson**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any other University.

A V Kiran

Aadithya P K

Abhinav R

Sidharth Hariharan

Aravind Kumar V

Place: Kollam

Date:15/04/25

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude to the Almighty and extend our sincere thanks to everyone who helped us complete this project successfully.

We would like to express our sincere gratitude to **Dr. Sajeeb R,** Principal, TKMCE, for providing us with all the necessary facilities and support for carrying out this project. We are extremely grateful to **Dr. Aneesh G Nath,** Associate Professor and Head of Department, Department of Computer Science and Engineering for their constructive guidance, advice, constant support, and technical guidance provided throughout the development of this project.

We would also like to express our sincere gratitude to our project coordinator, **Prof. Sherin M Wilson,** Assistant Professor, Department of Computer Science and Engineering, TKM College of Engineering, Kollam, for providing us with all the necessary facilities, guidance, and support throughout the project.

Our immense gratitude extends to all faculty members and technical staff in the Department of Computer Science and Engineering for their assistance and for providing us with the necessary facilities to complete the project. Finally, we thank our families and friends, whose encouragement and support contributed significantly to the successful fulfillment of this project.

A V Kiran
Aadithya P K
Abhinav R
Sidharth Hariharan
Aravind Kumar V

ABSTRACT

The AINS Hospital Website Management System is a web-based application developed to simplify and modernize the hospital appointment and consultation process. The objective of the project is to create an efficient platform for patients to book appointments, manage consultation histories, and track their real-time consultation status. The system allows a single user account to handle multiple patient profiles, making it convenient for families to manage healthcare collectively.

The scope of the project includes appointment booking, doctor consultation tracking, token queue management, and administrative control over doctor records and appointments. The methodology involves designing a user-friendly interface with secure login, dynamic appointment modules, and a real-time token update system managed by doctors. Administrators can manage doctor profiles and view appointments filtered by date or patient. Doctors can update the consultation status of each patient, which reflects in the patient's history, offering transparency and organized record-keeping.

Key findings include reduced manual effort in appointment scheduling, improved patient-doctor coordination, and enhanced user satisfaction through digital healthcare access. The system is currently built for a single hospital with one admin, but it is designed to be scalable for future multi-admin and multi-hospital extensions, contributing to a more efficient and transparent healthcare delivery process.

Table Of Contents

List Of Tables

List Of Figures

CHAPTER 1
INTRODUCTION

1.1 Project Introduction

Healthcare is a fundamental need, and with the growing demand for digital solutions in medical services, hospitals are increasingly turning toward technology to manage patient appointments and consultations. The ***AINS Hospital Website Management System*** is developed to bring efficiency, transparency, and ease of access to patients and hospital staff alike. Traditional appointment booking methods are often time-consuming and require physical presence, which this system aims to eliminate. By using this platform, patients can log in using their email credentials, manage multiple patient profiles under one account (ideal for families), and seamlessly book appointments with available doctors.

The platform maintains detailed appointment history for each patient, allowing users to keep track of past consultations. Doctors can log in to update the consultation status of their scheduled patients, and a live token system informs patients about their turn and current consultation progress. On the administrative side, the system empowers a single admin to add or update doctor profiles and view appointment logs filtered by date or patient.

This project brings together all stakeholders—patients, doctors, and administrators—on a single digital interface, offering a modern, centralized solution for managing hospital operations and enhancing the healthcare experience. It is designed to be user-friendly, secure, and adaptable to future expansions.

1.2 Background and Motivation

Hospitals often face challenges in maintaining accurate records, managing appointments, and coordination between departments. Traditional manual processes are not only time-consuming but also prone to human errors, delays, and miscommunication. With the rapid advancement in technology, there is a growing need for a system that ensures smooth, accurate, and secure management of hospital activities.

The motivation behind this project stems from the desire to improve the quality of healthcare services by introducing automation in appointment handling, patient data management, and administrative control. The system enables users to register and log in to manage multiple patient profiles under a single user account, facilitating a more convenient approach for families seeking healthcare for multiple members.

1.3 Objectives

The primary objective of the *Hospital Website Management System* is to provide an efficient, user-friendly, and digital solution to manage hospital consultations, appointments, and doctor-patient interactions. The system is designed to reduce manual efforts, save time, and improve the accessibility of healthcare services for patients and families.

Key objectives include:

1. **Patient Account Management:** Allow users to register and manage multiple patient profiles under a single account, simplifying family healthcare tracking.
2. **Online Appointment Booking:** Enable patients to book appointments with doctors through an intuitive interface, selecting date, time, and relevant details.
3. **Real-Time Token System:** Implement a live token tracking mechanism so that patients are aware of their position in the consultation queue and the current status of ongoing appointments.
4. **Consultation History:** Maintain detailed historical records of each patient's appointments and their consultation status.
5. **Doctor Dashboard:** Allow doctors to view scheduled appointments, mark patients as consulted, and update real-time data.
6. **Admin Control:** Provide a robust admin panel for managing doctor profiles, editing details, and reviewing appointment history based on date or patient.

Ultimately, the goal is to improve operational efficiency, reduce waiting times, and enhance transparency between patients, doctors, and administrators in a hospital environment.

1.4 Scope

The *Hospital Website Management System* is built to cover various aspects of hospital appointment and consultation management, with flexibility for future enhancements. The scope of the system spans across three main user roles—patients, doctors, and the administrator—each with unique functionalities.

For **patients**, the system offers:

1. Account creation and secure login using email and password.
2. Management of multiple patient profiles under one account.
3. Viewing past appointment records and consultation statuses.
4. Online appointment booking with preferred doctors.
5. Real-time token tracking on the day of consultation.

For **doctors**, the system allows:

1. Updating consultation statuses for each appointment.
2. Managing the current token in real-time to help patients monitor progress.

For the **admin**, the system provides:

1. Full control over doctor data including adding, editing, and removing doctors.
2. Viewing and filtering appointments by date or by patient.
3. Monitoring system usage and user interactions.

Currently, the system supports a **single hospital and one admin**, but the backend and frontend structures are designed to support **scalability**, enabling multi-admin and multi-hospital capabilities in future versions. This project focuses on enhancing healthcare service delivery through digital transformation, reducing manual errors, and improving user satisfaction.

CHAPTER 2
SYSTEM OVERVIEW

2.1 System Functions

The **AINS Hospital Website Management System** is developed with the objective of digitizing various core hospital activities, particularly focusing on patient-doctor interactions and administrative efficiency. The system offers a set of integrated and interactive functions designed to automate and simplify routine processes. The major system functions are as follows:

1. **User Registration and Authentication:**
   The system allows users to securely register and log in using an email and password. Authentication ensures that unauthorized users cannot access patient or doctor data. Each user has a unique ID, and the login credentials are encrypted for data security.

2. **Patient Profile Management:**
   Once a user is logged in, they can manage multiple patient profiles under a single account. This feature is especially useful for families where one member can book appointments for all dependents such as children, parents, or siblings.

3. **Doctor Discovery and Viewing:**
   Patients can browse through a list of doctors, view their specializations, check their availability by day and time, and read their qualifications and years of experience. This ensures transparency and helps users make informed choices.

4. **Real-Time Appointment Booking:**
   The system enables users to schedule appointments with available doctors based on their preferred date and time. Available slots are shown dynamically, and once booked, the appointment details are immediately reflected in the patient's dashboard.

5. **Token Generation and Queue Management:**
   Upon successful appointment booking, the system generates a unique token number for the patient. This token number determines the consultation order, helping the hospital manage patient flow efficiently.

6. **Consultation Status Updates:**
   Doctors can mark appointments as "Consulted" once a patient has been seen. Patients can then view their updated consultation status and history, including doctor feedback or comments if applicable.

7. **Administrative Management:**
   The admin dashboard provides access to a wide range of hospital operations such as adding or updating doctor profiles, managing doctor schedules, viewing patient records, and tracking overall appointment statistics.

These core functions work together to eliminate paperwork, reduce waiting time, enhance transparency, and improve the overall healthcare experience for both patients and staff.

2.2 Module Description

To organize and modularize the functionality of the system, it is divided into several interrelated modules. Each module is responsible for a distinct feature or process.

1. **Login Module:**
   This module provides secure registration and login features for users. It validates user credentials, handles login sessions, and provides a logout function to protect access to sensitive data.

2. **Patient Module:**
   Allows users to create and manage profiles for multiple patients under a single login. This is particularly useful for users who want to book appointments for family members.

It stores details like name, gender, date of birth, and contact information.

3. **Doctor Module:**
   Displays all available doctors along with their schedules, working days, and areas of specialization. The user can filter and select a doctor for booking appointments. This module also ensures that a maximum of 50 appointments can be made per doctor per day.

4. **Appointment Module:**
   Provides the interface for booking new appointments and viewing previous and upcoming ones. Patients can select a doctor and an available time slot. Doctors can update the appointment status after consultation, and patients can track this in their appointment history.During each appointment it automatically assigns a token number when an appointment is confirmed. This token helps manage the queue and ensures that patients are seen in order. It includes fields like token number, status (Waiting/Consulted), and appointment date.

5. **Admin Module:**
   Offers administrative control over the system. Admins can manage hospitals, add or update doctor details, modify their availability, and view appointment records and patient profiles. This module is essential for keeping the system updated and functional.

Each module is connected through well-defined relationships in the database, ensuring smooth information flow, minimal redundancy, and data integrity.

2.3 Operating Environment

The AINS Hospital Website Management System is designed to be platform-independent, lightweight, and adaptable to different hosting environments. It operates efficiently in both local and cloud-based server setups.

1. **Software Requirements:**
    a. **Operating System:** Windows 10 or later, Linux (Ubuntu preferred), or macOS
    b. **Frontend Technologies:** HTML5, CSS3, JavaScript
    c. **Backend Technologies:** PHP (Laravel framework optional) or Node.js
    d. **Database Server:** MySQL or MariaDB
    e. **Web Server:** Apache (with XAMPP/WAMP) or Node server
    f. **Browser Compatibility:** Google Chrome, Mozilla Firefox, Microsoft Edge, Safari
    g. **IDE:** Visual Studio Code, Sublime Text, or PHPStorm (optional)
    h. **Others:** Git for version control, Postman for API testing (optional)
2. **Hardware Requirements:**
    a. **Processor:** Intel Core i3 or above (or AMD equivalent)
    b. **RAM:** Minimum 4 GB (Recommended: 8 GB for better performance)
    c. **Storage:** 512 MB of free space for software installation and database

This setup ensures that the system is capable of running on most commonly available hardware and software infrastructures.

2.4 Implementation Constraints

Despite its strengths, the system has a few constraints due to the scope and scale of development. These limitations are as follows:

1. **Web-Only Access:**
   The current version is limited to web browsers. Mobile applications or native apps are not yet developed. This limits usability for users who prefer mobile platforms.
2. **Limited User Roles:**
   Only four user types are implemented: User, Patient, Doctor, and Admin. Additional roles like receptionist, billing staff, or lab technician are not currently supported.
3. **No Billing or Payment Integration:**
   The current version does not include modules for handling billing, invoicing, or online payment. These can be added in future enhancements.

4. **Security Constraints:**
   Basic authentication and encrypted passwords are implemented, but features such as Two-Factor Authentication (2FA), CAPTCHA, or SSL certification for production deployment are not part of this version.
5. **No Multi-language Support:**
   The interface currently supports only English, which may be a limitation in multilingual regions.

Even with these constraints, the system has been designed in a modular and scalable manner, ensuring that additional features and modules can be integrated in future releases without major redesign.

