Developing a Cloud-Based Application

Salam yaser ahmed Hammad 220201765
Hanan naser abdallah Abu Shaban 220191395
Malak ghassan basem Abu Sharar 220203511
Aya yaser Alkahloot 220200236
Software Development Department
Faculty of Information Technology
Islamic University of Gaza

A Requirement for the Course Advanced Software Engineering (SDEV 4304)
Instructor: Dr. Rebhi S. Baraka

Abstract

We developed a cloud-based healthcare management system (HMS) that is designed to streamline and automate the management of healthcare services. HMS offers a user-friendly interface for administrators, doctors, and patients to efficiently manage and track healthcare activities. It facilitates patient registration, uploading medical records, appointment scheduling, and interactive communication tools. With features like unique patient ID generation, appointment conflict resolution, and secure cloud-based data management, HMS ensures scalability and accessibility for healthcare providers of all sizes.

The system also incorporates advanced functionalities tailored to the roles of each user. Patients benefit from easy access to their medical history and seamless communication with healthcare providers. Doctors are equipped with tools to manage patient appointments, issue prescriptions, and send notifications, ensuring an efficient workflow. Administrators oversee the system's operations, including account management, billing, and secure data updates. Built on a robust SQL database structure, the HMS leverages modern cloud computing methodologies to offer a scalable, secure, and efficient solution. With integrated analytics, user activity tracking, and a well-documented deployment process, the system represents a comprehensive approach to improving healthcare management.

1. Introduction

The cloud-based Healthcare Management System (HMS) is a web application designed to streamline and automate the management of healthcare services. It serves as a centralized platform for patients, doctors, and administrators to collaborate efficiently, manage medical records, and track healthcare activities. HMS provides essential features such as patient registration, appointment scheduling with conflict resolution, uploading medical documents (e.g., lab reports and CT scans), and secure communication tools like notifications and emails. The system is tailored to address the needs of all user roles, ensuring an integrated and user-friendly experience.

To develop HMS, we followed a cloud-oriented software methodology, leveraging agile practices for iterative development, frequent feedback, and continuous collaboration. This approach enabled us to adapt to evolving system requirements while ensuring timely delivery of key features. The methodology focused on meeting the specified requirements, including secure cloud-based data storage, appointment management, and user activity tracking. Additionally, the deployment of HMS on a scalable and secure cloud platform ensures high availability, flexibility, and accessibility. By adhering to these practices, HMS effectively meets the demands of modern healthcare management, providing a reliable and comprehensive solution for administrators, doctors, and patients alike.

2. Cloud Software Application/Service Requirements

The cloud-based Healthcare Management System (HMS) is a cutting-edge web application designed to enhance the efficiency, scalability, and accessibility of healthcare services. The system serves as a centralized platform that streamlines processes for patients, doctors, and administrators. By leveraging secure cloud-based infrastructure, HMS ensures seamless collaboration among all users and provides robust tools for managing healthcare activities.

The application addresses key functionalities, including patient registration, secure storage of medical documents (e.g., lab reports and CT scans), conflict-free appointment scheduling, and real-time communication through notifications and emails. Each feature has been meticulously designed to meet the specific requirements outlined in the project specifications. For example, unique patient ID generation ensures a secure and streamlined identification process, while the administrator's exclusive data update rights guarantee data integrity and security. HMS also tracks all user activities to maintain transparency and provides analytics to support informed decision-making for healthcare providers.

HMS's User Stories:

For Patients:

1. Patient Registration

As a patient, I want to register on the platform and provide my medical details to access healthcare services.

Acceptance Criteria:

- The registration process should be user-friendly and intuitive.
- Patients should be able to provide necessary personal details and upload medical documents (e.g., lab reports, CT scans).
- Upon successful registration, patients should receive a unique ID and a confirmation notification.

2. **Appointment Scheduling**

As a patient, I want to schedule appointments with doctors without conflicts to receive timely medical care.

