RV COLLEGE OF ENGINEERING ®
BENGALURU-560059

(Autonomous Institution Affiliated to VTU, Belagavi)

"RVCE HEALTHCARE MANAGEMENT SYSTEM"

Report

Database Management Systems

(CD252IA)

Submitted By

Advith R (1RV22IS005)

Annant Sharma (1RV22IS010)

Arnav Jain (1RV22IS011)

Under the Guidance of

Dr. Kavitha S N

Associate Professor

in partial fulfillment for the award of degree of

Bachelor of Engineering

in

INFORMATION SCIENCE AND ENGINEERING

2024-25

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Mini Project work entitled '**RVCE HEALTHCARE MANAGEMENT SYSTEM**' has been carried out as a part of Database Management Systems Laboratory(CD252IA) in partial fulfillment for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year **2024-2025** by **Advith R(1RV22IS005), Annant Sharma (1RV22IS010), Arnav Jain(1RV22IS011),** who are bonafide students of **RV College of Engineering®**, Bengaluru. It is certified that all the corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of work prescribed by the institution for the said degree.

Dr.Kavitha S N
Associate Professor
Department of ISE,
RVCE, Bengaluru-59

Dr Mamatha GS
Head of the Department
Department of ISE,
RVCE, Bengaluru-59

Name of the Examiners | Signature with Date

1. _____ _____
2. _____ _____

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DECLARATION

We **Advith R(1RV22IS005), Annant Sharma (1RV22IS010), Arnav Jain(1RV22IS011),**are students of Fifth Semester B.E Department of Information Science and Engineering, **RV College of Engineering ®,** bearing **USN:1RV22IS005, 1RV22IS010, 1RV22IS011** hereby declare that the project titled "**RVCE HEALTHCARE MANAGEMENT SYSTEM**'" has been carried out as a part of **DATABASE MANAGEMENT SYSTEMS (CD252IA)** by us and submitted in partial fulfillment of the program requirements for the award of degree in Bachelor of Engineering in Information Science and Engineering of the **Visvevaraya Technological University, Belagavi** during the year **2023-2024.**

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other University.

Place: Bengaluru	**Name**	**Signature with Date**
Date:	**Advith R**	
	Annant Sharma	
	Arnav Jain	

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Terminology

A Healthcare Management System is a comprehensive digital platform designed to streamline and optimize various healthcare-related tasks, including patient records, doctor scheduling, and hospital administration. This system enhances efficiency and improves patient care by integrating multiple functionalities into a single, user-friendly interface. A key component of this system is the Electronic Medical Records (EMR) module, which serves as a digital version of traditional paper charts. It securely stores a patient's complete medical history, including diagnoses, medications, treatment plans, and past appointments, ensuring that healthcare providers have instant access to critical information for informed decision-making.

Additionally, the system features a Patient Portal, a secure online platform that empowers patients by providing easy access to their health records, allowing them to book appointments, receive test results, and directly communicate with healthcare professionals. This not only improves patient engagement but also enhances doctor-patient interactions. The Appointment Scheduling module further optimizes the process by allowing patients to book, manage, and reschedule their appointments effortlessly. Moreover, doctors can accept or decline appointments, notifying patients in real time when their appointment has been approved, reducing wait times and improving workflow efficiency.

To ensure financial transparency and accuracy, the system also includes a Bill Payment Record module. This feature maintains a detailed log of all transactions related to a patient's visits, ensuring that payments are accurately tracked. If a patient consults the same doctor multiple times, their billing receipt will reflect multiple records for each visit, preventing discrepancies and ensuring a clear financial history. By integrating these functionalities, a Healthcare Management System significantly enhances hospital operations, improves patient experiences, and ensures seamless communication between patients and healthcare providers

1.2 Purpose

The purpose of the RVCE Healthcare Management System is to provide a comprehensive solution for efficient management of healthcare services in our college medical center. These systems aim to improve the quality of care, enhance operational efficiency, and ensure timely access to healthcare for students, faculty, and staff and also to optimize the administrative processes to reduce the delay and errors.

1.3 Motivation

As students of RVCE, we recognized the growing need for a digital healthcare platform within our campus to address common challenges such as delays in accessing medical records, difficulties in booking appointments, and the absence of a centralized health information system.

The increasing reliance on technology across various domains inspired us to develop a solution that enhances efficiency and accessibility in healthcare services. Our system aims to bridge the gap between patients and healthcare providers by simplifying processes, ensuring seamless communication, and streamlining administrative tasks. By reducing the burden on healthcare staff**,** the platform automates routine operations such as appointment scheduling and medical record management, allowing healthcare professionals to focus on patient care rather than manual paperwork. Additionally, the system is designed to improve patient satisfaction by making healthcare services more accessible, transparent, and user-friendly, ultimately fostering a more efficient and responsive healthcare environment within the RVCE campus

1.4 Problem Statement

The healthcare services within RVCE face multiple challenges, primarily due to outdated manual processes and limited technological integration. Manual record-keeping is prone to errors and inefficiencies, making it difficult to retrieve or update patient information swiftly. This can lead to delays in treatment and a lack of consistency in healthcare delivery. Additionally, the absence of a structured appointment booking system results in long waiting times and underutilization of available resources, which ultimately affects patient satisfaction. Furthermore, students and staff have limited access to their health records, making it inconvenient to track medical history or share information with healthcare providers as needed.

The proposed RVCE Healthcare Management System seeks to address these issues comprehensively by digitizing and automating healthcare processes. By implementing a centralized system, it ensures a seamless and user-friendly experience for all stakeholders. Advanced features like telemedicine are integrated to enhance inclusivity and efficiency, enabling remote consultations and reducing the need for physical visits. This innovative approach aims to create a robust and reliable healthcare infrastructure that caters to the needs of the entire RVCE community.

1.5 Objective

The RVCE Healthcare Management System is designed with the primary objective of creating a centralized platform for managing patient records, doctor schedules, and appointment bookings, ensuring seamless and efficient healthcare operations within the campus. By providing a secure portal**,** the system enables students and staff to conveniently access their health records, empowering them to stay informed about their medical history and ongoing treatments. Additionally, to enhance accessibility and convenience, the system integrates telemedicine features that allow remote consultations, which are particularly beneficial during emergencies or for minor ailments, reducing the need for unnecessary hospital visits. Another key goal is to improve overall efficiency by automating routine administrative tasks, thereby reducing manual workload for healthcare staff and allowing them to focus more on patient care. Furthermore, the system ensures transparency in billing and financial management by maintaining detailed and accurate transaction records, preventing discrepancies and fostering trust between patients and healthcare providers. Through these objectives, the RVCE Healthcare Management System aims to enhance the overall healthcare experience, making it more efficient, accessible, and patient-centric.

1.6 Scope and Relevance

The scope of this project extends to all healthcare services within RVCE, ensuring a more efficient and accessible system for students and staff. It includes the digital storage and retrieval of medical records, minimizing errors and improving accessibility while allowing healthcare providers to make informed decisions based on accurate patient data. Additionally, the system enables real-time appointment scheduling and management, reducing waiting times and optimizing resource allocation to ensure that medical services are delivered promptly. To maintain financial transparency, the platform includes a comprehensive log that records all bill payments associated with a patient. If a patient visits the same doctor multiple times, their bill payment receipt will reflect multiple records for accurate transaction tracking. With an increasing number of students and staff on campus, implementing a robust healthcare management system is essential for ensuring timely and efficient service delivery. By leveraging technology, this project aligns with the college's mission to embrace innovation for improving campus life, streamlining healthcare processes, and enhancing patient experience. Furthermore, it serves as a model for other institutions aiming to modernize their healthcare systems, demonstrating how digital transformation can optimize medical services in an academic setting.

CHAPTER-2
REQUIREMENT SPECIFICATION

2.1 Specific Requirements

The RVCE healthcare Platform needs to fulfill a comprehensive set of specific requirements to function effectively, offering a seamless, secure, and responsive experience for users. These requirements are categorized into functional and non-functional needs to ensure that the system operates efficiently and is user-friendly.