CHAPTER 3

SYSTEM DESIGN

3.1 ER Diagram

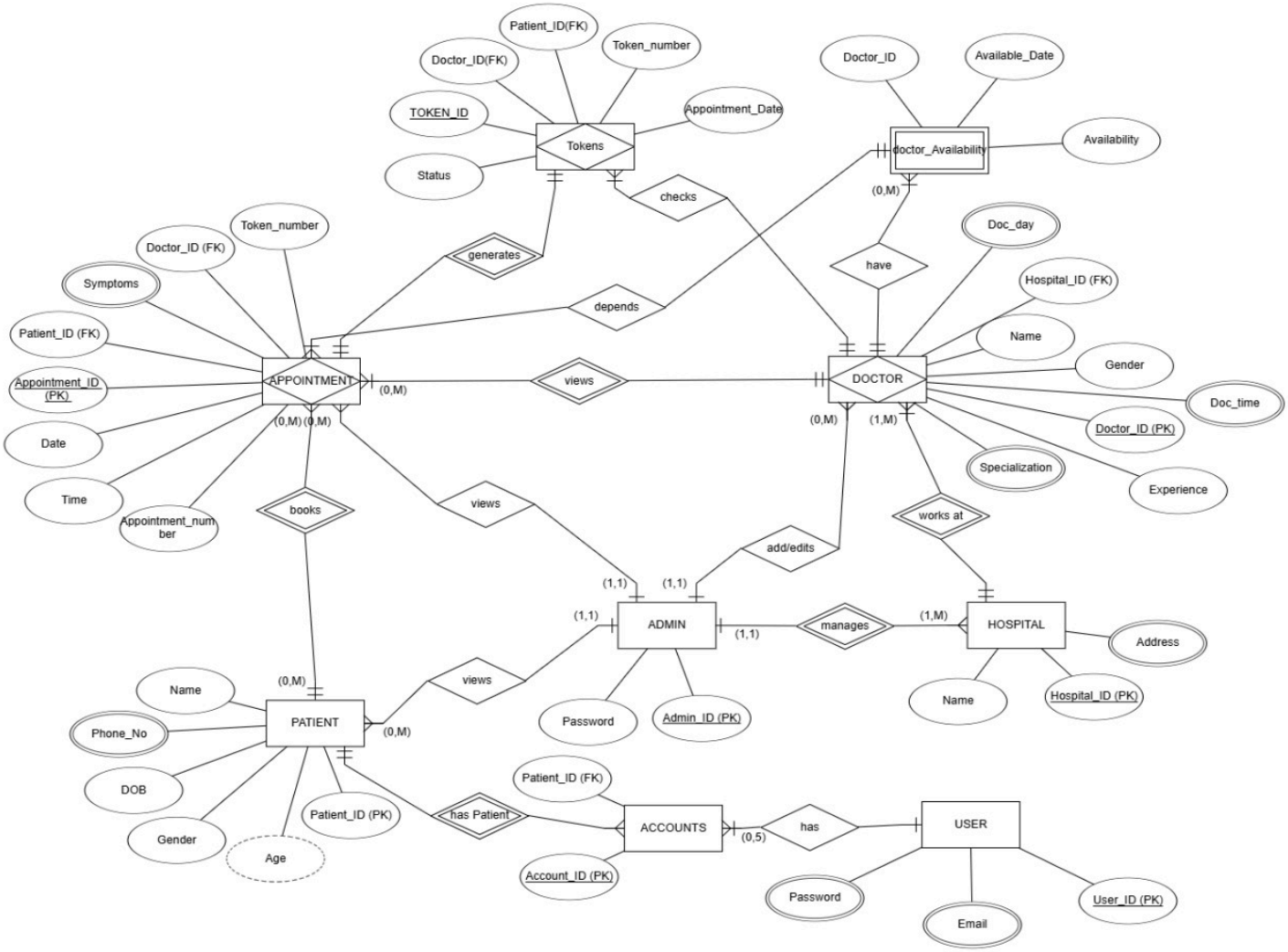

Fig 3.1 ER Diagram

1. **ENTITIES AND ATTRIBUTES**
    a. **DOCTOR**
        i. Doctor_ID (PK): Unique identifier for each doctor.
        ii. Name: Full name of the doctor.

iii. Gender: Gender of the doctor.
iv. Specialization: Medical specialty (e.g., cardiology, orthopedics).
v. Experience: Years of experience.
vi. Doc_day: Days available for appointment.
vii. Doc_time: Available time slots.
viii. Hospital_ID (FK): References the hospital the doctor works at.

b. **PATIENT**
   i. Patient_ID (PK): Unique identifier for each patient.
   ii. Name: Full name of the patient.
   iii. Phone_No: Contact number.
   iv. DOB: Date of birth.
   v. Gender: Gender of the patient.
   vi. Age: Age of the patient (possibly derived from DOB).

c. **APPOINTMENT**
   i. Appointment_ID (PK): Unique identifier for the appointment.
   ii. Doctor_ID (FK): Doctor assigned for the appointment.
   iii. Patient_ID (FK): Patient who books the appointment.
   iv. Token_number: Token assigned for the appointment.
   v. Symptoms: Patient's symptoms or reason for visit.
   vi. Date: Date of the appointment.
   vii. Time: Time of the appointment.
   viii. Appointment_number: Unique number associated with the appointment.

d. **HOSPITAL**
   i. Hospital_ID (PK): Unique identifier for the hospital.
   ii. Name: Name of the hospital.
   iii. Address: Location/address of the hospital.

e. **ADMIN**
   i. Admin_ID (PK): Unique identifier for the admin.
   ii. Password: Admin password for access.

f. **TOKENS**
   i. TOKEN_ID: Unique token number.

ii. Token_number: Number assigned to patient visits.
iii. Patient_ID (FK): References the patient assigned to the token.
iv. Doctor_ID (FK): References the doctor the token is assigned to.
v. Appointment_Date: Date for the token/appointment.
vi. Status: Status of the token (e.g., active, used, cancelled).

g. **DOCTOR_AVAILABILITY**
   i. Doctor_ID: References the doctor.
   ii. Available_Date: Date the doctor is available.
   iii. Availability: Indicates if the doctor is available.

h. **USER**
   i. User_ID (PK): Unique identifier for users.
   ii. Email: Email address of the user.
   iii. Password: Password for user access

i. **ACCOUNTS**
   i. Account_ID (PK): Unique account ID.
   ii. Patient_ID (FK): References the patient.
   iii. Password: Account login password.

2. RELATIONSHIPS AND EXPLANATION

a. **views** (Between PATIENT and APPOINTMENT)
   i. **Cardinality**: One patient can view many appointments, and each appointment can be viewed by many patients.
   ii. **Explanation**: Patients can view available appointment slots, and appointments can be viewed by multiple patients (e.g., for time slot checking or token status).

b. **books** (Between PATIENT and APPOINTMENT)
   i. **Cardinality**: One patient can book many appointments, but each appointment is booked by one patient.
   ii. **Explanation**: Patients book appointments with doctors; this captures which patient booked which appointment.

c. **views** (Between DOCTOR and APPOINTMENT)
   i. **Cardinality**: One doctor can view many appointments, and each appointment can be viewed by one doctor.
   ii. **Explanation**: Doctors can see all appointments scheduled with them.

d. **depends** (Between TOKENS and APPOINTMENT)
   i. **Cardinality**: A token depends on one appointment, but each appointment can have many tokens.
   ii. **Explanation**: Tokens are generated for appointments; the token is issued only if the appointment exists.

e. **generates** (Between APPOINTMENT and TOKENS)
   i. **Cardinality**: Each appointment can generate many tokens.
   ii. **Explanation**: After booking an appointment, a token is generated to manage the queue.

f. **checks** (Between TOKENS and DOCTOR_AVAILABILITY)
   i. **Cardinality**: Each token checks one doctor's availability.
   ii. **Explanation**: Before assigning a token, the system checks if the doctor is available on that date.

g. **have** (Between DOCTOR and DOCTOR_AVAILABILITY)
   i. **Cardinality**: One doctor can have **many** availability records
   ii. **Explanation**: Each doctor has a schedule of availability across multiple days.

h. **views** (Between ADMIN and APPOINTMENT)
   i. **Cardinality**: One admin can view **many** appointments
   ii. **Explanation**: Admin can monitor or manage the appointments in the system.

i. **adds/edits** (Between ADMIN and DOCTOR)
   i. **Cardinality**: One admin can add or edit **many** doctors
   ii. **Explanation**: Admin is responsible for managing doctor records.

j. **manages** (Between ADMIN and HOSPITAL)
   i. **Cardinality**: One admin manages **one** hospital

ii. **Explanation**: Each hospital is managed by an admin (e.g., hospital profile, admin-level control).

k. **works at** (Between DOCTOR and HOSPITAL)

i. **Cardinality**: One hospital can have **many** doctors; each doctor works at **one** hospital

ii. **Explanation**: Doctors are affiliated with hospitals

l. **has Patient (Between ACCOUNTS and PATIENT)**

i. **Cardinality**: One account is linked to **one** patient

ii. **Explanation**: Each patient has an account for logging in and managing their appointments

m. **has** (Between USER and ACCOUNTS)

i. **Cardinality**: One user can have **many** accounts

ii. **Explanation**: This may allow multiple user profiles or patient roles under a single user (e.g., family accounts).

RELATIONSHIP SCHEMA

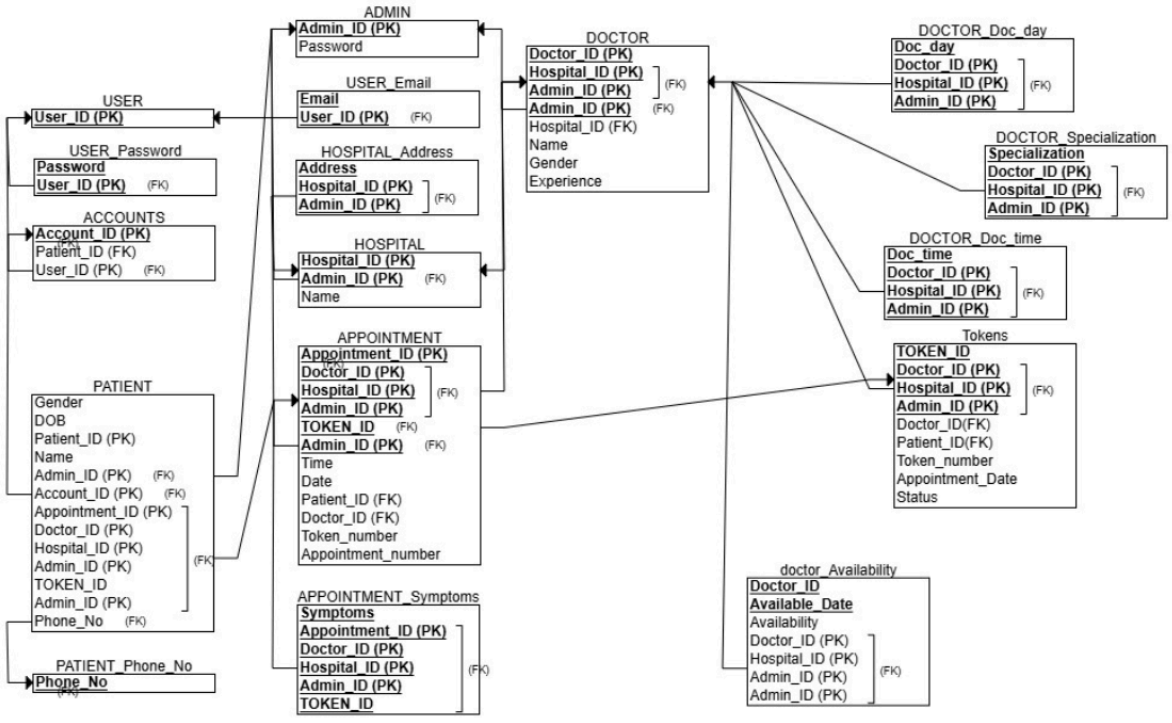

Fig 3.2 Relationship Schema

1. **USER**
    a. Attributes: User_ID (PK)
    b. Description: Represents a system user, possibly a patient or admin.
    c. Related Tables:
        i. USER_Password: Stores Password, tied via User_ID.
        ii. USER_Email: Stores email addresses.