Acceptance Criteria:

- Patients should be able to select doctors based on specialty and availability.
- The system should resolve scheduling conflicts automatically to avoid overlapping appointments.
- Patients should receive notifications confirming their appointments.

3. **Medical Document Upload**

As a patient, I want to upload my medical documents so that doctors can access them for consultations.

Acceptance Criteria:

- Patients should be able to upload documents in various formats (e.g., PDF, images, videos).
- The system should securely store documents in the patient's profile and notify them of successful upload.

4. Notifications

As a patient, I want to receive notifications about appointment confirmations, approvals, and updates to stay informed.

Acceptance Criteria:

- Notifications should be sent via email and in-app messaging for all system events.

For Doctors:

1. **Subscription and Registration**

   As a doctor, I want to subscribe and register by providing my necessary information to access the system's features.

Acceptance Criteria:

- Doctors must provide personal and professional details (e.g., license number, specialty).
- Registration is subject to administrator approval.

2. **Specialty-Based Classification**

   As a doctor, I want my specialty to be recorded during registration so that patients can find the right doctor.

Acceptance Criteria:

- Doctors select their specialty (e.g., Dermatologist, Nephrologist) during registration.
- The system stores and displays specialties for patient searches and filtering.

3. **Generate Prescriptions**

   As a doctor, I want to generate and share prescriptions for my patients to ensure they receive proper treatment.

Acceptance Criteria:

- Doctors can create prescriptions directly in the system.
- Prescriptions are securely stored and shared with patients.

4. **Patient Management**

   As a doctor, I want to view and manage my assigned patients to provide effective medical care.

Acceptance Criteria:

- Doctors should have access to a dashboard displaying detailed patient profiles and medical histories.
- Tools should be available for doctors to add notes, prescribe treatments, and manage follow-ups.
- The system should allow doctors to track patient progress and update treatment plans.

4. **Notifications to Patients**

As a doctor, I want to send notifications to my patients to provide updates, reminders, or prescriptions.

Acceptance Criteria:

- Doctors should be able to compose and send notifications directly through the system.
- Notifications should be delivered securely via email .

5. **Appointment Management and History**

As a doctor, I want to view and manage appointments with my patients to organize my schedule efficiently and I want to access the history of appointments with my patients for better treatment planning.

Acceptance Criteria:

- Doctors should be able to view scheduled appointments and reschedule them if necessary.
- Notifications should be sent to patients about any changes in appointments.
- The system displays a log of past appointments with notes and outcomes.
- Appointment history is searchable and linked to patient profiles.

6. **Cancel Appointment**

As a doctor, I want to cancel appointments if necessary to manage my schedule effectively.

Acceptance Criteria:

- Doctors can cancel appointments for patients and send a notification that the appointment has been cancelled.

For Administrators:

1. **Unique Patient ID Generation**
   As an administrator, I want to generate unique IDs for patients to ensure secure identification.

Acceptance Criteria:

- Unique patient IDs should be automatically generated upon successful registration.
- The IDs should be securely stored and linked to the corresponding patient profiles.

2. **Activity Tracking**
   As an administrator, I want to track and log all user activities to ensure transparency and maintain system integrity.

Acceptance Criteria:

- The system should log user activities, such as registrations, appointments, and document uploads, with timestamps.
- Administrators should be able to access activity logs for auditing purposes.

3. **Data Management**
   As an administrator, I want to manage and update data securely in the cloud to maintain system accuracy and security.

Acceptance Criteria:

- Only administrators should have the rights to modify or update cloud-stored data.
- Updates should be logged and verified for integrity.

4. **Billing Management**
   As an administrator, I want to handle patient billing to ensure accurate payment processing.

Acceptance Criteria:

- Administrators should be able to generate invoices and track payments.
- Billing information should be securely stored and accessible for auditing.