2.1.1 Functional Requirements

The system includes a user registration and login feature, allowing students, faculty, and staff to register using their unique IDs and passwords. The registration process incorporates validation measures to ensure that only authorized individuals can create accounts. A secure login system with role-based access is implemented for administrators and medical staff, ensuring confidentiality and controlled access to sensitive information. The electronic medical records module maintains a digital repository of patient prescriptions, enabling healthcare providers to update records in real-time for accurate and up-to-date medical history. The appointment scheduling system allows users to book, reschedule, or cancel appointments with healthcare providers. It displays available time slots and sends confirmation messages upon successful booking while also allowing healthcare staff to manage schedules efficiently by viewing and updating appointments in real-time. Additionally, the system maintains a comprehensive bill payment record that logs all transactions associated with a patient. If a patient visits the same doctor multiple times, their bill payment receipt will reflect multiple records, ensuring accurate tracking of transactions and financial transparency.

2.1.2 Non-Functional Requirements

1. **Scalability**:
    - The system should handle increasing numbers of users and data without performance degradation. It should be designed to accommodate future expansions, such as additional features or larger databases.

2. **Usability**:
    - User-friendly interface suitable for all age groups, ensuring ease of use for students, staff, and healthcare providers. The interface should follow accessibility guidelines to cater to users with disabilities.

3. **Security**:
    - Ensure data confidentiality through encryption and secure authentication methods. Regular audits and updates should be performed to address potential vulnerabilities.

4. **Reliability**:
    - Guarantee system uptime for critical healthcare services, with backup systems in place to prevent data loss during failures.

2.2.2 Hardware Requirements

The system is designed with a strong emphasis on scalability, ensuring that it can efficiently handle an increasing number of users and a growing volume of medical data without compromising performance. Its architecture is built to support future expansions, including the integration of advanced features, additional healthcare services, and larger databases as the needs of the institution evolve. Usability is another key focus, with a well-structured and intuitive user interface that caters to students, faculty, and healthcare providers, making navigation seamless for individuals of all age groups. The system follows accessibility guidelines to accommodate users with disabilities, ensuring inclusivity and ease of use for everyone.

Security is a top priority, with robust measures such as encryption protocols, multi-factor

authentication, and secure login mechanisms in place to safeguard sensitive medical data. Regular security audits and software updates are conducted to address potential vulnerabilities, ensuring that the system remains resilient against cyber threats. Additionally, strict access controls are implemented to protect confidential patient information, ensuring that only authorized personnel can access or modify critical records.

Reliability is another crucial aspect, as uninterrupted access to healthcare services is essential. The system guarantees high uptime, with fail-safe mechanisms in place to ensure continuous operation even in the event of technical issues. Backup systems are integrated to prevent data loss, enabling quick recovery in case of failures. By incorporating these features, the platform provides a secure, efficient, and scalable healthcare management solution that enhances the overall healthcare experience for students, staff, and medical professionals at RVCE.

2.2.3 Software Requirements

The development environment for the Stock Trading Platform is designed to ensure efficiency, flexibility, and seamless collaboration. The primary Integrated Development Environment (IDE**)** used for development is Visual Studio Code (v1.96)**,** providing a lightweight yet powerful coding experience with extensive extensions and debugging capabilities.

For the frontend, the platform utilizes HTML, CSS, and JavaScript (ECMAScript 2023) to create a dynamic and responsive user interface. Additionally, React (v18.2.0) is used for building modular and interactive UI components, ensuring a smooth and engaging user experience.

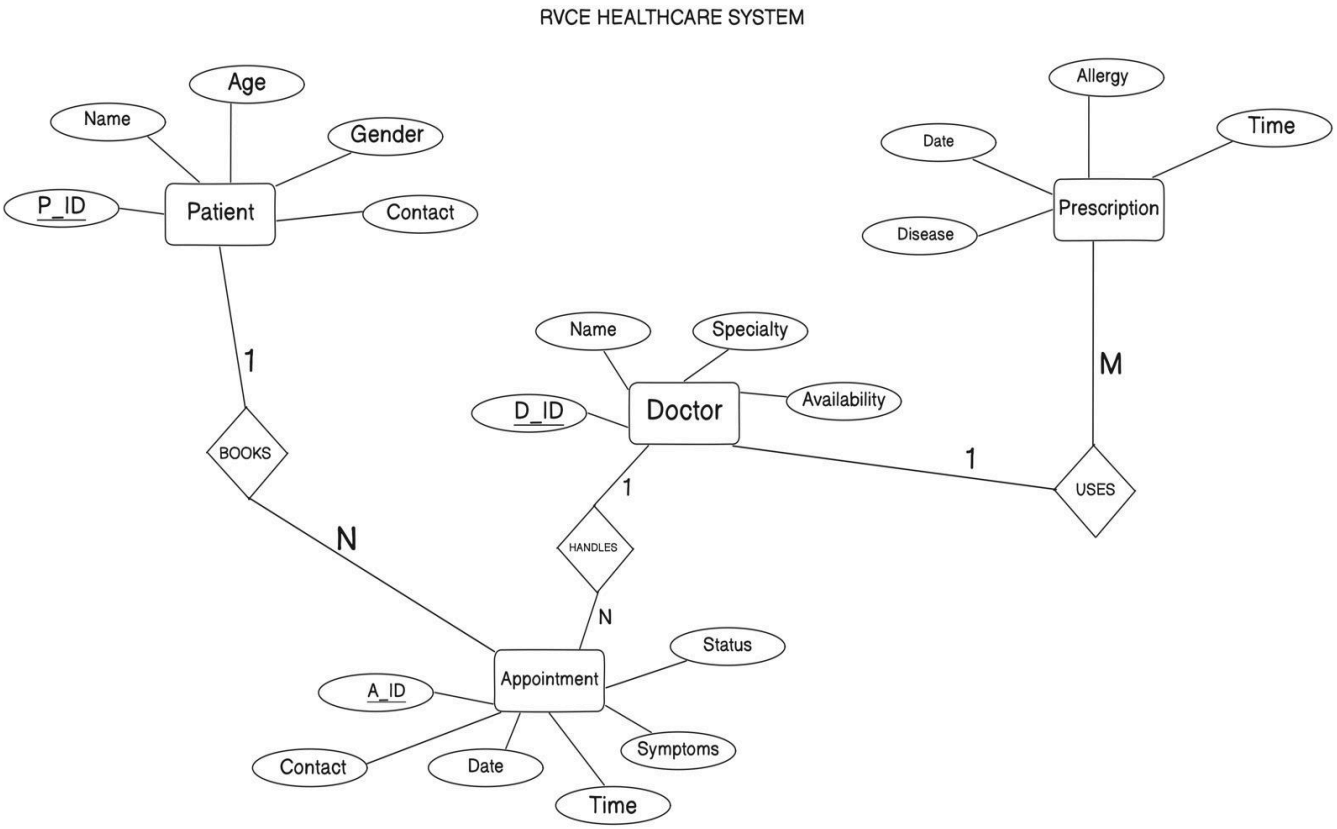
On the backend, the system is powered by Python 3.10, offering robust support for high-performance computation and data handling. The backend framework can be either Flask (v2.3) for lightweight applications or Django (v4.1) for more structured and scalable development, ensuring flexibility based on project requirements.

For version control**,** Git is used to manage source code efficiently, allowing developers to track changes, collaborate, and maintain code integrity. The project is hosted on GitHub, enabling seamless collaboration, pull requests, and issue tracking to streamline development workflows. This well-structured development environment ensures an optimal balance of performance, maintainability, and scalability for the platform.