2. **ACCOUNTS**
    a. Attributes: Account_ID (PK), Patient_ID (FK), User_ID (PK)
    b. Description: Links users to patients, each user has one account.
3. **PATIENT**
    a. Attributes: Patient_ID (PK), Name, Gender, DOB, Phone_No (FK), etc.
    b. Description: Contains patient profile data.
    c. Relationships:
        i. Linked to User_ID through Account_ID.
        ii. Connected to appointment via Appointment_ID.
        iii. Linked to token via TOKEN_ID.
4. **PATIENT_Phone_No**
    a. Attributes: Phone_No (PK)
    b. Description: Stores phone numbers for patients.
5. **ADMIN**
    a. Attributes: Admin_ID (PK), Password
    b. Description: Represents hospital administrators who manage hospitals and users.
6. **HOSPITAL**
    a. Attributes: Hospital_ID (PK), Name, Admin_ID (FK)
    b. Description: Represents a hospital entity.
    c. Relationship: Each hospital is managed by one admin.
7. **HOSPITAL_Address**
    a. Attributes: Address, Hospital_ID (PK)
    b. Description: Holds hospital address details.

8. **DOCTOR**
    a. Attributes: Doctor_ID (PK), Name, Gender, Experience, etc.
    b. Description: Doctor profiles.
    c. Foreign Keys: Hospital_ID, Admin_ID
9. **DOCTOR_Specialization**
    a. Attributes: Specialization, Doctor_ID (PK)
    b. Description: Contains doctor specialties.
10. **DOCTOR_Doc_day**
    a. Attributes: Doc_day, Doctor_ID (PK)
    b. Description: Represents days when the doctor is available.
11. **DOCTOR_Doc_time**
    a. Attributes: Doc_time, Doctor_ID (PK)
    b. Description: Represents the specific time slots for availability.
12. **doctor_Availability**
    a. Attributes: Doctor_ID, Available_Date, Availability
    b. Description: Full availability record of a doctor including date.
13. **APPOINTMENT**
    a. Attributes: Appointment_ID (PK), Time, Date, Token_number, etc.
    b. Foreign Keys: Doctor_ID, Hospital_ID, Patient_ID, Admin_ID, TOKEN_ID
    c. Description: Appointment bookings between patient and doctor.
14. **APPOINTMENT_Symptoms**
    a. Attributes: Symptoms, Appointment_ID (PK)
    b. Description: Records the symptoms reported during booking.
15. **Tokens**
    a. Attributes: TOKEN_ID (PK), Doctor_ID, Patient_ID, Appointment_Date, Token_number, Status
    b. Description: Generated after an appointment is booked to manage queue.

3.2 Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.

1. Normalization Principles:
    a. **First Normal Form (1NF):**
        i. Ensures that each column contains atomic (indivisible) values.
        ii. No repeating groups or arrays.
    b. **Second Normal Form (2NF):**
        i. Achieved when the table is in 1NF and all non-key columns are fully dependent on the entire primary key.
        ii. Eliminates partial dependencies in composite keys.
    c. **Third Normal Form (3NF):**
        i. Achieved when the table is in 2NF and all columns are directly dependent on the primary key only.
        ii. No transitive dependencies (non-key → non-key).
2. Table-wise Normalization Check:
    a. Accounts
        i. **1NF:** Yes (Atomic values).
        ii. **2NF:** Yes (Only one primary key – Patient_ID).
        iii. **3NF:** Yes (No transitive dependencies).
    b. Admin
        i. **1NF:** Yes (Atomic data, encrypted password).
        ii. **2NF:** Yes (No partial dependency).
        iii. **3NF:** Yes (All non-key attributes depend on primary key admin_id).
    c. Appointment
        i. **1NF:** Yes (No multi-valued fields).
        ii. **2NF:** Yes (Token and Appointment Number are dependent on composite key Doctor_ID + Date).
        iii. **3NF:** Yes (No indirect dependencies).

d. Doctor
   i. **1NF:** Yes.
   ii. **2NF:** Yes.
   iii. **3NF:** Yes (although doc_time and doc_day could be abstracted into a separate schedule table if more detailed logic is introduced).

e. Doctor_availability
   i. **1NF:** Yes.
   ii. **2NF:** Yes (composite PK: Doctor_ID + Available_Date).
   iii. **3NF:** Yes.

f. Hospital
   i. **1NF:** Yes.
   ii. **2NF:** Yes.
   iii. **3NF:** Yes (no derived data or transitive dependencies).

g. Patient
   i. **1NF:** Yes.
   ii. **2NF:** Yes (no composite key).
   iii. **3NF:** Yes.

h. Tokens
   i. **1NF:** Yes.
   ii. **2NF:** Yes (each token is atomic and linked with unique Token_ID).
   iii. **3NF:** Yes.

i. Users
   i. **1NF:** Yes.
   ii. **2NF:** Yes.
   iii. **3NF:** Yes.

3. **Normalization Analysis**
   a. **1st Normal Form (1NF):**
      i. Each table has a defined primary key and there are no repeating groups or arrays. For example, in the Patient table, each record is unique based on Patient_ID.
   b. **2nd Normal Form (2NF)**

i. All non-key attributes are fully functionally dependent on the primary key. For instance, in the Appointment table, all fields depend on Appointment_ID.

c. **3rd Normal Form (3NF)**:

i. No transitive dependencies are present. Attributes in the patient table (Name, Gender, DOB, Age) are solely dependent on Patient_ID without relying on other non-key attributes.

Normalization was applied rigorously in this project to ensure that the hospital database remains organized, efficient, and scalable for future requirements.

3.3 Triggers

Database triggers are procedural code executed automatically in response to certain events on a specified table. Triggers help automate workflows and ensure real-time data accuracy.

1. **Trigger: before_appointment_insert**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `before_appointment_insert` BEFORE INSERT ON `appointment` FOR EACH ROW BEGIN
    DECLARE next_token INT;
    DECLARE available_slots INT;

    SELECT Availability INTO available_slots
    FROM doctor_availability
    WHERE Doctor_ID = NEW.Doctor_ID AND Available_Date = NEW.Date;

    IF available_slots IS NULL OR available_slots <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Doctor is fully booked or does not consult on this date.';
    END IF;

    SELECT IFNULL(MAX(Token_Number), 0) + 1
    INTO next_token
    FROM appointment
    WHERE Doctor_ID = NEW.Doctor_ID AND Date = NEW.Date;

    SET NEW.Token_Number = next_token;

    UPDATE doctor_availability
    SET Availability = Availability - 1
    WHERE Doctor_ID = NEW.Doctor_ID AND Available_Date = NEW.Date;
END
```

Fig 3.3.1 trigger 1

2. **Trigger: before_insert_doctor**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `before_insert_doctor` BEFORE INSERT ON `doctor` FOR EACH ROW BEGIN
    DECLARE new_id INT;

    SELECT COALESCE(MAX(CAST(SUBSTRING(Doctor_ID, 4, 3) AS UNSIGNED)), 0) + 1
    INTO new_id FROM doctor WHERE Doctor_ID LIKE 'ADR%';

    SET NEW.Doctor_ID = CONCAT('ADR', LPAD(new_id, 3, '0'));
END
```

Fig 3.3.2 trigger 2

3. **Trigger: generate_admin_id**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `generate_admin_id` BEFORE INSERT ON `admin` FOR EACH ROW BEGIN
    DECLARE new_id VARCHAR(10);
    DECLARE random_part VARCHAR(6);

    -- Generate a random alphanumeric part (6 characters)
    SET random_part = CONCAT(
        CHAR(FLOOR(65 + (RAND() * 26))),  -- Random uppercase letter
        FLOOR(RAND() * 10),  -- Random digit
        CHAR(FLOOR(65 + (RAND() * 26))),
        FLOOR(RAND() * 10),
        CHAR(FLOOR(65 + (RAND() * 26))),
        FLOOR(RAND() * 10)
    );

    -- Generate final admin_id (e.g., ADM5G7K8)
    SET new_id = CONCAT('ADM', random_part);

    -- Assign the new ID
    SET NEW.admin_id = new_id;

    -- Encrypt password before storing
    SET NEW.password = AES_ENCRYPT(NEW.password, 'your_secret_key');
END
```

Fig 3.3.3 trigger 3

4. **Trigger: before_insert_patient**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `before_insert_patient` BEFORE INSERT ON `patient` FOR EACH ROW BEGIN
    DECLARE new_id INT;

    SELECT COALESCE(MAX(CAST(SUBSTRING(Patient_ID, 4, 3) AS UNSIGNED)), 0) + 1
    INTO new_id FROM patient WHERE Patient_ID LIKE 'AIN%';

    SET NEW.Patient_ID = CONCAT('AIN', LPAD(new_id, 3, '0'));
END
```

Fig 3.3.4 trigger 4

5. **Trigger: prevent_overbooking**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_overbooking` BEFORE UPDATE ON `doctor_availability` FOR EACH ROW BEGIN
    DECLARE booked_count INT;

    SELECT COUNT(*) INTO booked_count
    FROM appointment
    WHERE Doctor_ID = NEW.Doctor_ID AND Date = NEW.Available_Date;

    IF NEW.Availability < booked_count THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot reduce availability below already booked tokens.';
    END IF;
END
```

Fig 3.3.5 trigger 5

6. **Trigger: before_insert_appointment**

```
CREATE DEFINER=`root`@`localhost` TRIGGER `before_insert_appointment` BEFORE INSERT ON `appointment` FOR EACH ROW BEGIN
    DECLARE next_number INT;

    -- Get the maximum Appointment_Number for the same Patient_ID
    SELECT COALESCE(MAX(Appointment_Number), 0) + 1
    INTO next_number
    FROM appointment
    WHERE Patient_ID = NEW.Patient_ID;