HMS's UML Diagram:

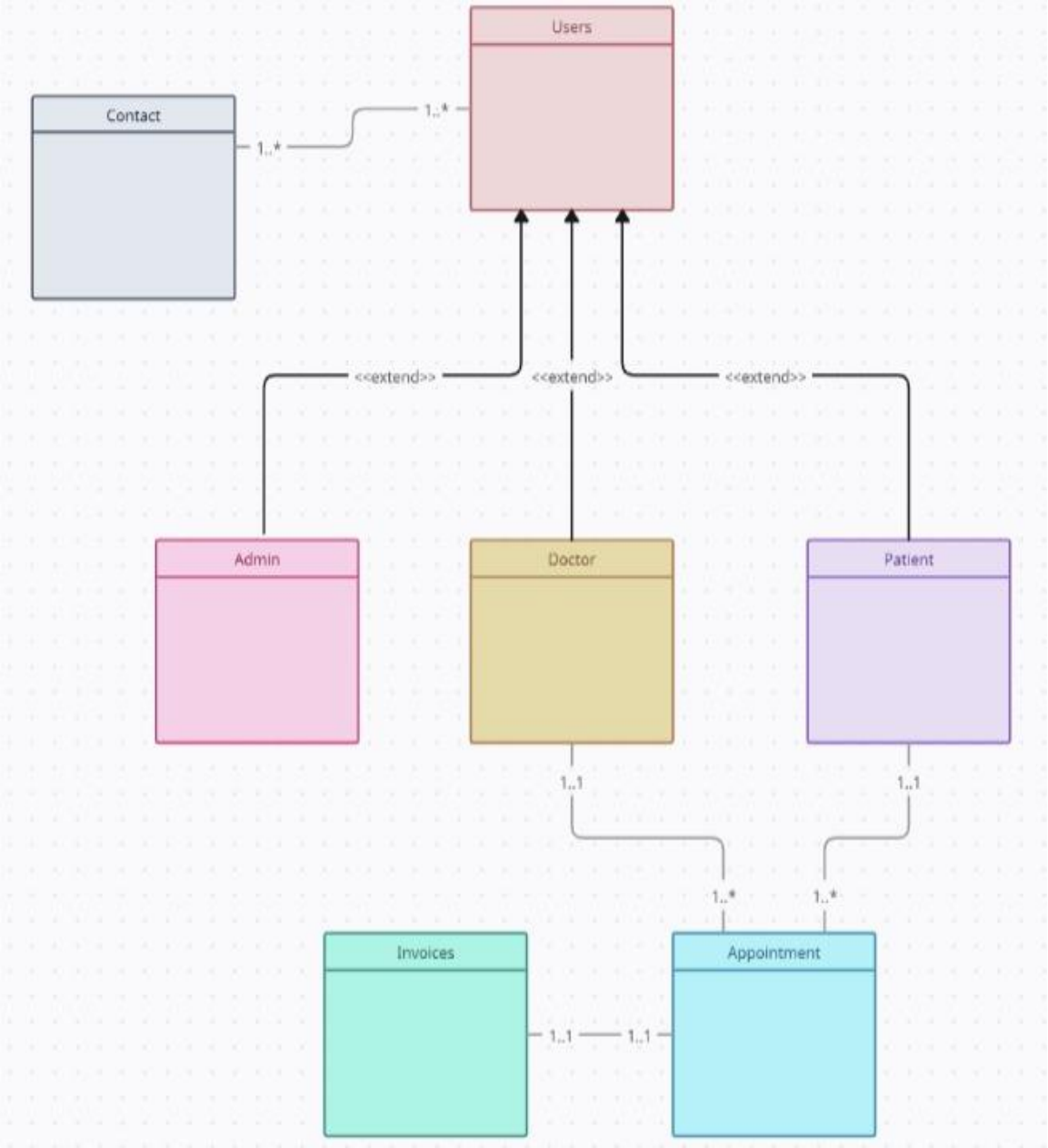


Figure 1- HMS's Componenr UML Diagram

HMS's Use Cases Scenario:

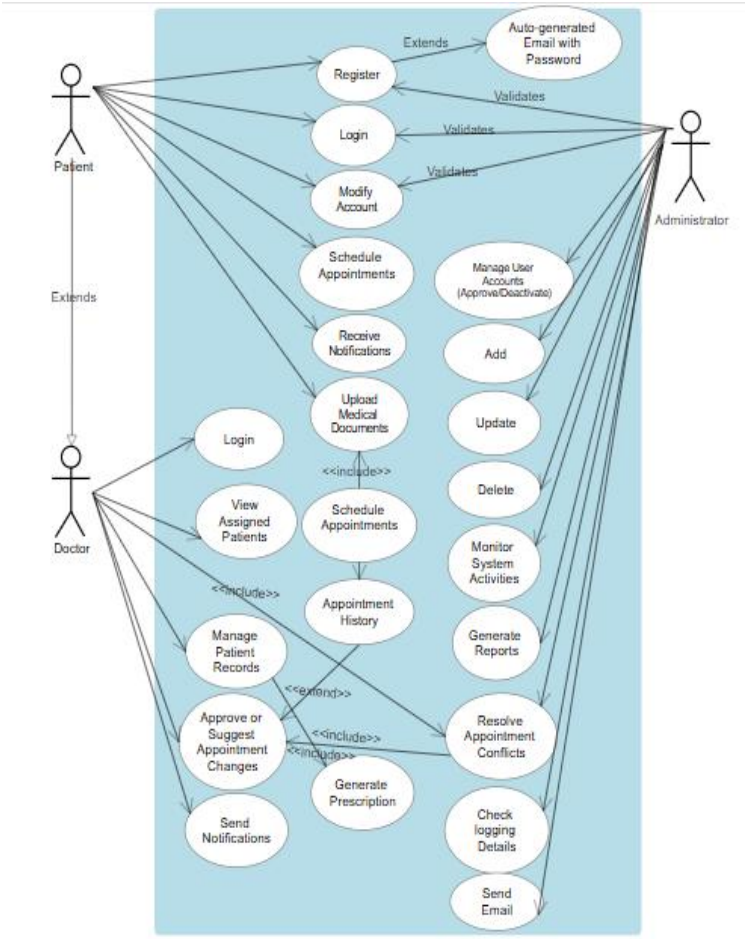


Figure 2- HMS's Use Case Scenario

Use Case 1: Patient Registration

Identifier	P01
Actors	Patient, Administrator, System
Goal	To enable patients to register and subscribe to the application.
Precondition	The patient accesses the application's registration page.
Postcondition	The patient is registered with a unique ID and can log in.
Main Scenario	1. Patient navigates to the registration page.
	2. Patient enters personal details and uploads necessary documents.
	3. System validates the entered information.
	4. Administrator verifies the documents.
	5. Administrator approves the registration.
	6. System generates a unique patient ID.
	7. Patient receives a confirmation email with login credentials.

Use Case 2: Upload Medical Documents

Identifier	P02
Actors	Patient, System
Goal	To allow patients to upload medical files for doctor access.
Precondition	The patient is logged into the system.
Postcondition	Files are uploaded and associated with the patient's profile.
Main Scenario	1. Patient navigates to the upload section.
	2. Patient selects files to upload (e.g., CT scans, lab results).
	3. System validates file formats and sizes.
	4. System stores files in the cloud database.
	5. Patient receives confirmation of a successful upload.

Use Case 3: Schedule Appointment

Identifier	P03
Actors	Patient, Doctor, System
Goal	To facilitate appointment scheduling between patients and doctors..
Precondition	The patient is logged in and has selected a doctor.
Postcondition	The appointment is scheduled without conflicts.
Main Scenario	1. Patient navigates to the appointment section.
	2. Patient selects a doctor and preferred time.
	3. System checks for scheduling conflicts.
	4. System confirms the availability.
	5. Appointment details are stored and notifications are sent to both parties.