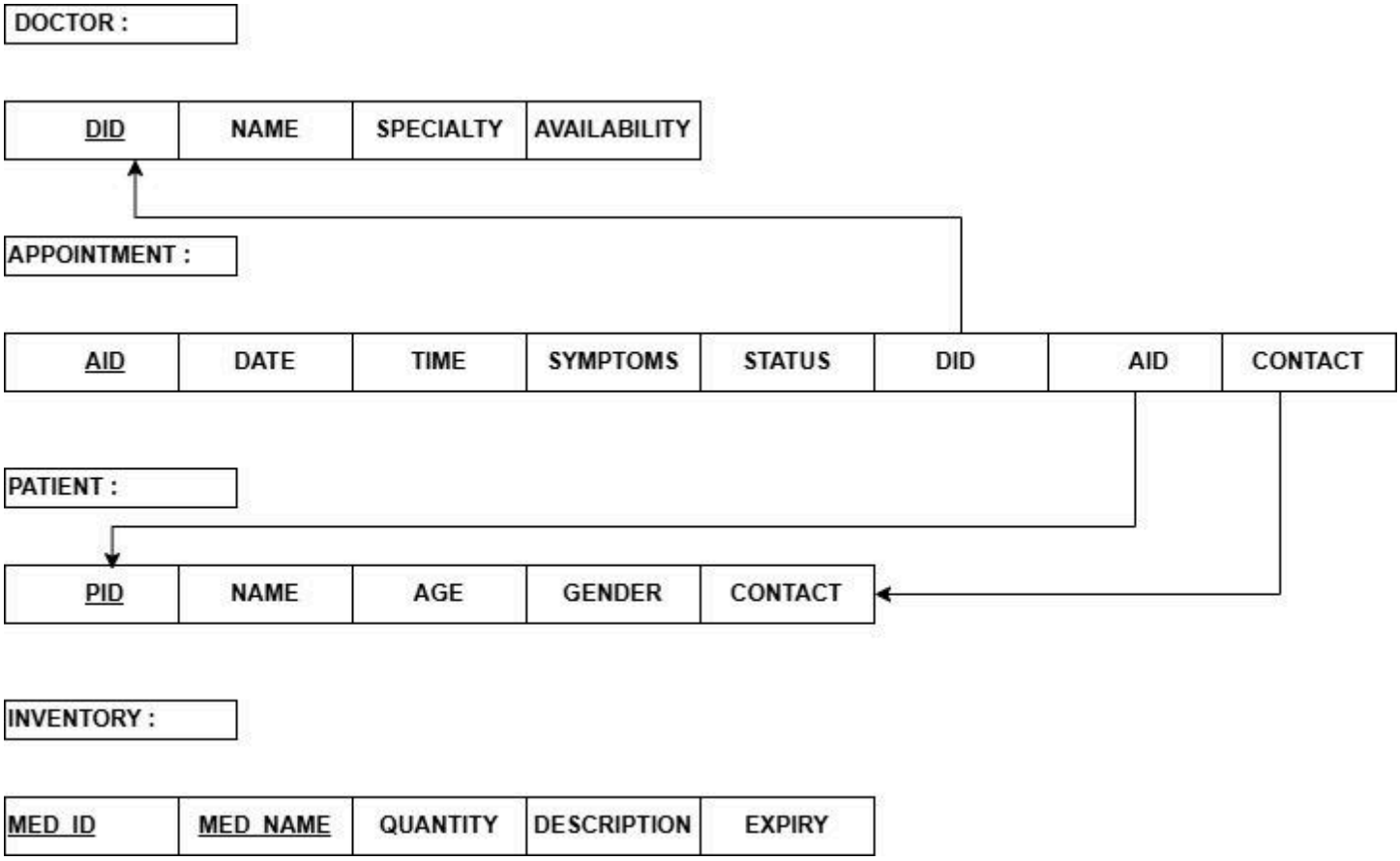
CHAPTER -3

DESIGN

3.1 E-R Diagram



3.1.1 Schema Representation



3.2 Normalization

3.2.1 Schema after Normalization

DOCTOR :

DID	NAME	SPECIALTY	AVAILABILITY

APPOINTMENT :

AID	DATE	TIME	SYMPTOMS	STATUS	DID	AID	CONTACT

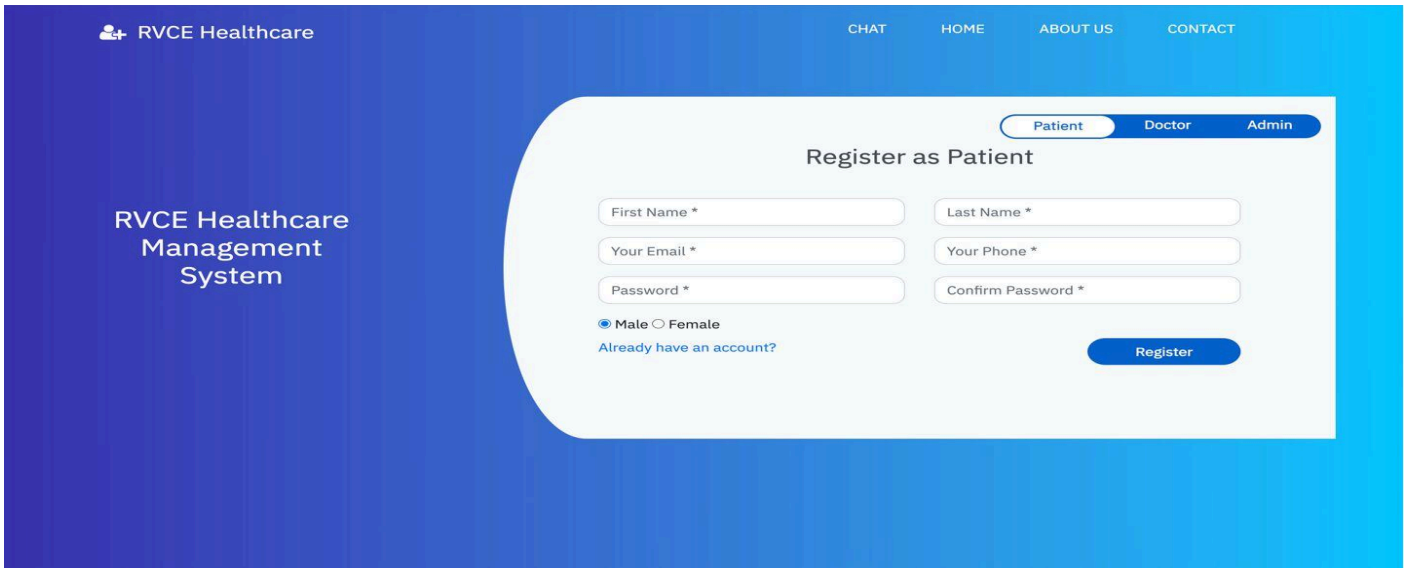
PATIENT :

PID	NAME	AGE	GENDER	CONTACT

INVENTORY :

MED_ID	MED_NAME	QUANTITY	DESCRIPTION	EXPIRY

3.3 Front End Design



IMPLEMENTATION DETAILS

4.1 Database Implementation

The database implementation for the Stock Trading Platform involves the creation of several tables to manage data related to users, companies, stocks, transactions, and market data. Below are the SQL queries used to create these tables:

4.1.1 Table Creation

Appointment Table

The appointmenttb table is a database table designed for managing appointments in a hospital management system. It stores details such as patient information (pid, fname, lname), contact details, assigned doctor, fees, appointment date and time, and status flags (userStatus, doctorStatus) for tracking appointment progress.

```
CREATE TABLE appointmenttb (
  pid int(11) NOT NULL,
  ID int(11) NOT NULL,
  fname varchar(20) NOT NULL,
  lname varchar(20) NOT NULL,
  gender varchar(10) NOT NULL,
  email varchar(30) NOT NULL,
  contact varchar(10) NOT NULL,
  doctor varchar(30) NOT NULL,
  docFees int(5) NOT NULL,
  appdate date NOT NULL,
  apptime time NOT NULL,
  userStatus int(5) NOT NULL,
  doctorStatus int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Doctor Table

The `doctb` table is designed to store information about doctors in a hospital management system. It includes fields for login credentials (`username`, `password`), contact information (`email`), specialization (`spec`), and consultation fees (`docFees`).