    -- Assign the next available Appointment_Number
    SET NEW.Appointment_Number = next_number;
END
```

Fig 3.3.6 trigger 6

These triggers automate crucial backend tasks and maintain the integrity of the system data without manual intervention.

3.4 Data Description(Description of tables, fields, primary and foreign keys)

A well-structured relational database consists of multiple tables with relationships maintained via keys. Below is a summary of the primary tables used in the AINS Hospital Website Management System:

1. **accounts**
    a. Stores financial or login-related data linked to patients.
    b. Account_ID: Unique identifier for the account.
    c. Patient_ID: Foreign key linking to the patient table (Primary Key here).
2. **admin**
    a. Stores administrator login credentials.
    b. admin_id: Unique admin identifier (Primary Key).
    c. password: Encrypted admin password (varbinary).
3. **appointment**
    a. Stores appointment booking details.
    b. Appointment_ID: Unique ID for each appointment (Primary Key, Auto Increment).
    c. Date: Date of the appointment.
    d. Time: Time of the appointment.
    e. Symptoms: Symptoms reported by the patient.
    f. Doctor_ID: Foreign key referencing the doctor table.
    g. Patient_ID: Foreign key referencing the patient table.
    h. Token_Number: Queue number assigned to the patient.
    i. Appointment_Number: Internal reference number (used for sequence or count).
4. **doctor**
    a. Stores doctor profile information.
    b. Doctor_ID: Unique ID for each doctor (Primary Key).
    c. Name: Doctor's name.
    d. Specialization: Medical specialty (e.g., Cardiology).
    e. Experience: Years of experience.

f. Gender: Male/Female/Other.
g. Hospital_ID: Foreign key referencing the hospital table.
h. doc_time: Time slot(s) the doctor is available.
i. doc_day: Days of the week the doctor is available.

5. **doctor_availability**
   a. Tracks a doctor's availability on a specific date.
   b. Doctor_ID: Foreign key referencing the doctor table (Composite Primary Key).
   c. Available_Date: Date of availability (Composite Primary Key).
   d. Availability: Number of patients the doctor can take on that day.
6. **hospital**
   a. Stores hospital location and basic info.
   b. Hospital_ID: Unique hospital ID (Primary Key, Auto Increment).
   c. Name: Hospital name.
   d. Address: Hospital address.
7. **patient**
   a. Stores patient profile data.
   b. Patient_ID: Unique patient ID (Primary Key).
   c. Name: Patient's full name (Unique).
   d. Gender: Patient's gender.
   e. DOB: Date of birth.
   f. Age: Patient's age.
   g. Phone_No: Contact number.
8. **tokens**
   a. Tracks the token number and consultation status for a patient on a specific day.
   b. Token_ID: Unique identifier (Primary Key, Auto Increment).
   c. Doctor_ID: Foreign key linking to doctor.
   d. Patient_ID: Foreign key linking to patient.
   e. Token_Number: Queue number assigned to the patient.
   f. Appointment_Date: Date of appointment.
   g. Status: Pending or Consulted (default is Pending).
9. **users**

a. Stores credentials for users (patients/admin/staff).
b. User_ID: Unique user ID (Primary Key).
c. Email: Unique email address.
d. Password: User's password (usually hashed).

Each table is normalized and carefully linked using foreign keys to maintain referential integrity, ensuring a robust database design that supports all the operations of the hospital system.

CHAPTER 4

TOOLS AND TECHNOLOGIES USED

In order to develop a responsive, secure, and efficient hospital management system, a combination of modern tools and technologies has been employed. These tools span across database management, server-side and client-side development, and various utilities for debugging, version control, and interface testing. This chapter outlines the software components and development technologies utilized in the implementation of the **AINS Hospital Website Management System**.

4.1 DBMS software

1. **MySQL (Version 8.0)**[2]

   MySQL is an open-source relational database management system used for storing and managing the hospital data. It was chosen for its speed, flexibility, ease of use, and wide adoption in web-based applications.

   Key Features:

   a. Supports complex queries, joins, triggers, and transactions.
   b. Compatible with major programming languages like PHP and Java.
   c. Offers high performance and scalability for handling real-time patient and appointment data.

4.2 Development languages

Several programming and scripting languages were used for building the different components of the system:

1. **SQL (Structured Query Language):**

   Used to create and manage database schema, perform queries, create triggers, and enforce data relationships using primary and foreign keys.

2. **JavaScript:**
   JavaScript was used for client-side scripting to make the user interface interactive, validate forms, and handle real-time events like appointment slot display and token status updates.

4.3 Front-end tools

For the visual and interactive interface, the following front-end technologies were used:

1. **HTML5:**
   Provides the basic structure of the web pages such as forms, navigation bars, and content containers.
2. **CSS3:**
   Used for styling the HTML content, including layout designs, color themes, button styling, and responsive formatting for different screen sizes.
3. **Bootstrap 5:**
   A popular front-end framework used to create a mobile-first, responsive design with built-in UI components such as navbars, buttons, modals, and forms.

4.4 List of other Tools used

To support development, testing, and version control, the following tools were also used:

1. **Visual Studio Code (VS Code)**
   a. A lightweight and powerful source-code editor.
   b. Extensions like MySQL integration were used for real-time coding and testing.
2. **Git**
   a. A version control tool used to track code changes and collaborate with team members.
   b. Allows safe experimentation and rollbacks in case of errors.

3. **Canva**[5]

    a. Canva is an intuitive graphic design platform that enables users to create professional presentations, reports, and other visual content.

    b. It was used to design and structure the project's user interface mockups and visual assets.

4. **Google Docs**

    a. Google Docs is a cloud-based word processor that allows real-time collaboration and document editing.

    b. It was used for documentation, writing project reports, and collaboratively drafting code explanations and system design overviews.

5. **ERDPlus**

    a. ERDPlus is an online diagramming tool that allows users to design Entity-Relationship Diagrams (ERDs) and generate Relational Schemas.

    b. It offers an easy-to-use interface to create, visualize, and document database relationships and schema designs.

    c. For your hospital management system, ERDPlus would have been used to represent the structure of the hospital database, highlighting tables such as appointments, patients, doctors, and their relationships like foreign keys, primary keys, and other database constraints.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 MODULES

1. **LOGIN MODULE**
    a. It allows the user to register the account for appointment in the hospital, then login the user and proceed for appointment.
    b. It also provides a logout facility for the user logged in.

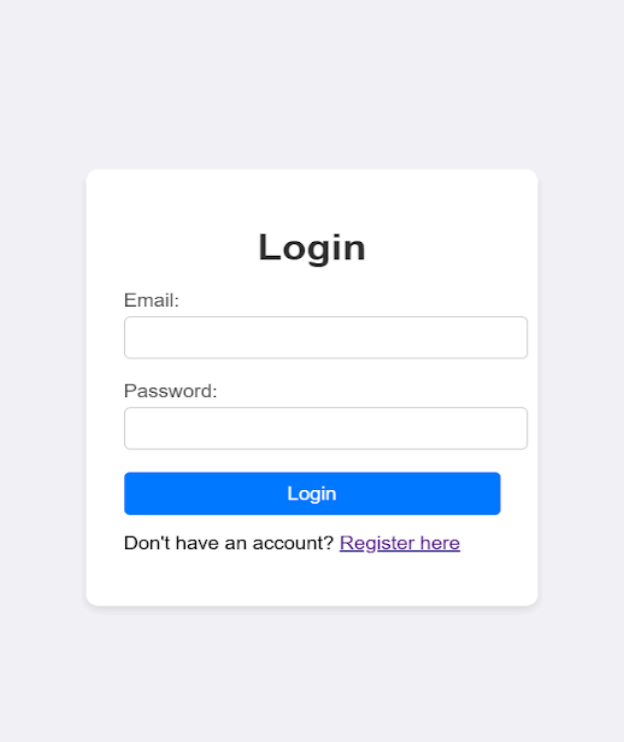

Fig 5.1.1 Login

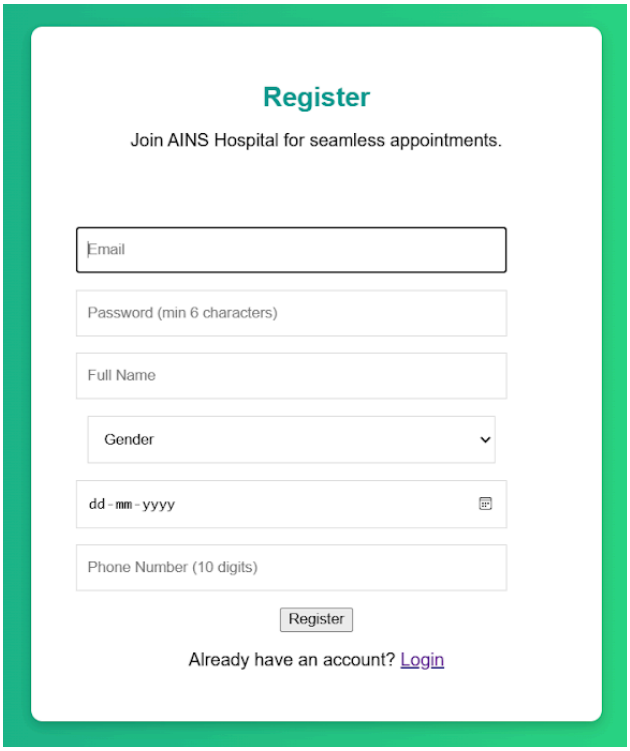

Fig 5.1.2 Register

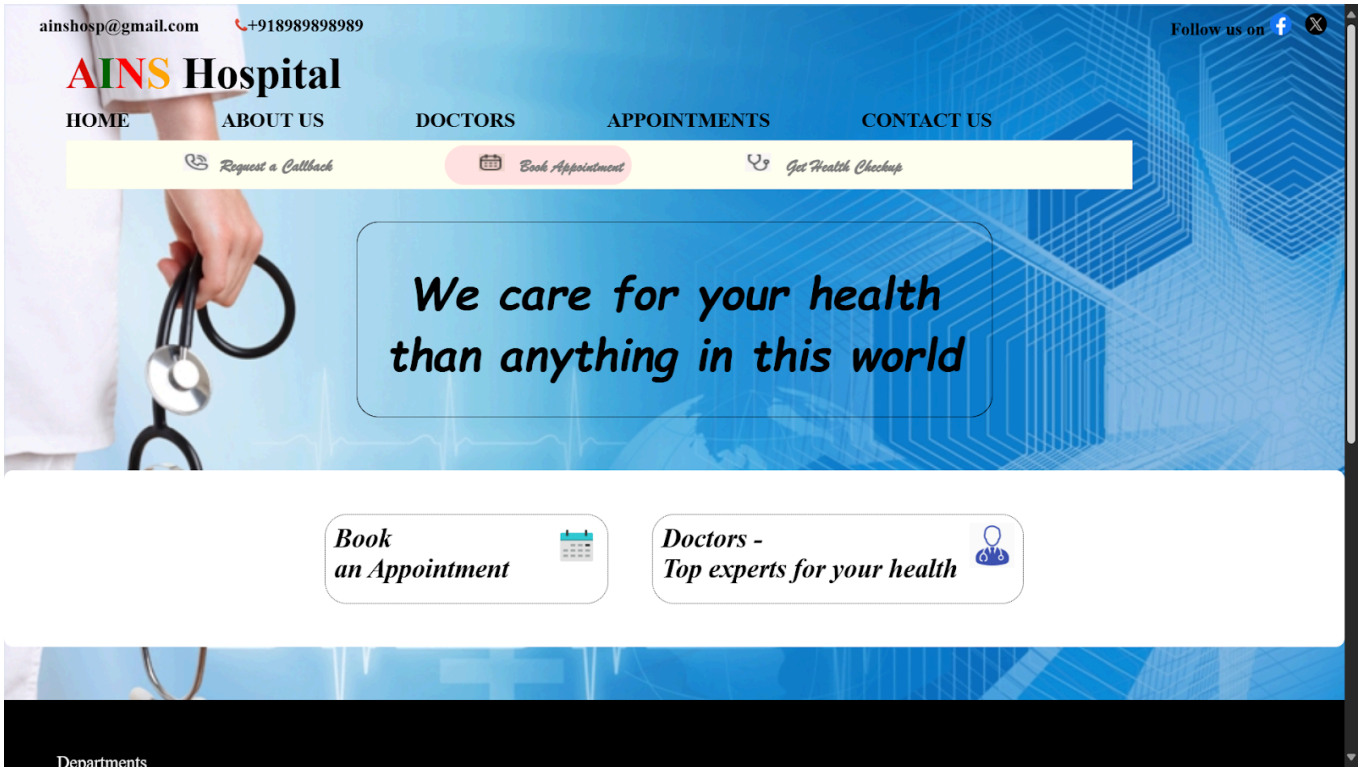

Fig 5.1.3 Home page of website

```
mysql> select * from users;
+---------+----------------------+--------------------------------------------------------------+
| User_ID | Email                | Password                                                     |
+---------+----------------------+--------------------------------------------------------------+
| 8RDSOF  | njan@gmail.com       | $2b$12$jbVA1JL2nSMsZjcTzaD6juw8/KAwQowpE1L2XknExlAgHOfg2yKdu |
| 9H07Q6  | kirancomtvm@gmail.com | $2b$12$wnVOuXfUU/Vf8Mpn/hZam.tmOdbWcQskxYw0waXLB2/p6bbFr0vqS |
| MR4339  | ad@gmail.com         | $2b$12$mjxHbq1qX1tRbYbyzpAqheHbGHUUzzKhPQrkGT8.5pf7/ARBAjtFS |
| QYV7TG  | 230274@tkmce.ac.in   | $2b$12$AR.p4SyhcUAEYhLcRTslPewo/a8MkpiHMWMhVWCUW9yLufhZ7oxZ2 |
+---------+----------------------+--------------------------------------------------------------+
```

Fig 5.1.4 users table