Use Case 4: Administrator Account Management

Identifier	A01
Actors	Administrator, System
Goal	To manage patient and doctor accounts.
Precondition	The administrator is logged in.
Postcondition	Account data is updated securely.
Main Scenario	1. Administrator accesses the account management dashboard.
	2. Administrator views the list of users (patients/doctors).
	3. Administrator selects a user account to update.
	4. Administrator edits account details and saves changes.
	5. System updates the data in the cloud database.

Use Case 5: Doctor Subscription

Identifier	D01
Actors	Doctor, Administrator, System
Goal	To enable doctors to register and access the application.
Precondition	The doctor accesses the log in page.
Postcondition	The doctor is registered and can log in to the application.
Main Scenario	1. Doctor navigates to the log in page.
	2. Doctor enters personal details and specialization.
	3. System validates the entered information.
	4. Administrator verifies the details provided.
	5. Administrator approves the registration.

Use Case 6: Manage Patient Records

Identifier	D02
Actors	Doctor, System
Goal	To allow doctors to view and manage their patient records.
Precondition	The doctor is logged into the system.
Postcondition	The patient records are accessed and updated as necessary.
Main Scenario	1. Doctor logs into the system.
	2. Doctor navigates to their patient list.
	3. Doctor selects a patient's record.
	4. Doctor reviews the medical history and uploads new diagnoses.
	5. System updates the patient's record in the database.

Use Case 7: Cloud Data Backup and Security

Identifier	S01
Actors	System
Goal	To ensure data security and periodic backups.
Precondition	The system is operational.
Postcondition	Data integrity is maintained, and backups are securely stored.
Main Scenario	1. System runs a scheduled backup task.
	2. Data is encrypted and uploaded to secure storage.
	3. Backup logs are generated and stored.
	4. Administrator is notified of successful backups.

Use Case 8: Notification Management

Identifier	S02
Actors	System, Patient, Doctor
Goal	To send timely notifications to users.
Precondition	A trigger event occurs (e.g., appointment creation).
Postcondition	Notifications are sent and acknowledged by users.
Main Scenario	1. Event occurs (e.g., appointment scheduled).
	2. System generates a notification message.
	3. System sends notifications via email and in-app.
	4. Users acknowledge the notification.

Requirements specifications for Healthcare Management System (HMS) :

Patient Functionality:

- Patients should be able to register on the platform and provide necessary personal and medical details.
- Patients should be able to upload required medical documents (e.g., lab reports, CT scans, prescriptions).
- Patients should be able to schedule appointments with doctors based on specialty and availability.
- Patients should receive notifications regarding their appointment status, document uploads, and updates from doctors.

Doctor Functionality:

- Doctors should be able to register and provide necessary information for accessing the system.
- Doctors should be able to view and manage assigned patients, including access to their medical records and uploaded documents.
- Doctors should manage patient care by prescribing treatments, updating progress, and scheduling follow-ups.
- Doctors should use a dashboard to manage appointments, reschedule if needed, and send notifications or emails to patients.

Administrator Functionality:

- Administrators should have access to a dashboard for reviewing and approving patient registrations and able to generate unique IDs for each registered patient.
- Administrators should be able to update and manage patient and doctor data stored in the system while tracking and logging all user activities for auditing and reporting purposes.
- Administrators should handle billing processes, including invoice generation and payment tracking.

System Functionality:

- The system should store and retrieve user data securely using a combination of SQL databases.
- The system should provide appointment management features for patients and doctors, including conflict resolution.
- The system should send notifications and emails to users based on specific events (e.g., registration approval, appointment confirmations).
- The system should record all user activities.
- The system should ensure data integrity and allow only authorized administrators to update records.

Performance and Security:

- The application should provide a seamless and responsive user experience across all devices.
- The system should be scalable to handle a large number of users and data efficiently.
- Appropriate security measures (e.g., encryption, access control) should be implemented to protect user data and ensure privacy.
- Regular data backups should be performed to prevent data loss.