```
CREATE TABLE `doctb` (
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `spec` varchar(50) NOT NULL,
  `docFees` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Patient Table

The `patreg` table is a database table designed for storing patient registration details in a hospital management system. It includes fields for personal information (`fname`, `lname`, `gender`), contact details (`email`, `contact`), and account credentials (`password`, `cpassword`).

```
CREATE TABLE `patreg` (
  `pid` int(11) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `email` varchar(30) NOT NULL,
  `contact` varchar(10) NOT NULL,
```

```
  `password` varchar(30) NOT NULL,
  `cpassword` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.1.2 Table Population

Populate Table Users

INSERT INTO contact (name , email , contact , message) VALUES

('Anu', 'anu@gmail.com', '7896677554', 'Hey Admin'),

(' Viki', 'viki@gmail.com', '9899778865', 'Good Job, Pal'),

('Ananya', 'ananya@gmail.com', '9997888879', 'How can I reach you?'),

('Aakash', 'aakash@gmail.com', '8788979967', 'Love your site'),

('Mani', 'mani@gmail.com', '8977768978', 'Want some coffee?'),

('Karthick', 'karthi@gmail.com', '9898989898', 'Good service'),

('Abbis', 'abbis@gmail.com', '8979776868', 'Love your service'),

('Asiq', 'asiq@gmail.com', '9087897564', 'Love your service. Thank you!'),

('Jane', 'jane@gmail.com', '7869869757', 'I love your service!');

This SQL `INSERT INTO` statement adds multiple rows of data into a table named `contact`. Each row represents a record containing a name (`name`), email address (`email`), contact number (`contact`), and a message (`message`) from users or customers interacting with the system. It is typically used to store user feedback or inquiries.

Populate Table Doctors

INSERT INTO doctb (username, password, email, spec, docFees) VALUES

('ashok', 'ashok123', 'ashok@gmail.com', 'General', 500),

('arun', 'arun123', 'arun@gmail.com', 'Cardiologist', 600),

('Dinesh', 'dinesh123', 'dinesh@gmail.com', 'General', 700),

('Ganesh', 'ganesh123', 'ganesh@gmail.com', 'Pediatrician', 550),

('Kumar', 'kumar123', 'kumar@gmail.com', 'Pediatrician', 800),

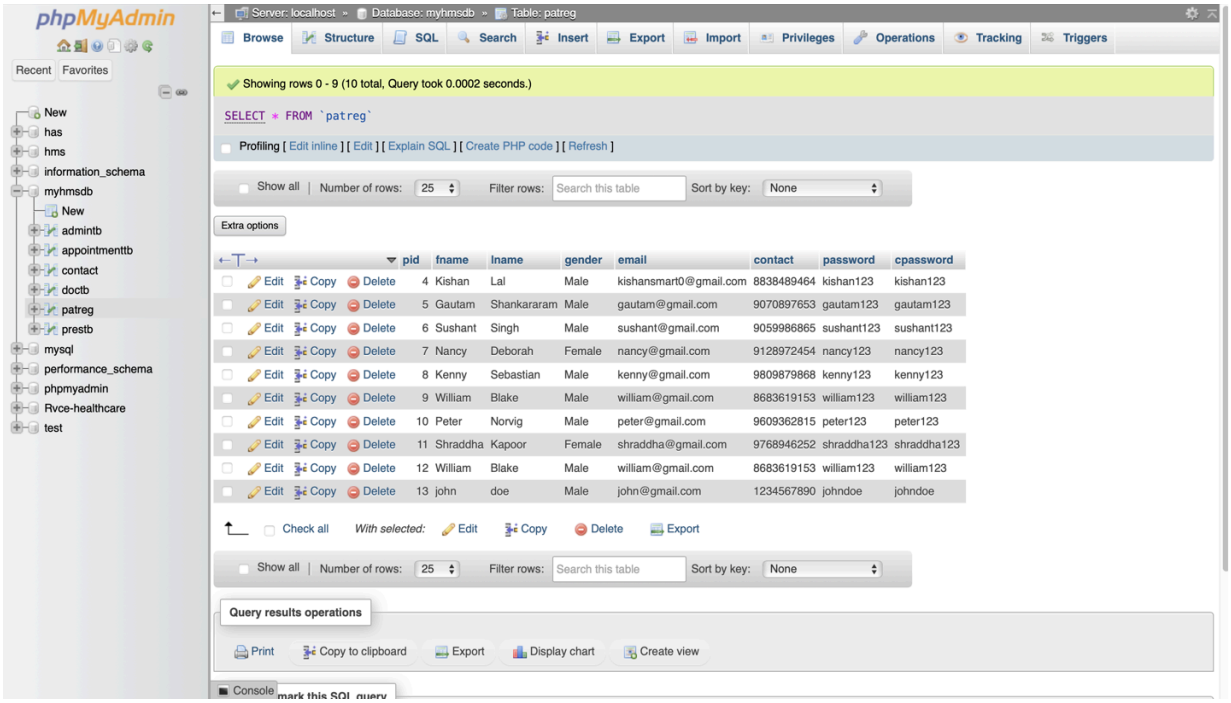
('Amit', 'amit123', 'amit@gmail.com', 'Cardiologist', 1000),

('Abbis', 'abbis123', 'abbis@gmail.com', 'Neurologist', 1500),

('Tiwary', 'tiwary123', 'tiwary@gmail.com', 'Pediatrician', 450);

The `INSERT INTO doctb` statement adds records of doctors, including their `username`, `password`, `email`, `specialization (spec)`, and consultation fees (`docFees`), into the database. This table is used to manage doctor profiles and their professional details in the hospital management system.

4.1.3 Query Execution and Output



4.2 Front End Implementation

Frontend technologies used in this code:

1. HTML5
2. Bootstrap 4.3.1 CSS
3. Bootstrap JavaScript
4. Custom CSS (style1.css)
5. Google Fonts (IBM Plex Sans)
6. Font Awesome icons
7. Custom JavaScript for form validation (password matching, alpha-only input)

4.2.1 Form Creation

Login Forms

The login functionality includes separate forms for Doctors, Patients, and Admins, each requiring essential fields such as Username/Email and Password. Upon submission, the login data is sent via a POST request to the respective PHP endpoints—func1.php for Doctors, func3.php for Patients, and func.php for Admins. These endpoints process the login credentials, verifying the information against the database and handling authentication. This setup ensures that each user type is correctly authenticated and granted appropriate access based on their role.

Patient Registration Form

The registration form collects essential user information, including First Name, Last Name, Email, Phone Number, Password, and Gender. To ensure valid input, JavaScript functions are implemented for input validation, such as restricting names to alphabetic characters using the "alphaOnly" function, verifying the password's length with a minimum of six characters, and checking for password confirmation match through the "checklen" and "check" functions. Real-time feedback is provided for password matching, with color-coded messages like "Matched" or "Not Matching" to guide users. Once all the fields are validated, the form submits the data via a POST request to func2.php for further processing.

4.2.2 Connectivity to the Database

The front-end forms communicate with the server through HTTP POST requests, sending the data to corresponding PHP scripts (func2.php, func1.php, func3.php, func.php), which handle the interaction with the database. The PHP backend establishes a connection with the database to store patient registration information and authenticate users based on their credentials. The scripts validate user input before communicating with the database, likely MySQL or PostgreSQL, ensuring proper data submission. To enhance security, password fields use the "password" input type to mask the entered data, and additional input validation helps prevent common errors, ensuring that the data submitted is both safe and accurate.

4.2.3Security Features

The security features of the system include the use of "password" input type for password fields, which ensures that the input is masked and not visible to others. Additionally, input validation is implemented to mitigate common errors, ensuring that only valid data is submitted. This helps prevent issues such as incorrect formatting or malicious input, thereby enhancing the safety and integrity of the data being handled. These measures work together to safeguard user information and ensure secure data submission.

TESTING AND RESULTS

The Testing phase is an essential part of verifying that the RVCE Hospital Management System functions accurately and efficiently. Various types of testing, including Database Testing, Front-End Testing, and System Testing, were performed to ensure the reliability of the system.

5.1 Database Testing

Database testing is crucial to ensure that the system's database operations, such as CRUD (Create, Read, Update, Delete), transaction handling, and data integrity, work as expected. The following tests were conducted:

5.1.1 Test Cases

Patient Profile Creation Test

Objective: Verify that new patients can be added successfully.

Test Input: Patient's first name, last name, email, phone number

and password.

Expected Result: A new entry Patient profile page should appear.

Appointment Record Test

Objective: Ensure that appointment bookings are properly recorded.

Test Input: Appointment date, appointment time, Specialization.

Expected Result: A new appointment is created with the correct details.

Admin Data Integrity Test

Objective: Confirm that all the bookings and details are recorded correctly in the admin database.

Test Input: Admin login credentials to check database.

Expected Result: Data should appear in the table, with correct appointment details.

Prescription Test

Objective: Test the update functionality when doctors write a prescription .

Test Input: Prescription filled correctly by the doctor.

Expected Result: The prescription should reflect in the patients database.

5.2 Front End Testing

Front end testing focuses on validating the user interface, usability, and user interaction flow. The tests are designed to ensure that users can interact with the platform seamlessly, and the platform responds appropriately to various user inputs.

5.2.1 Test Cases

Login Form Validation

Objective: Ensure the login form accepts valid credentials and rejects invalid ones.

Test Input: Valid and invalid user credentials.

Expected Result: Successful login for valid credentials and an error message for invalid ones.

Sign Up Form Validation

Objective: Test the user registration form for correct input validation. Test Input:

New user details (first name, last name, email, password, phone number).

Expected Result: Successful registration with correct form validation (e.g., email format check, password strength check).

Booking Appointment Validation

Objective: Verify that the patient is able to book an appointment on a chosen date with the chosen doctor.

Test Input: Specialization, Doctor, Appointment Date, Appointment Time.

Expected Result: The appointment is reflected in both Doctors and patients databases.

Prescription Validation

Objective: Prescription written by the doctor for an individual patient.

Test Input: Disease, Allergies, prescription.

Expected Result: The Prescription is reflected in the patients database.

5.3 System Testing

System testing evaluates the entire platform's integration, functionality, and overall performance. This phase includes testing the interactions between the front end, back end, and the database.

5.3.1 Test Cases

End-to-End Appointment Booking Test

Objective: Verify that the Appointment is reflecting in both Patient database and Doctor Database.

Test Input: Specialization, Doctor name, Appointment Date, Appointment Time.

Expected Result: Front-end is updated, appointment is recorded in the databases.

Admin Test

Objective: Ensure all changes made by the admin are reflected.

Test Input: Add Doctor details or remove Doctor details.

Expected Result: Correct responses, such as updated data in the database.

Security Testing

Objective: Test the security of user authentication, data transmission, and storage.

Test Input: Attempt login with incorrect credentials, intercept data during transmission.

Expected Result: Unauthorized login attempts should fail, and data should be encrypted during transmission.

API Testing

To ensure the correctness and performance of the APIs. The following test cases were conducted:

Login API Test

Objective: Test the login functionality.

Test Input: Post request with a user’s email and password.

Expected Result: Successful authentication and a token returned for session management.

Book Appointment API Test

Objective: Test the convenience of booking an appointment .

Test Input: Post request with patient details, appointment date, appointment time.
Expected Result: Successful booking, with updates on the appointment status.

This testing phase ensures that all components of the Hospital Management System are functioning correctly, and any issues are identified and resolved before deployment, the system's robustness and reliability are ensured.

CONCLUSION

In conclusion, the DBMS model for the RVCE Hospital Management System provides an efficient, secure, and streamlined solution for managing hospital operations. By integrating patient registration, doctor and admin authentication, and seamless database connectivity, the system ensures data accuracy, accessibility, and security. The use of input validation and backend processing enhances the reliability of the system, while its modular design simplifies scalability and maintenance. This model serves as a foundation for further development, enabling RVCE Hospital to deliver better healthcare services through improved data management and operational efficiency.

6.1 Limitations

Despite the system's success, there are several limitations that need to be addressed to ensure its long-term efficiency and security. One of the primary challenges is scalability. As the number of patients, doctors, and administrators increases, the system may experience performance bottlenecks if it is not optimized for scalability. Without proper indexing, efficient database design, and load-balancing techniques, queries on large datasets could result in slower response times, impacting user experience and system reliability.

Another limitation is the limited role-based access control (RBAC). While the system provides separate login forms for doctors, patients, and administrators, it lacks advanced RBAC features that would enforce more granular restrictions on user permissions. This limitation could lead to unauthorized access to sensitive information if proper restrictions on database operations are not enforced. Additionally, security concerns persist in the current implementation, as the system only offers basic input validation and masked password fields. The absence of advanced security measures like data encryption, two-factor authentication, and protection against SQL injection increases the risk of potential vulnerabilities and cyberattacks. Furthermore, the error handling and logging capabilities are insufficient, making it challenging to trace and debug issues in real time, especially during unexpected failures. Lastly, the system's dependency on PHP for the backend, while functional, may not be the most efficient choice for handling larger, more complex operations, suggesting that a transition to more modern, robust frameworks or languages may be necessary as the system scales.

6.2 Future Enhancements

1. **Improved Security Measures**:

Implement robust security protocols, including encryption of sensitive data (e.g., passwords and personal information), two-factor authentication, and protection against SQL injection and cross-site scripting (XSS) attacks.

2. **Role-Based Access Control (RBAC)**:

   Introduce advanced role-based access controls to restrict actions based on user roles. For example, administrators can manage all records, while doctors and patients access only their respective data.

3. **Scalability and Performance Optimization**:

   Optimize the database with indexing and efficient query structures. Use caching mechanisms to improve response times for frequently accessed data and support larger datasets as the hospital expands.

4. **Responsive and User-Friendly Interface**:

   Redesign the user interface to be more intuitive and mobile-responsive, enabling seamless access across devices like smartphones, tablets, and desktops.

5. **Integration with Third-Party Tools**:

   Enable integration with external systems like insurance platforms, pharmacy management systems, and diagnostic tools for streamlined operations and reduced manual work.

REFERENCES

GeeksforGeeks: Articles on database management, PHP programming, and hospital management systems.

https://www.geeksforgeeks.org

TutorialsPoint: Guides on database design, system development, and frontend-backend integration.

https://www.tutorialspoint.com

"Design and Implementation of a Hospital Management System" – Published in the *International Journal of Computer Science and Information Security (IJCSIS).*

IEEE Xplore Digital Library for relevant technical papers on hospital management systems.

https://ieeexplore.ieee.org

"Database Design for a Hospital Management System" – Published in the *Journal of Software Engineering and Applications.*

APPENDIX A - CODE SNIPPETS

Appointments