```
mysql> select * from accounts;
+------------+------------+
| Account_ID | Patient_ID |
+------------+------------+
| 9H07Q6     | AIN001     |
| 9H07Q6     | AIN002     |
| 9H07Q6     | AIN003     |
| MR4339     | AIN004     |
| 9H07Q6     | AIN005     |
| 8RDSOF     | AIN006     |
| QYV7TG     | AIN007     |
+------------+------------+
```

Fig 5.1.5 accounts table

2. **PATIENT MODULE**

   a. Here the patient can select the name of patient under the same user id to book appointment in case if family members logged in with single user credentials.

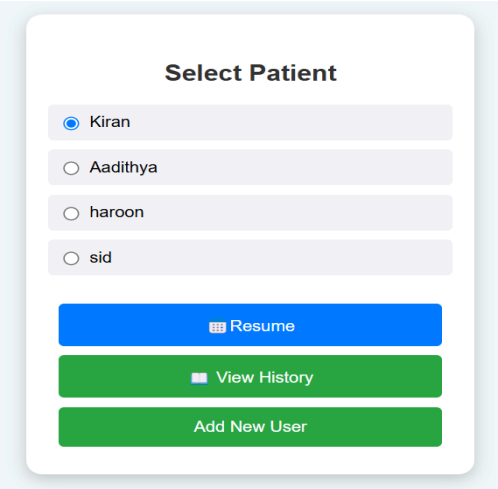

Fig 5.2.1 Select patient

```
mysql> select * from patient;
+------------+----------+--------+------------+-----+------------+
| Patient_ID | Name     | Gender | DOB        | Age | Phone_No   |
+------------+----------+--------+------------+-----+------------+
| AIN001     | Kiran    | Male   | 2005-10-11 |  19 | 8848974677 |
| AIN002     | Aadithya | Male   | 2006-01-19 |  19 | 9539344676 |
| AIN003     | haroon   | Male   | 2005-10-11 |  19 | 9847152403 |
| AIN004     | vasu     | Female | 2005-03-01 |  20 | 9539344676 |
| AIN005     | sid      | Male   | 2005-10-16 |  19 | 9446011856 |
| AIN006     | Akhil    | Male   | 2008-09-11 |  13 | 9633696501 |
| AIN007     | Akash    | Male   | 2006-03-03 |  19 | 9847152403 |
+------------+----------+--------+------------+-----+------------+
```

Fig 5.2.2 patient table

3. **DOCTOR MODULE**

   a. It has the details of doctors available for the appointment booking with 50 appointments daily and also allows booking appointments for selected doctor with confirmed appointments.

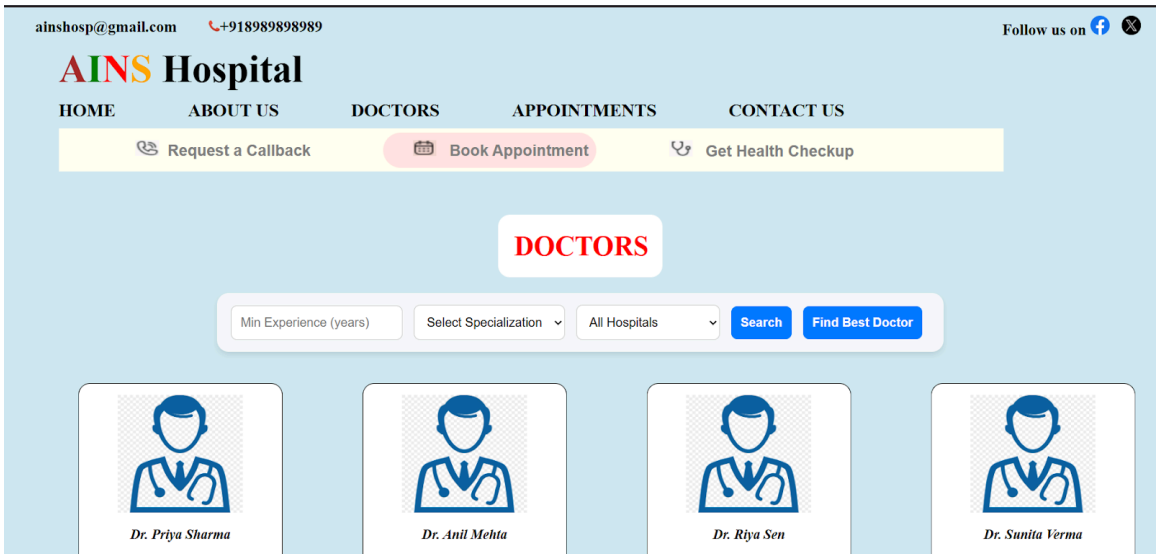

Fig 5.3.1 Doctors list

```
mysql> select* from doctor;
+-----------+------------------+-------------------+------------+--------+-------------+--------------+-------------+
| Doctor_ID | Name             | Specialization    | Experience | Gender | Hospital_ID | doc_time     | doc_day     |
+-----------+------------------+-------------------+------------+--------+-------------+--------------+-------------+
| ADR001    | Dr. Priya Sharma | Dermatology       |          8 | Female |           2 | 10:00-14:00  | Mon, Wed    |
| ADR002    | Dr. Anil Mehta   | Neurology         |         15 | Male   |           3 | 9:00 - 12:00 | Mon Tue Fri |
| ADR003    | Dr. Riya Sen     | Pediatrician      |         10 | Female |           1 | 09:30-13:30  | Tue, Thu    |
| ADR004    | Dr. Arjun Nair   | Orthopedic        |          7 | Male   |           4 | NULL         | NULL        |
| ADR005    | Dr. Sunita Verma | General Physician |         11 | Female |           5 | 11:00-15:00  | Mon, Fri    |
| ADR006    | Dr. Ramesh Yadav | Oncology          |         14 | Male   |           3 | 08:30-12:30  | Wed, Sat    |
| ADR007    | Dr. Meera Joshi  | Orthopedic        |          9 | Female |           2 | NULL         | NULL        |
| ADR008    | Dr. Sanjay Gupta | Dermatology       |         13 | Male   |           1 | 09.30-13.30  | Mon, Thu    |
| ADR009    | Dr. Alok Pandey  | General Physician |          6 | Male   |           4 | 14.00-18.00  | Wed, Fri    |
| ADR010    | Dr. Neha Kapoor  | Cardiology        |         12 | Female |           1 | N/A          | N/A         |
| ADR011    | Dr. Amit Sinha   | Neurology         |          9 | Male   |           2 | 09:00-13:00  | Tue, Fri    |
| ADR012    | Dr. Kavita Rao   | Dermatology       |          7 | Female |           3 | 11:30-16:00  | Wed, Sat    |
| ADR013    | Dr. Rajesh Khanna | Orthopedic       |         10 | Male   |           4 | 08:00-12:00  | Mon, Wed    |
| ADR014    | Dr. Alisha Verma | Ophthalmology     |         14 | Female |           5 | 12:00-17:00  | Tue, Thu    |
| ADR015    | Dr. Manish Tiwari | Ophthalmology    |         11 | Male   |           6 | 09:00-13:30  | Mon, Fri    |
| ADR016    | Dr. Ritu Sharma  | General Physician |          8 | Female |           7 | NULL         | NULL        |
| ADR017    | Dr. Suresh Mehta | General Physician |         13 | Male   |           8 | 10:30-14:30  | Wed, Sat    |
| ADR018    | Dr. Poonam Jain  | General Physician |          5 | Female |           9 | 14:00-18:00  | Tue, Thu    |
| ADR019    | Dr. Harish Patel | Ophthalmology     |          6 | Male   |          10 | 09:00-12:30  | Mon, Fri    |
| ADR020    | Dr. Vikram Sharma | Cardiology       |         16 | Male   |           1 | 08:00-12:00  | Mon, Thu    |
| ADR021    | Dr. Pooja Mehta  | Dermatology       |          9 | Female |           2 | NULL         | NULL        |
| ADR022    | Dr. Sanjay Kapoor | Neurology        |         12 | Male   |           3 | 10:00-14:00  | Tue, Fri    |
| ADR023    | Dr. Anjali Rao   | Pediatrician      |          6 | Female |           4 | NULL         | NULL        |
| ADR024    | Dr. Harsh Tandon | Orthopedic        |         14 | Male   |           5 | 09:30-13:30  | Wed, Sat    |
| ADR025    | Dr. Simran Kaur  | Dermatology       |         11 | Female |           6 | 11:00-15:00  | Mon, Wed    |
| ADR026    | Dr. Ramesh Iyer  | Dermatology       |         13 | Male   |           7 | 9:00 - 12:00 | Mon Tue Fri |
| ADR027    | Dr. Nidhi Sharma | Dermatology       |          7 | Female |           8 | 12:30-16:30  | Tue, Thu    |
| ADR028    | Dr. Rajat Verma  | General Physician |          8 | Male   |           9 | NULL         | NULL        |
| ADR029    | Dr. Swati Joshi  | Dermatology       |          5 | Female |          10 | 14:00-18:00  | Wed, Fri    |
+-----------+------------------+-------------------+------------+--------+-------------+--------------+-------------+
```

Fig 5.3.2 doctor table