Healthcare Management System (HMS) Works at Various Levels of Abstraction:

1. User Interaction Layer (Frontend):

The frontend provides the interface through which patients, doctors, and administrators interact with the Healthcare Management System (HMS). It is designed to be simple, responsive, and user-friendly to ensure a seamless experience for all users.

- **Patients:**
  Patients access the system through a web interface where they can register, schedule appointments, upload medical documents (e.g., lab reports, CT scans), and receive notifications.

- **Doctors:**
  Doctors use dashboards to view and manage patient records, track appointments, prescribe treatments, and send notifications.

- **Administrators:**
  Administrators manage user accounts, monitor system activities and handle billing. They have access to specialized administrative panels.

Technology Stack:

- HTML5 and CSS3 for structuring and styling the web pages.
- JavaScript for client-side interactivity.
- Bootstrap for creating responsive and consistent designs.

2. Application Logic Layer (Backend):

The backend is responsible for processing all business logic, handling user requests, and managing communication with the database.

- **Core Functionalities:**
  - Scheduling appointments with conflict resolution to avoid overlaps.
  - Verifying patient registration and handling secure authentication.
  - Managing role-based access to features for patients, doctors, and administrators.
  - Handling notifications for events like appointment confirmations and updates.

- **APIs:**
  PHP scripts are used to handle requests from the frontend and interact with the database for CRUD operations.

3. Data Management Layer :

The data layer ensures the secure storage and management of healthcare data, supporting both structured and unstructured formats.

- **Structured Data (SQL):**
  A relational database (e.g., MySQL) is used to store patient details, appointment schedules, medical records metadata.

- **Security Measures:**
  - All data is encrypted during storage and transit.
  - Role-based access controls restrict unauthorized modifications.

4. Cloud Infrastructure Layer:

The cloud infrastructure ensures the scalability, reliability, and availability of the HMS, meeting the needs of both small clinics and larger healthcare organizations.

- **Hosting:**
  The application is hosted on a cloud platform and a secure hosting provider, ensuring robust uptime and performance.

- **Load Balancing:**
  Traffic is distributed across servers to handle peak loads and ensure smooth operation.

- **Backup and Recovery:**
  Automated backups are scheduled to protect against data loss, and disaster recovery mechanisms are in place for quick restoration.

3. Software Architecture and Design

Based on the requirements of the Healthcare Management System (HMS), we decided to build a three-tier architecture system that separates the application into three distinct layers for enhanced scalability, maintainability, and performance.

The architecture includes:

1. Presentation Tier (Frontend):

   Also known as the frontend, this layer is responsible for handling the user interface and displaying information to the users. It consists of an HTML-based frontend enhanced with CSS, JavaScript, and Bootstrap, enabling patients, doctors, and administrators to interact with HMS effectively.

2. Application Tier (Backend):

   Also known as the backend, this layer encompasses the business logic and application processing. It includes a PHP-based backend that processes requests received from the frontend, interacts with the MySQL database, and returns responses to the users.

3. Data Tier (Database):

   Also known as the database tier, this layer is responsible for storing and managing the data required by HMS. A MySQL database is used to store and retrieve data related to patients, doctors, administrators, appointments, and the overall system.

This three-tier design ensures modularity, scalability, and secure data handling, aligning with the requirements of the Healthcare Management System.