```
CREATE TABLE  appointmenttb  (
  pid  int(11) NOT NULL,
  ID  int(11) NOT NULL,
  fname  varchar(20) NOT NULL,
  lname  varchar(20) NOT NULL,
  gender  varchar(10) NOT NULL,
  email  varchar(30) NOT NULL,
  contact  varchar(10) NOT NULL,
  doctor  varchar(30) NOT NULL,
  docFees  int(5) NOT NULL,
  appdate  date NOT NULL,
  apptime  time NOT NULL,
  userStatus  int(5) NOT NULL,
  doctorStatus  int(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Doctor

```
CREATE TABLE `doctb` (
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
```

```
  `email` varchar(50) NOT NULL,
  `spec` varchar(50) NOT NULL,
  `docFees` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Patients

```
CREATE TABLE `patreg` (
  `pid` int(11) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `gender` varchar(10) NOT NULL,
  `email` varchar(30) NOT NULL,
  `contact` varchar(10) NOT NULL,
  `password` varchar(30) NOT NULL,
  `cpassword` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

User login authentication

```
if(isset($_POST['patsub'])){
    $email=$_POST['email'];
    $password=$_POST['password2'];
```

```
        $query="select * from patreg where email='$email' and password='$password';";
    $result=mysqli_query($con,$query);
    if(mysqli_num_rows($result)==1)
    {
       while($row=mysqli_fetch_array($result,MYSQLI_ASSOC)){
          $_SESSION['pid'] = $row['pid'];
          $_SESSION['username'] = $row['fname']." ".$row['lname'];
          $_SESSION['fname'] = $row['fname'];
          $_SESSION['lname'] = $row['lname'];
          $_SESSION['gender'] = $row['gender'];
          $_SESSION['contact'] = $row['contact'];
          $_SESSION['email'] = $row['email'];
       }
       header("Location:admin-panel.php");
    }
    else {
            echo("<script>alert('Invalid Username or Password. Try Again!');
        window.location.href = 'index1.php';</script>");
    }
}
```

Doctor authentication

```
if(isset($_POST['docsub1'])){
        $dname=$_POST['username3'];
        $dpass=$_POST['password3'];
        $query="select * from doctb where username='$dname' and password='$dpass';";
        $result=mysqli_query($con,$query);
        if(mysqli_num_rows($result)==1)
        {
   while($row=mysqli_fetch_array($result,MYSQLI_ASSOC)){

                    $_SESSION['dname']=$row['username'];

   }
              header("Location:doctor-panel.php");
        }
        else{
   // header("Location:error2.php");
   echo("<script>alert('Invalid Username or Password. Try Again!');
         window.location.href = 'index.php';</script>");
  }
```

```
}
```