```
mysql> select* from doctor_availability;
+-----------+----------------+--------------+
| Doctor_ID | Available_Date | Availability |
+-----------+----------------+--------------+
| ADR001    | 2025-04-16     |           50 |
| ADR001    | 2025-04-17     |           50 |
| ADR001    | 2025-04-18     |           50 |
| ADR002    | 2025-04-17     |           50 |
| ADR002    | 2025-04-18     |           50 |
| ADR002    | 2025-04-20     |           50 |
| ADR003    | 2025-04-18     |           50 |
| ADR003    | 2025-04-19     |           50 |
| ADR005    | 2025-04-17     |           50 |
| ADR005    | 2025-04-20     |           50 |
| ADR006    | 2025-04-18     |           50 |
| ADR006    | 2025-04-19     |           50 |
+-----------+----------------+--------------+
```

Fig 5.3.3 doctor_availability table

4. **APPOINTMENT MODULE**

   a. Here the patient can book appointments for the required doctor, view their appointment history and the consultation updates of last consulted token on a particular day.

   b. It also allows doctors to mark the consulted appointment after each consultation readily available for patients to see it on their appointment history.

This page helps you to book appointment for the available doctor the selected user on date.

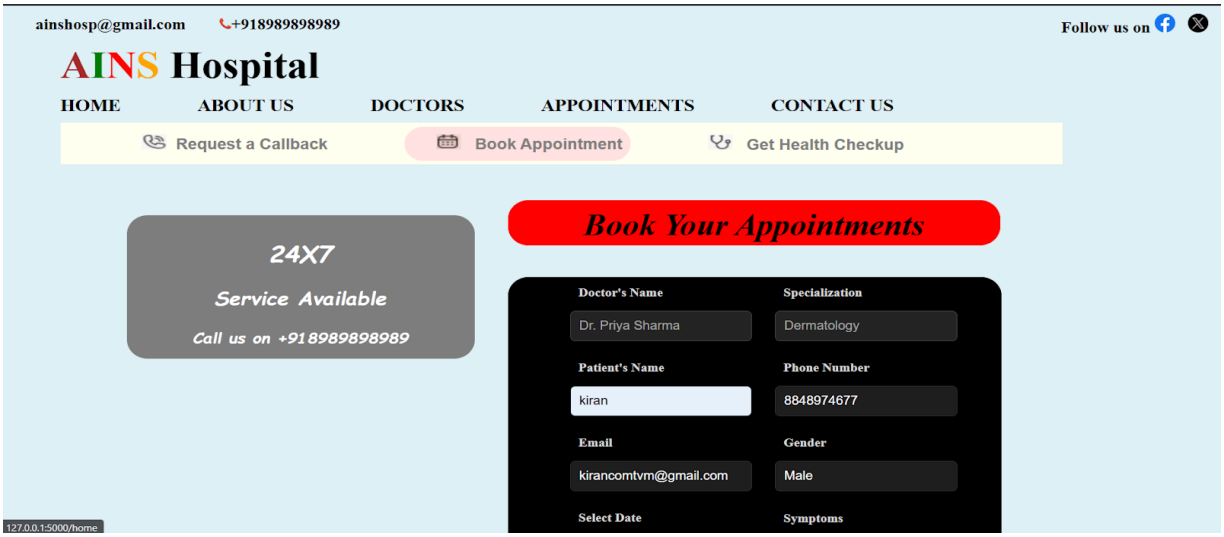

Fig 5.4.1 Appointment Page

Fig 5.4.2 Appointment History Page

This page is controlled by a doctor who consults patients and marks the consultation status of each patient and makes pending tokens available for patients to see.

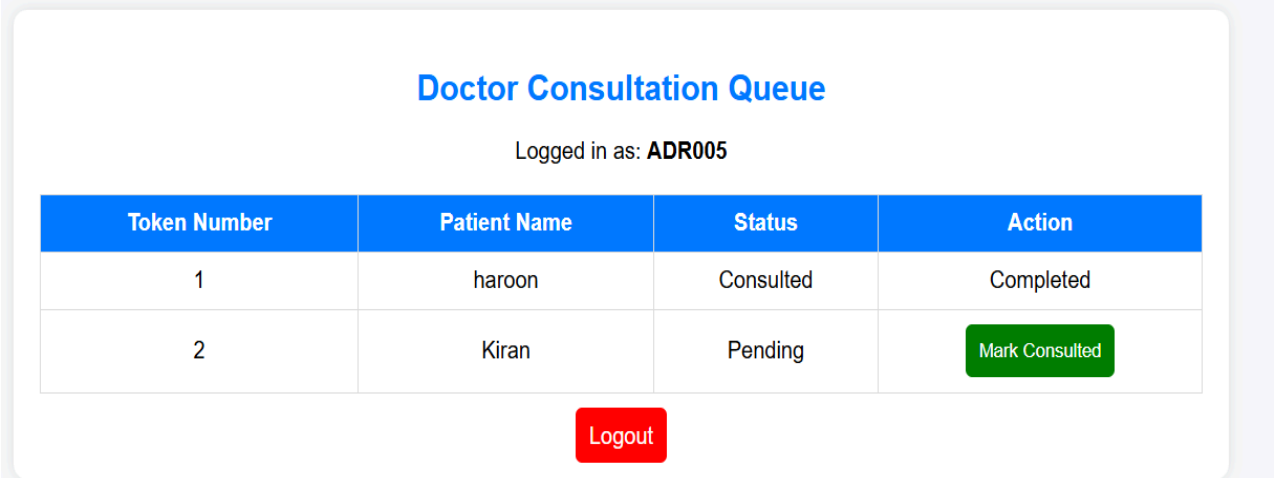

Fig 5.4.3 Doctor Consultation Management Page

```
mysql> select * from appointment;
+----------------+------------+----------+-----------------------------+-----------+------------+--------------+--------------------+
| Appointment_ID | Date       | Time     | Symptoms                    | Doctor_ID | Patient_ID | Token_Number | Appointment_Number |
+----------------+------------+----------+-----------------------------+-----------+------------+--------------+--------------------+
|              1 | 2025-03-27 | 14:17:53 | fever                       | ADR005    | AIN001     |            1 |                  1 |
|              2 | 2025-03-26 | 14:21:31 | wound                       | ADR005    | AIN003     |            1 |                  1 |
|              3 | 2025-03-26 | 14:25:57 | cough and cold              | ADR005    | AIN001     |            2 |                  2 |
|              9 | 2025-03-28 | 18:07:13 | Stomach pain and eyes yellow | ADR001    | AIN001     |            1 |                  3 |
|             10 | 2025-03-27 | 18:08:34 | Leg pain                    | ADR002    | AIN001     |            1 |                  4 |
|             11 | 2025-03-28 | 18:09:15 | Cough                       | ADR003    | AIN001     |            1 |                  5 |
|             12 | 2025-03-29 | 18:09:54 | Headache                    | ADR005    | AIN001     |            1 |                  6 |
|             13 | 2025-03-30 | 18:10:41 | Hair loss and body pain     | ADR006    | AIN001     |            1 |                  7 |
|             14 | 2025-03-27 | 18:11:18 | Pimples                     | ADR008    | AIN001     |            1 |                  8 |
|             15 | 2025-03-28 | 18:11:57 | Severe shivering            | ADR009    | AIN001     |            1 |                  9 |
|             16 | 2025-03-31 | 18:12:34 | Leg pain                    | ADR013    | AIN001     |            1 |                 10 |
+----------------+------------+----------+-----------------------------+-----------+------------+--------------+--------------------+
```

Fig 5.4.4 appointment table

```
mysql> select * from tokens;
+----------+-----------+------------+--------------+--------------+------------------+-----------+
| Token_ID | Doctor_ID | Patient_ID | Patient_Name | Token_Number | Appointment_Date | Status    |
+----------+-----------+------------+--------------+--------------+------------------+-----------+
|        1 | ADR005    | AIN003     | haroon       |            1 | 2025-03-26       | Consulted |
|        2 | ADR005    | AIN001     | Kiran        |            2 | 2025-03-26       | Pending   |
+----------+-----------+------------+--------------+--------------+------------------+-----------+
```

Fig 5.4.5 tokens table

```
mysql> select * from hospital;
+-------------+-------------------+-----------------------------+
| Hospital_ID | Name              | Address                     |
+-------------+-------------------+-----------------------------+
|           1 | AINS - Mumbai     | 123 Main Street, Mumbai     |
|           2 | AINS - Delhi      | 45 Green Avenue, Delhi      |
|           3 | AINS - Bangalore  | 78 Sunshine Road, Bangalore |
|           4 | AINS - Hyderabad  | Rajaji Nagar, Hyderabad     |
|           5 | AINS - Chennai    | College Road, Chennai       |
|           6 | AINS - Kolkata    | Park Street, Kolkata        |
|           7 | AINS - Pune       | FC Road, Pune               |
|           8 | AINS - Jaipur     | MI Road, Jaipur             |
|           9 | AINS - Chandigarh | Sector 17, Chandigarh       |
|          10 | AINS - Ahmedabad  | SG Highway, Ahmedabad       |
+-------------+-------------------+-----------------------------+
```

Fig 5.4.6 hospital table

5. **ADMIN MODULE**

   a. Here the admin can manage the hospital , add/edit doctor schedules, view patient details and appointments.

The login page provides a interface for admin to enter their credentials and access the system

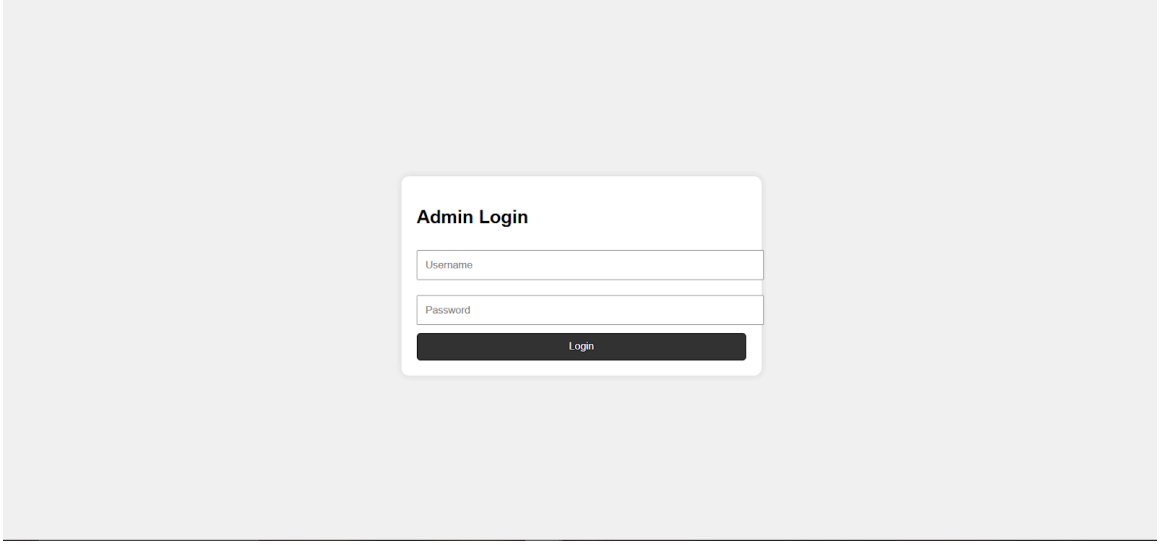

Fig 5.5.1 Admin Login page

The admin dashboard page allows authorized administrators to access and manage the system's backend functionalities.

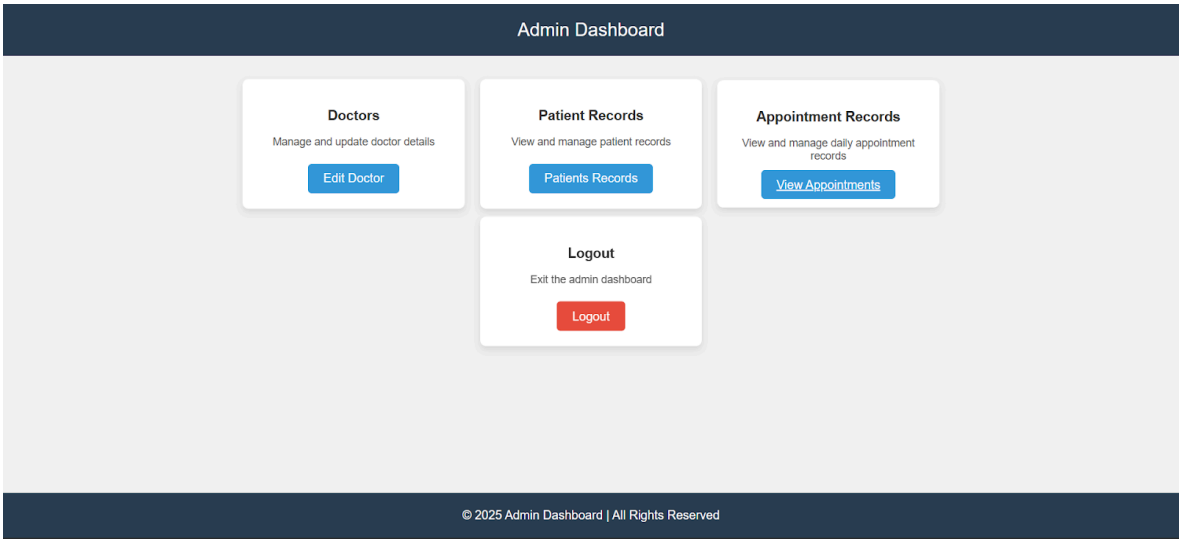

Fig 5.5.2 Admin dashboard

The Edit doctor page displays a comprehensive list of registered doctors, allowing admin to view their details and availability.

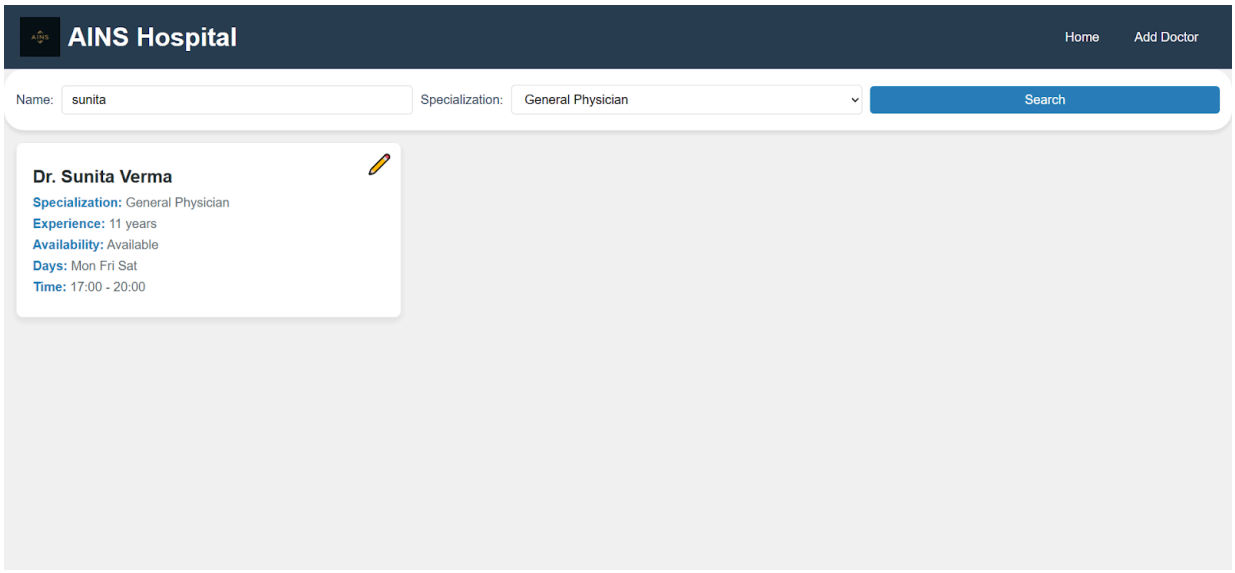

Fig 5.5.3 Edit/Add doctors page

The update doctor page enables authorized admin to modify and manage existing doctor's schedules in the system.

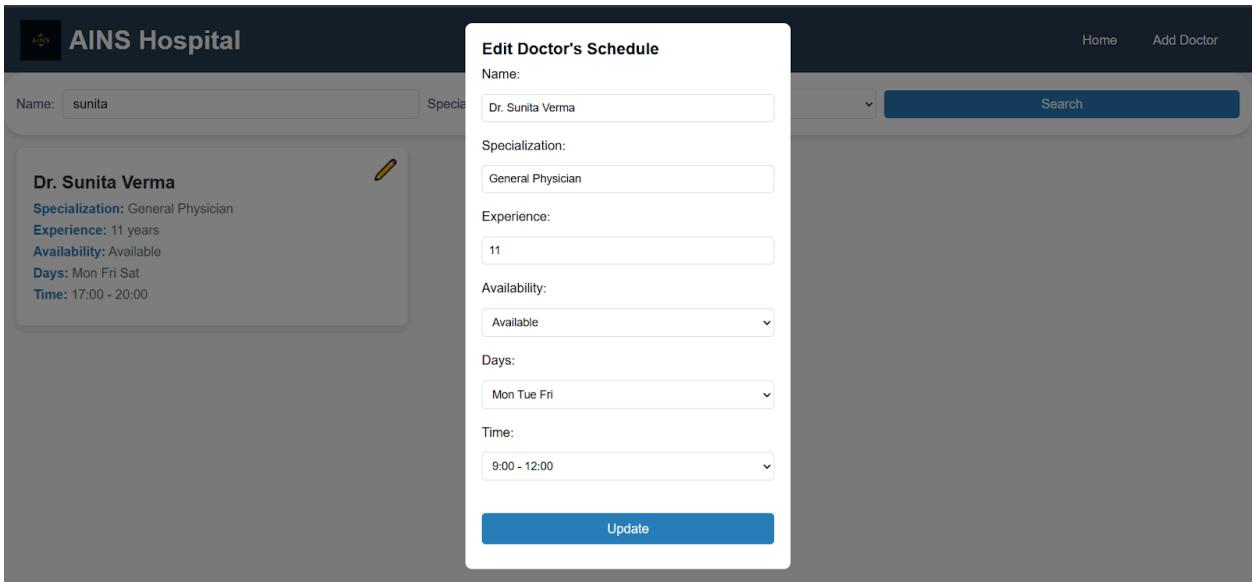

Fig 5.5.4 Edit doctor page

The add doctor page allows authorized users to input and register new doctor details into the system.

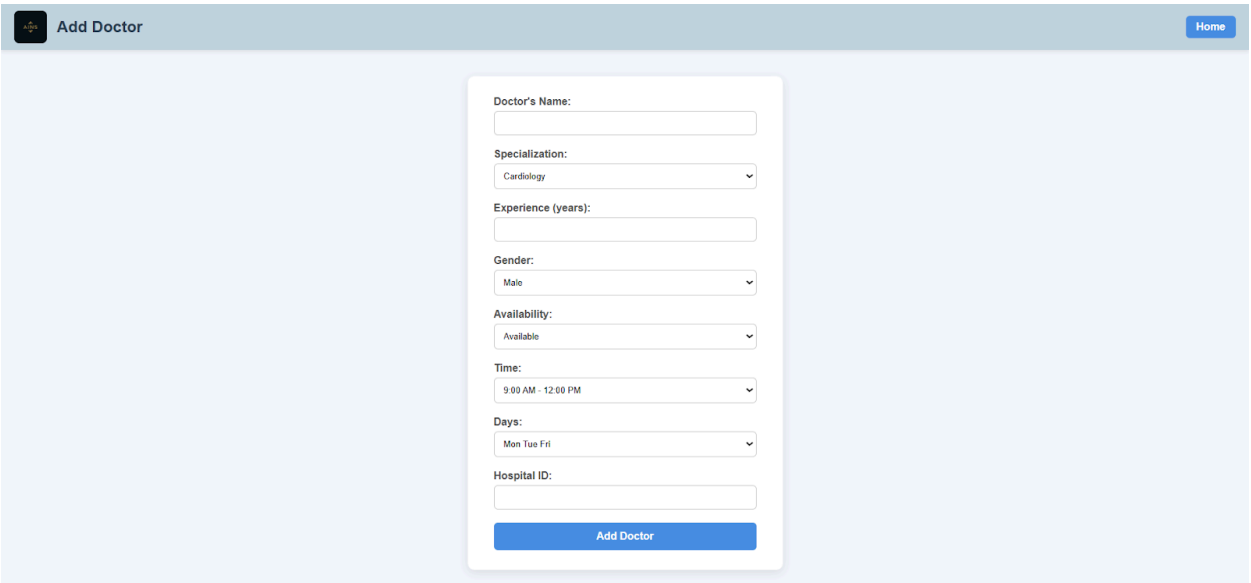

Fig 5.5.5 Add doctor

The view patient page provides a detailed overview of registered patients, allowing admin to access and review their information.

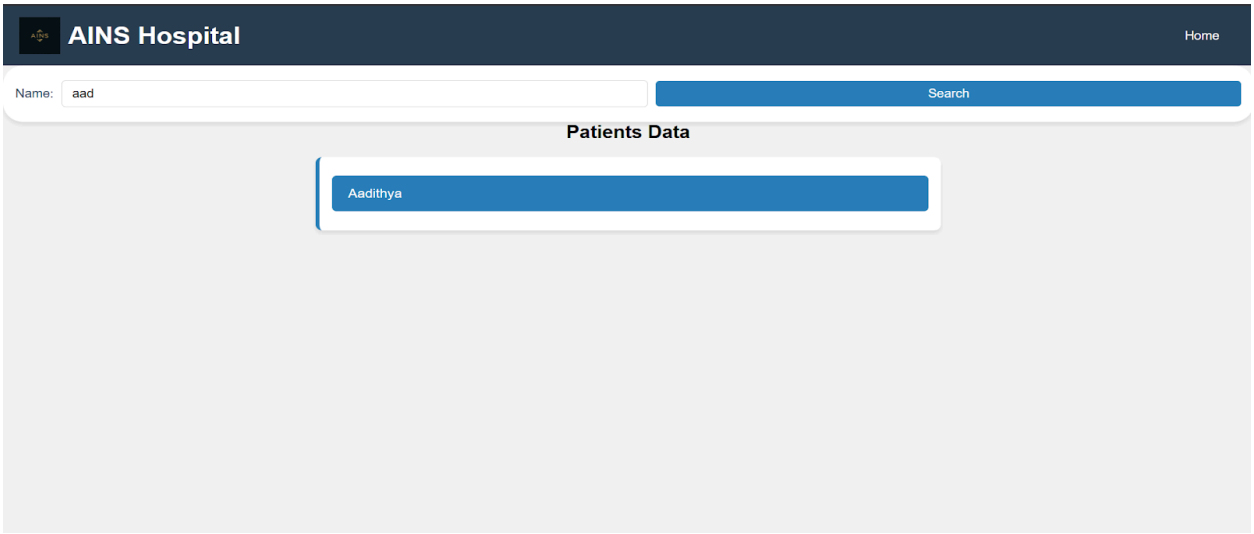

Fig 5.5.6 Manage patients

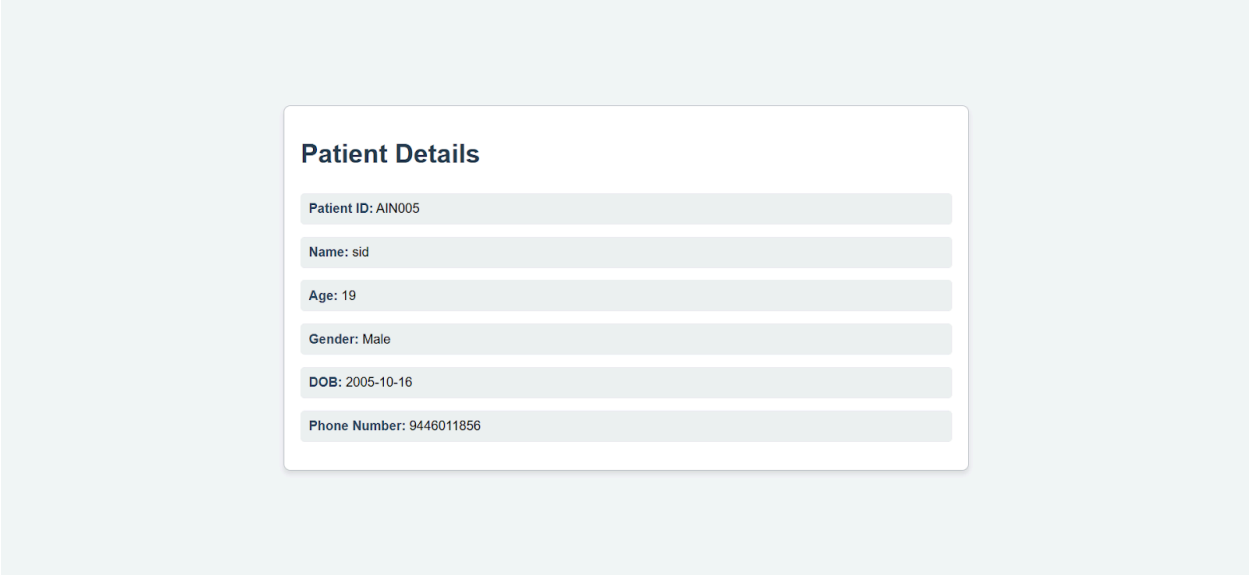

Fig 5.5.7 Patient details page

The appointment record page displays all appointments, allowing admin to view and track patient-doctor interactions.

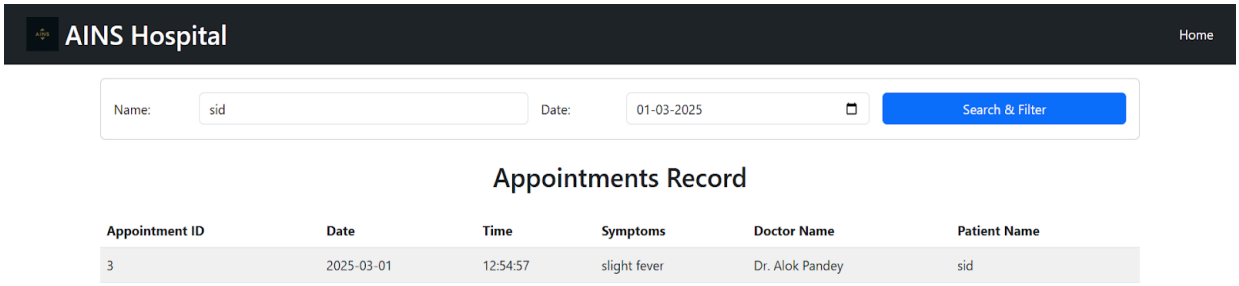

Fig 5.5.8 Appointment Records

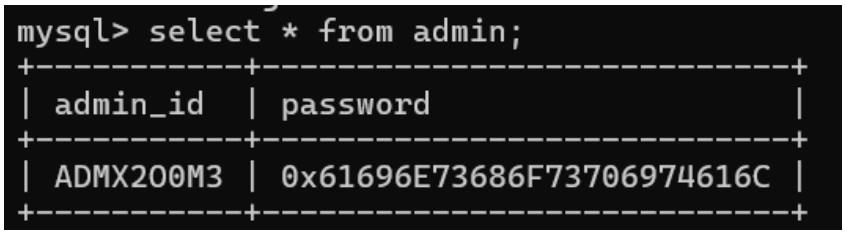

Fig 5.5.9 admin table

CHAPTER 6

TIMELINE AND TASK DISTRIBUTION

6.1 TIMELINE

S. No.	Milestone	Description	Date/Period
1	Initial Presentation	Submission of: Problem Identification, ER Diagram, Communication Plan, Ethics, Time Management, and Status.	31 December 2024
2	Requirement Analysis & Planning	Functional analysis, use-case definition, and system workflow planning.	20–24 December 2024
3	Finalized ER & Schema Diagram Submission	Submission of the finalized ER diagram and normalized database schema.	26 January 2025
4	System Design Phase	Frontend wireframes, backend architecture, and flow diagrams.	01–12 January 2025
5	Database Setup & Backend Configuration	Database creation, user authentication logic, and doctor-patient models.	15–25 January 2025
6	Frontend UI Implementation	Login, registration, patient dashboard, appointment UI screens.	27 Jan – 10 Feb 2025
7	Core Module Development	Appointment booking, token generation, appointment history, doctor dashboard logic.	11–22 February 2025
8	Admin Panel & Doctor Management	Admin CRUD for doctor details, appointment tracking features.	23–28 February 2025
9	Interim Presentation	Demonstration of implemented modules and SQL queries.	03 March 2025
10	Real-Time Queue & Token System Integration	Logic for displaying token numbers and doctor consultation status updates.	08–13 March 2025
11	Testing & Debugging	Functional and integration testing, fixing bugs, database query validation.	15–21 March 2025
12	Final Deployment & Hosting	Deploying website to hosting platform, adding security layers.	22–28 March 2025
13	Documentation & Final Report Preparation	Preparing project report, presentation, adding system screenshots and diagrams.	05–13 Apr 2025
14	Final Presentation, Demonstration & Submission	Complete system demonstration, viva, and final report submission.	15 April 2025

Fig 6.1 Timeline

6.2 TASK DISTRIBUTION

1. Login module : ARAVIND KUMAR V
2. Patient module : ABHINAV R
3. Doctor module : SIDHARTH HARIHARAN
4. Appointment module and backend development(for modules : login, patient, doctor, appointment) : A V KIRAN
5. Admin module and its backend development : AADITHYA P K

CHAPTER 7

CONCLUSION

1. Project Objective Achieved:
    a. The system fulfills its core objective of simplifying hospital appointment booking and consultation tracking through a centralized online platform.
2. User Account Flexibility:
    a. It allows a single user account to manage multiple patient profiles, making it especially useful for families to coordinate healthcare in one place.
3. Functional Scope Covered:
    a. Core functionalities like appointment booking, doctor consultation tracking, token queue management, and doctor record administration are fully implemented.
4. User-Friendly Design:
    a. The interface is intuitive and secure, ensuring ease of use for patients, doctors, and administrators while maintaining data privacy.
5. Real-Time Updates:
    a. The dynamic token system and real-time consultation status updates improve transparency and reduce patient waiting time uncertainty.
6. Doctor Interaction Module:
    a. Doctors can update patient consultation statuses instantly, which reflect in the patient's history, enhancing continuity and clarity in care delivery.
7. Administrative Capabilities:
    a. Admin users have complete control over managing doctor profiles and appointments, with features to view and filter records by patient or date.
8. Improved Healthcare Efficiency:
    a. The system reduces the need for manual appointment scheduling, cuts down on errors, and streamlines doctor-patient coordination.

9. Technology and Methodology:
    a. Implemented using modern web development practices, ensuring security, scalability, and responsive design across devices.
10. Scalability and Future Scope:
    a. Though initially developed for a single hospital with one admin, the architecture is designed to support multi-hospital and multi-admin models in the future.
11. Social Impact:
    a. Enhances patient satisfaction by offering accessible digital healthcare services and promoting a paperless, streamlined process.

CHAPTER 8

REFERENCES

1. Oracle. (2025). *Using Triggers*. Oracle® Database SQL Language Reference. Retrieved April 14, 2025 https://docs.oracle.com
2. MySQL. (Version 8.0). *MySQL 8.0 Reference Manual*. Retrieved April 14, 2025 https://dev.mysql.com/doc/refman/8.0/en/
3. W3Schools. (2025). *MySQL Tutorial*. Retrieved from https://www.w3schools.com
4. GeeksforGeeks. (2025). *DBMS and MySQL tutorials*. Retrieved April 14, 2025 https://www.geeksforgeeks.org
5. Canva. (2025). *Free Presentation and Design Templates*. Retrieved April 14, 2025 https://www.canva.com