Admin authentication

```
if(isset($_POST['adsub'])){
        $username=$_POST['username1'];
        $password=$_POST['password2'];
        $query="select * from admintb where username='$username' and password='$password';";
        $result=mysqli_query($con,$query);
        if(mysqli_num_rows($result)==1)
        {
                $_SESSION['username']=$username;
                header("Location:admin-panel1.php");
        }
        else
                // header("Location:error2.php");
                echo("<script>alert('Invalid Username or Password. Try Again!');
         window.location.href = 'index.php';</script>");
}
```

Update prescription

```
if(isset($_POST['prescribe']) && isset($_POST['pid']) && isset($_POST['ID']) && isset($_POST['appdate']) && isset($_POST['apptime']) && isset($_POST['lname']) && isset($_POST['fname'])){
  $appdate = $_POST['appdate'];
  $apptime = $_POST['apptime'];
  $disease = $_POST['disease'];
  $allergy = $_POST['allergy'];
  $fname = $_POST['fname'];
  $lname = $_POST['lname'];
  $pid = $_POST['pid'];
  $ID = $_POST['ID'];
  $prescription = $_POST['prescription'];

                $query=mysqli_query($con,"insert into prestb(doctor,pid,ID,fname,lname,appdate,apptime,disease,allergy,prescription) values ('$doctor','$pid','$ID','$fname','$lname','$appdate','$apptime','$disease','$allergy','$prescription')");
    if($query)
    {
      echo "<script>alert('Prescribed successfully!');</script>";
```

```
    }
    else{
        echo "<script>alert('Unable to process your request. Try again!');</script>";
    }
}
```

Search appointments

```
if(isset($_POST['app_search_submit']))
{
    $contact=$_POST['app_contact'];
    $query = "select * from appointmenttb where contact= '$contact';";
    $result = mysqli_query($con,$query);
    $row=mysqli_fetch_array($result);

    if($row['fname']=="" & $row['lname']=="" & $row['email']=="" & $row['contact']=="" & $row['doctor']=="" & $row['docFees']=="" & $row['appdate']=="" & $row['apptime']==""){
        echo "<script> alert('No entries found! Please enter valid details');
        window.location.href = 'admin-panel1.php#list-doc';</script>";
```

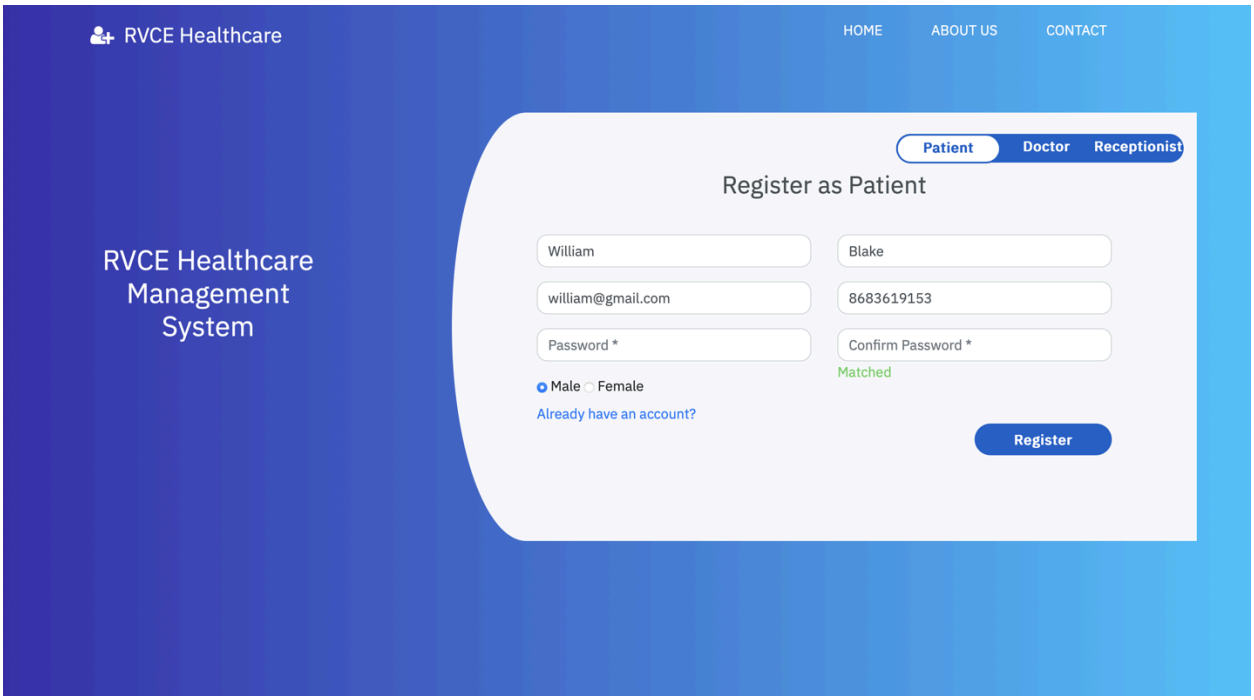
```
}
else {
    // Code to display appointment details in a table
    $fname = $row['fname'];
    $lname = $row['lname'];
    $email = $row['email'];
    $contact = $row['contact'];
    $doctor = $row['doctor'];
    $docFees= $row['docFees'];
    $appdate= $row['appdate'];
    $apptime = $row['apptime'];

    // Determine appointment status
    if(($row['userStatus']==1) && ($row['doctorStatus']==1))
    {
        $appstatus = "Active";
    }
    if(($row['userStatus']==0) && ($row['doctorStatus']==1))
    {
        $appstatus = "Cancelled by You";
    }
    if(($row['userStatus']==1) && ($row['doctorStatus']==0))
```

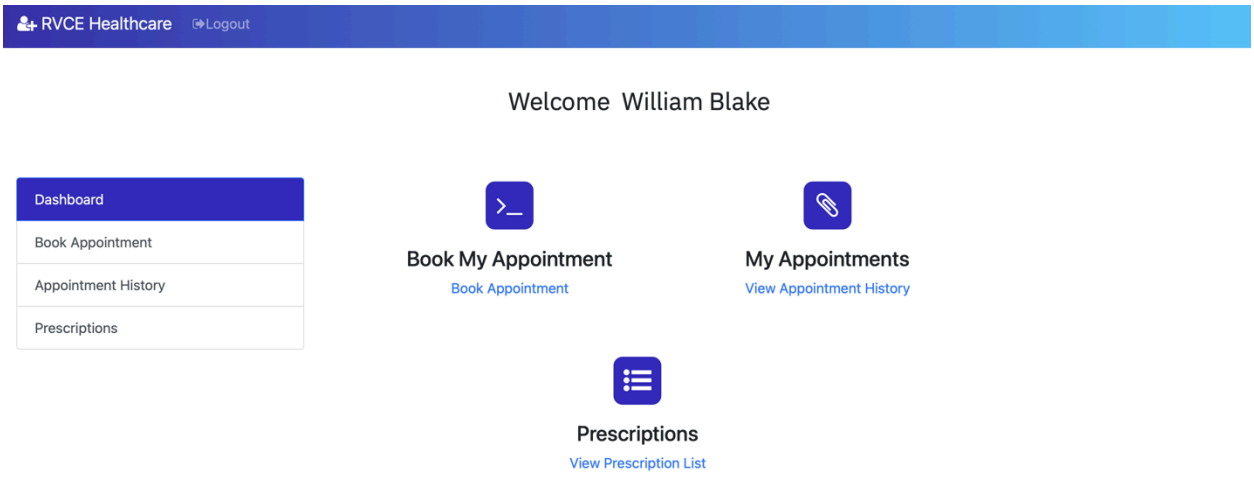
```
        {
            $appstatus = "Cancelled by Doctor";
        }

        // Display appointment details in a table
    }
}
```

APPENDIX B-SCREENSHOTS

Registration Page:

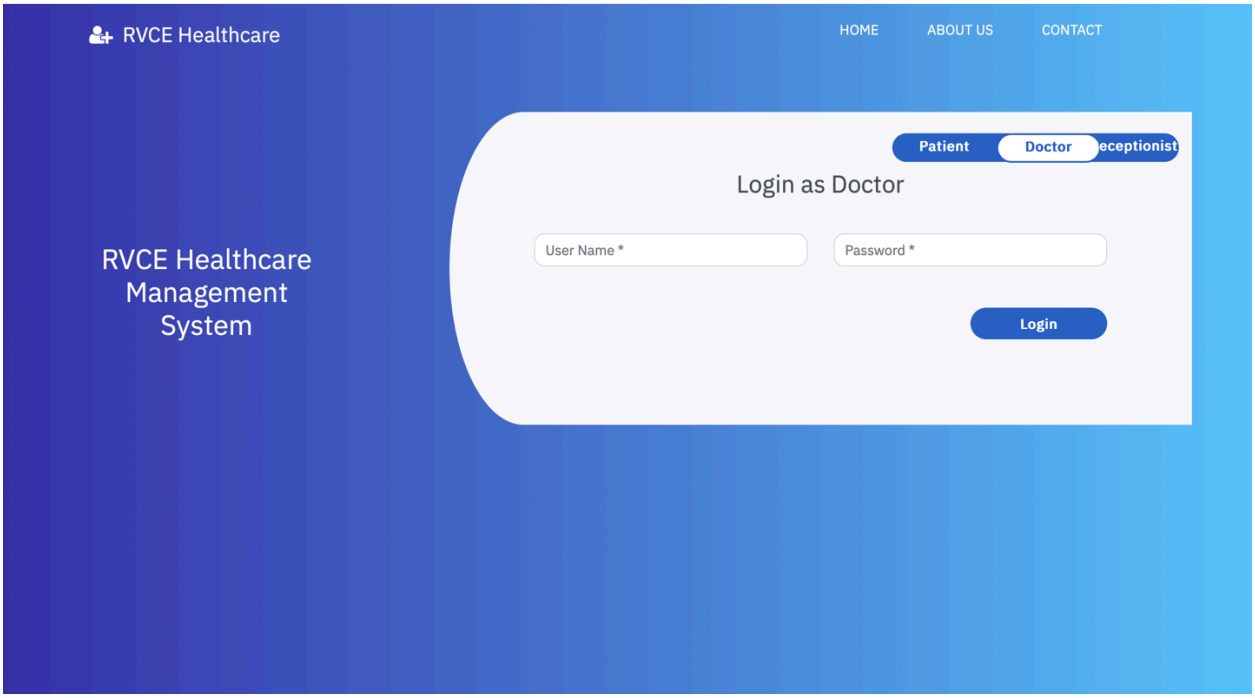
1. Patient



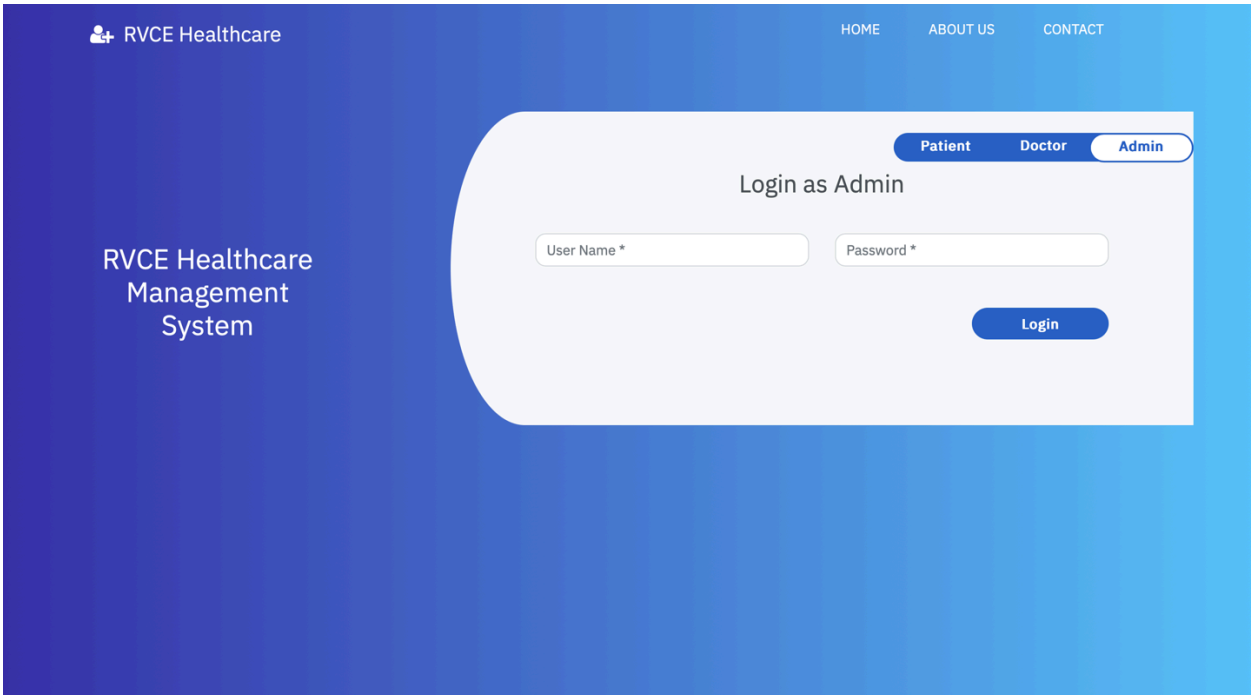


Login Page:

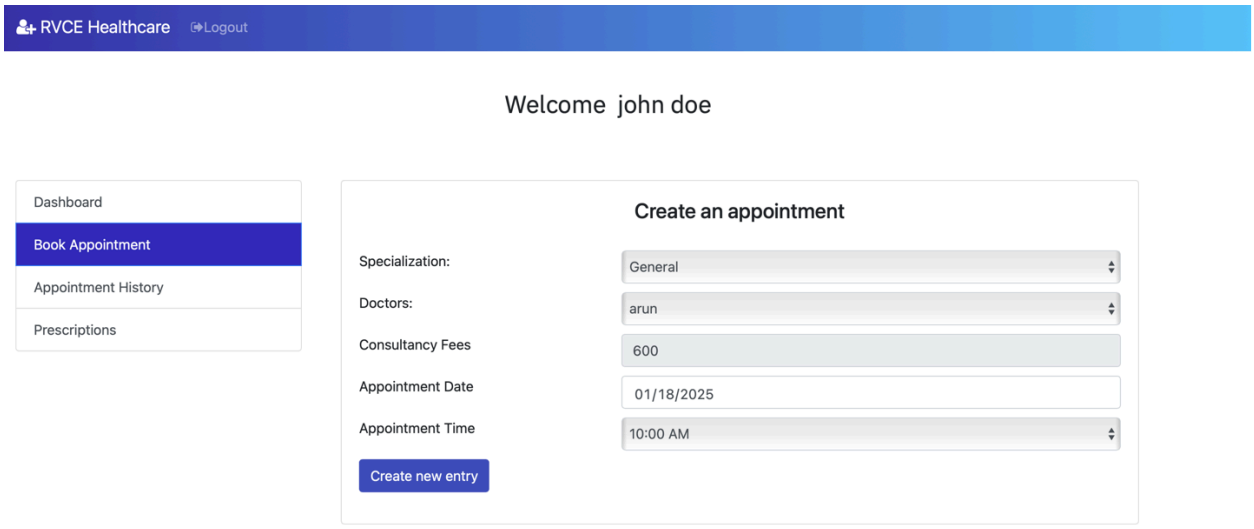
1. Doctor



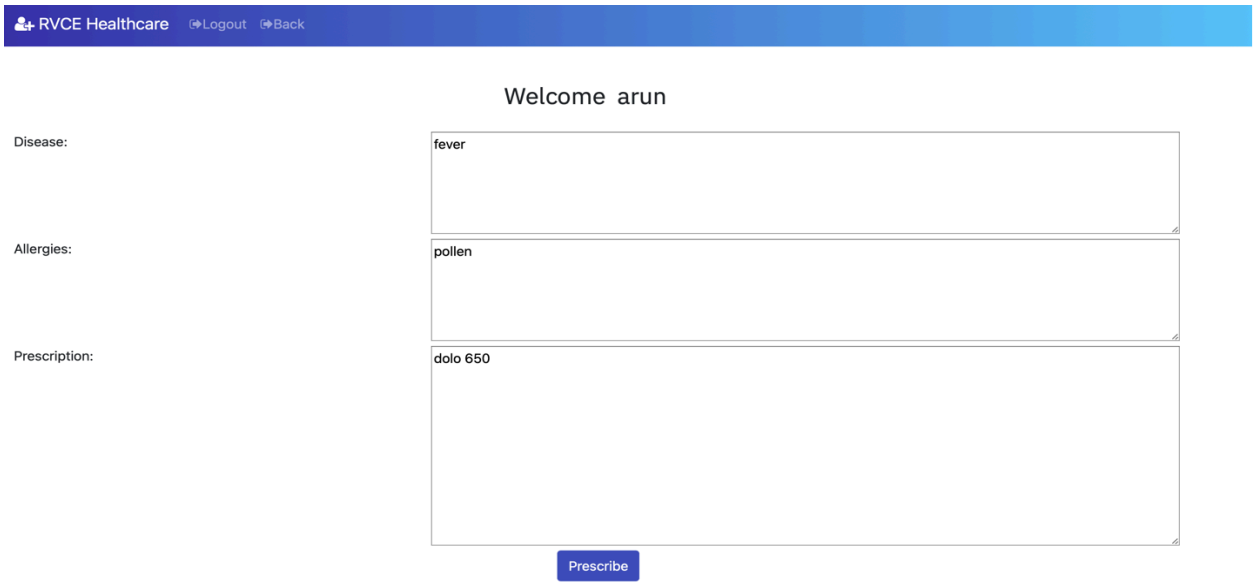
2. Admin



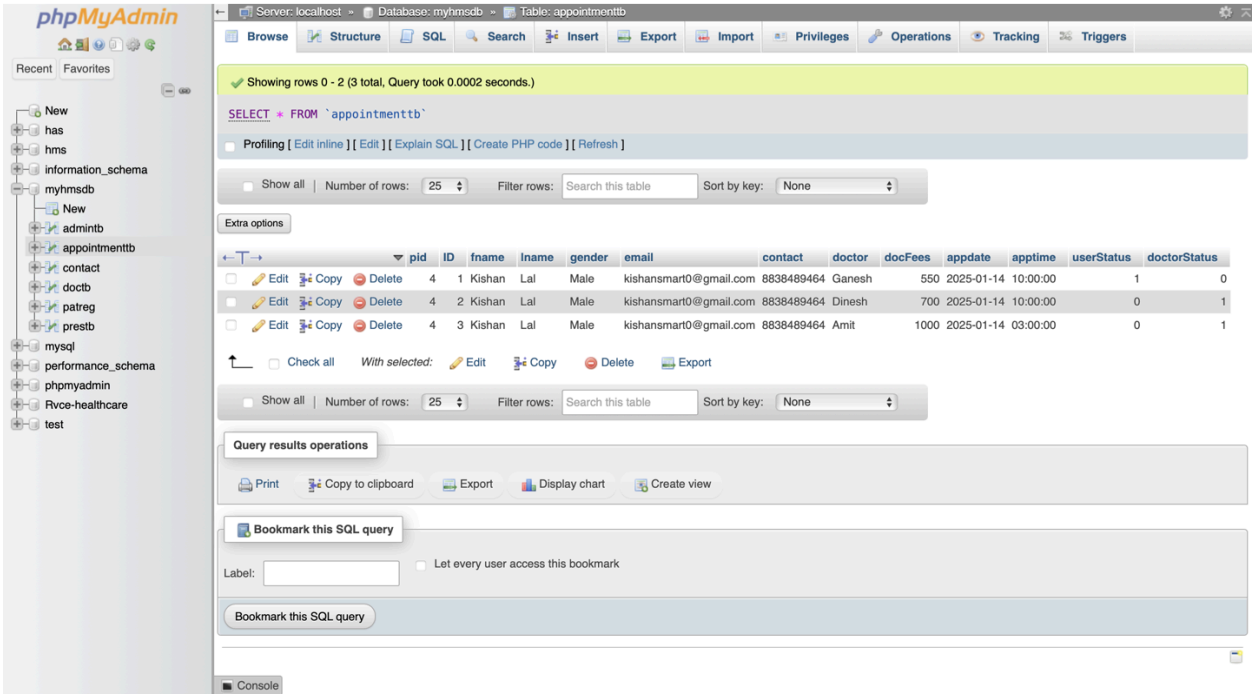
Booking an Appointment as Patient



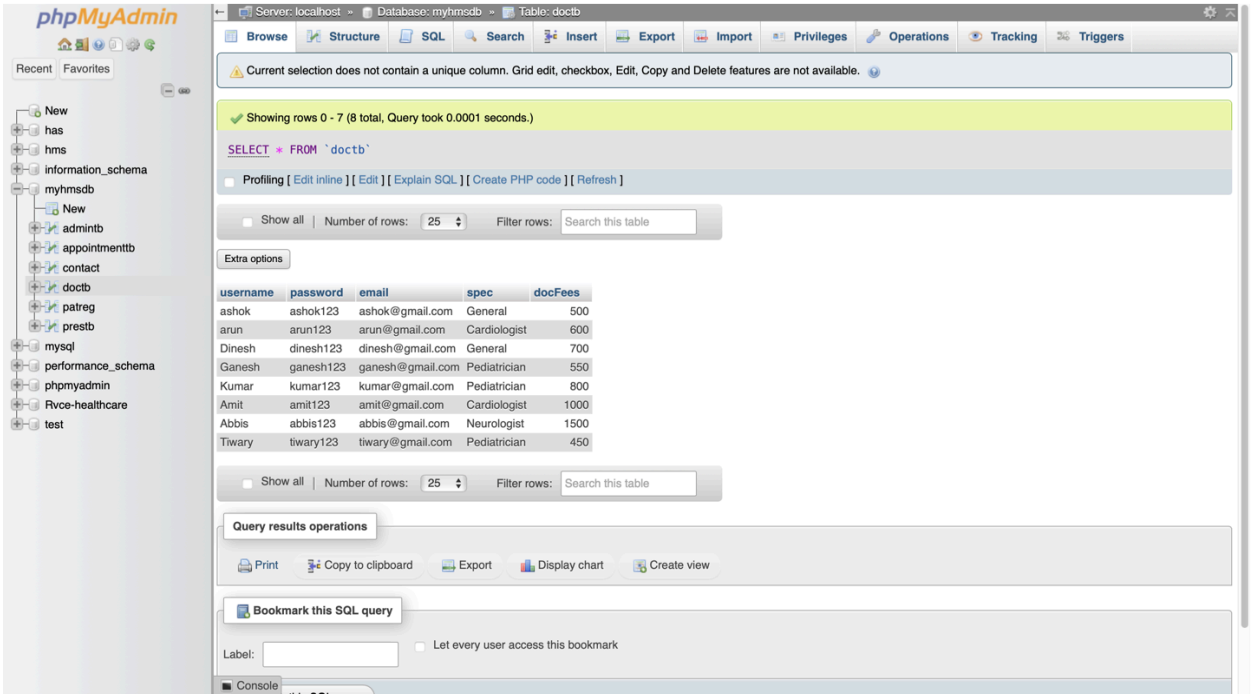
Prescription Page



Appointment View



Doctor View


phpMyAdmin
Recent Favorites
New
has
hms
information_schema
myhmsdb
New
admintb
appointmenttb
contact
doctb
patreg
prestb
mysql
performance_schema
phpmyadmin
Rvce-healthcare
test
Server: localhost » Database: myhmsdb » Table: doctb
Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers
Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
Showing rows 0 - 7 (8 total, Query took 0.0001 seconds.)
SELECT * FROM `doctb`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options
username password email spec docFees
ashok ashok123 ashok@gmail.com General 500
arun arun123 arun@gmail.com Cardiologist 600
Dinesh dinesh123 dinesh@gmail.com General 700
Ganesh ganesh123 ganesh@gmail.com Pediatrician 550
Kumar kumar123 kumar@gmail.com Pediatrician 800
Amit amit123 amit@gmail.com Cardiologist 1000
Abbis abbis123 abbis@gmail.com Neurologist 1500
Tiwary tiwary123 tiwary@gmail.com Pediatrician 450
Show all | Number of rows: 25 Filter rows: Search this table
Query results operations
Print Copy to clipboard Export Display chart Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Console