6. ChatGPT (2025). *Project planning and code generation support*. OpenAI. Retrieved April 14, 2025 https://chat.openai.com
7. GitHub. (2025). *GitHub: Where the world builds software*. Retrieved April 15, 2025, from https://github.com

APPENDICES

Appendix I: Vision, Mission and Program Educational Objectives (PEOs)

VISION

To be a center of excellence imparting quality education in Computer Science and Engineering and transforming students to critical thinkers and lifelong teams capable of developing environment friendly and economically feasible solutions to real world problems

MISSION

- To provide a strong foundation in Computer Science and Engineering, prepare students for professional career and higher education, and inculcate research interest.
- To be abreast of the technological advances in a rapidly changing world.
- To impart skills to come up with socially acceptable solutions to real world problems, upholding ethical values.

PROGRAMME EDUCATIONAL OBJECTIVES(PEOs)

PEO 1: Excel in professional career by acquiring knowledge in mathematics, science and engineering and apply the knowledge in the design of hardware and software solution for challenging problems of the society

PEO 2: Pursue higher studies and research thereby engages in lifelong learning by adapting to the current trends in the area of Computer Science and Engineering

PEO 3: Ability to Provide socially acceptable and economically feasible computer oriented solutions to real world problems with teamwork, while maintaining environmental balance, quality and cognizance of the underlying principles of ethics.

Appendix II: Program Outcomes

1. **PO1 -Engineering Knowledge:** Apply fundamental principles of mathematics, science, and engineering to solve complex problems.
2. **PO2 -Problem Analysis:** Identify, formulate, and analyze complex engineering problems to reach conclusions based on solid data.
3. **PO3 -Design/Development of Solutions:** Design solutions and develop systems that meet specified needs while considering safety, cultural, and environmental factors.
4. **PO4 -Investigation of Complex Problems:** Use research-based knowledge and methods to investigate complex issues and derive conclusions.
5. **PO5 -Modern Tool Usage:** Select and apply appropriate techniques, resources, and modern engineering and IT tools to complex engineering activities.
6. **PO6 -The Engineer and Society:** Apply knowledge to assess societal, health, safety, legal, and cultural issues, and understand the engineer's responsibility in these contexts.
7. **PO7 - Environment and Sustainability:** Understand the impact of engineering solutions in societal and environmental contexts and demonstrate knowledge of sustainable development.
8. **PO8 - Ethics:** Apply ethical principles and commit to professional ethics and responsibilities in engineering practice.
9. **PO9 - Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and multidisciplinary settings.
10. **PO10 - Communication**: Communicate effectively on complex engineering activities with peers and the broader community.
11. **PO11 - Project Management and Finance:** Demonstrate knowledge of project management principles and apply them to manage projects efficiently.
12. **PO12 - Life-long Learning**: Recognize the need for continuous learning and engage in independent learning to stay current with technological advancements.

Appendix III: Course Outcomes (COs)

1. **CO1**- Explain DBMS characteristics
2. **CO2**- Construct ER models
3. **CO3**- Develop PL/SQL programs
4. **CO4**- Explain storage structures
5. **CO5**- Explain transaction concepts
6. **CO6**- Design and develop database applications

Appendix IV: Fulfillment of Programme Outcomes (POs)

1. **PO1: Engineering Knowledge**

   Justification:

   The project applies DBMS concepts such as relational schema design, normalization, and indexing. Engineering knowledge is evident in the implementation of complex table relationships, use of ENUM types, and optimized queries for real-time appointment and token updates.

2. **PO2: Problem Analysis**

   Justification:

   Analyzed challenges in hospital operations like overbooking, availability clashes, and appointment mismanagement. These were solved using BEFORE INSERT triggers, unique constraints (e.g., unique_appointment), and FOREIGN KEY checks to maintain data integrity.

3. **PO3: Design/Development of Solutions**

   Justification:

   Designed an efficient multi-table system (appointment, tokens, doctor_availability, accounts) with well-defined primary and foreign keys. Implemented real-time solutions like dynamic token assignment using custom logic in triggers and backend Flask queries.

4. **PO4: Investigation of Complex Problems**

   Justification:

   Investigated complex appointment constraints like duplicate bookings, token overflow, and weekday-based availability. Developed MySQL **triggers** (before_appointment_insert, prevent_overbooking) and validation logic to handle these cases without manual input.

5. **PO5: Modern Tool Usage**

   Justification:

   Used MySQL Workbench for schema design and real-time query testing, and integrated Flask for RESTful APIs. Leveraged stored procedures and triggers for backend logic automation, enhancing the system's responsiveness and reliability.

6. **PO6: The Engineer and Society**

   Justification:

   Through database-driven automation, the system reduces in-hospital crowding and manual errors, supporting safer, faster patient access to healthcare services—especially important during public health emergencies.

7. **PO7: Environment and Sustainability**

   Justification:

   Minimized paper usage and physical queuing through digital tokens and appointment history storage. Database backup and data redundancy mechanisms help sustain system reliability and long-term patient data management.

8. **PO8: Ethics**

   Justification:

   Role-based access (doctor, patient, admin) and secure handling of credentials (e.g., varbinary for password hashing) reflect ethical practices in storing and accessing sensitive health information in compliance with data protection standards.

9. **PO9: Individual and Team Work**

   Justification:

   The system architecture was designed for multi-user access with clear role distinctions stored in the accounts and users tables. Effective table design facilitates simultaneous operations by different users without conflict.

10. **PO10: Communication**

    Justification:

    Database fields like consultation_status and real-time token updates allow patients and doctors to exchange timely status updates via the web interface, powered by efficient query-response handling in the backend.

11. **PO11: Project Management and Finance**

    Justification:

    Backend DBMS structure supports scheduling, resource management (doctor availability), and system auditing. The doctor_availability and appointment tables collectively manage time slots and hospital capacity efficiently.

12. **PO12: Life-long Learning**

    Justification:

    Working with advanced DBMS features like indexing, composite keys, triggers, and stored procedures provided hands-on learning and problem-solving, preparing for future enhancements like multi-hospital or cloud-based extensions.