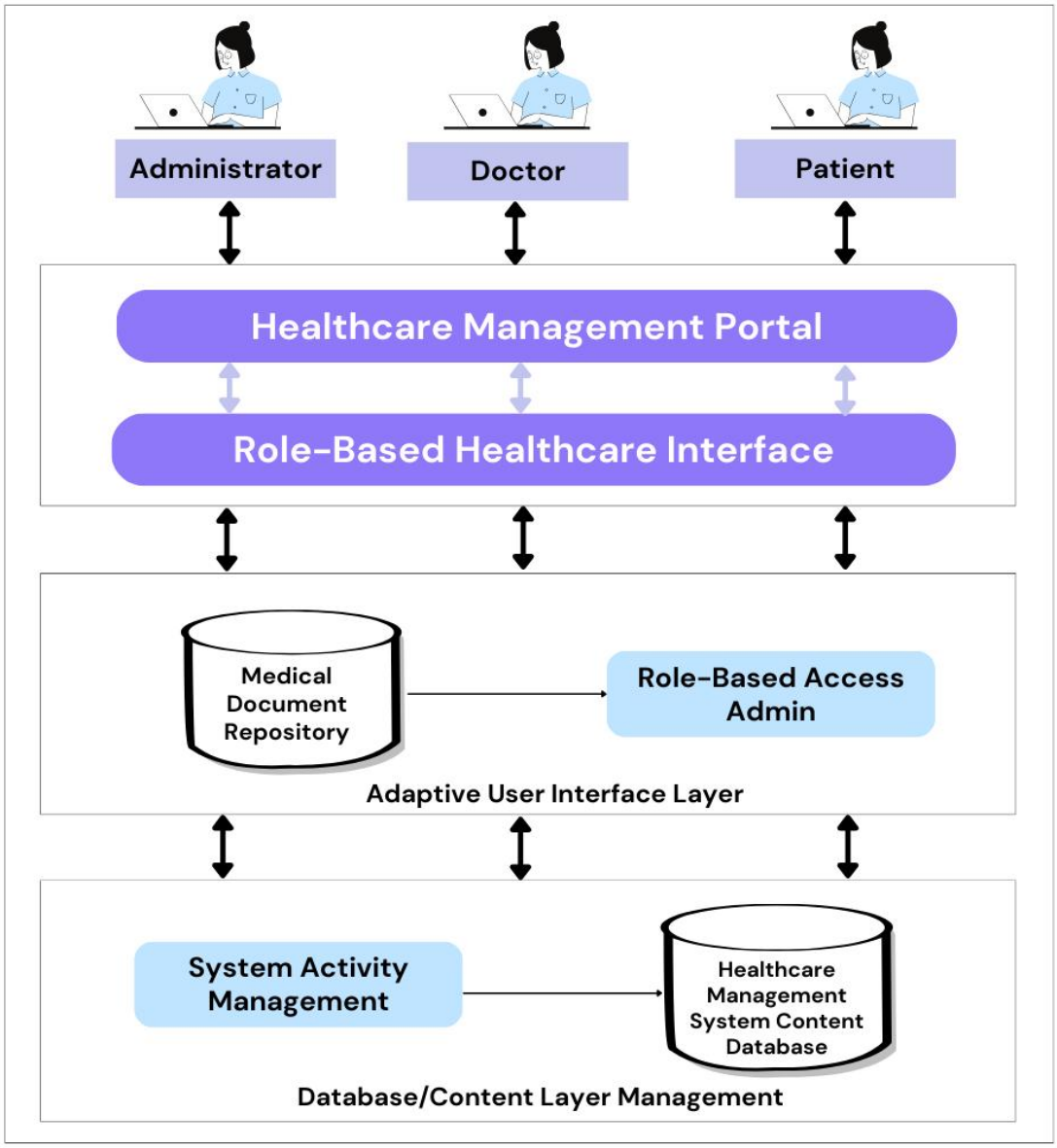


Figure 3- Abstract View of HMS's Architecture

Functional Components of HMS:

- The Healthcare Management System (HMS) provides user registration and login functionality to allow patients, doctors, and administrators to access their respective accounts securely.
- **Patients** can register on the system, upload medical documents such as lab reports and CT scans, view their medical history, and schedule appointments with their preferred doctors. They also receive notifications for appointment confirmations, reminders, and updates.

- **Doctors** have the ability to view and manage patient profiles, track appointments, prescribe treatments, and send notifications to patients. They can also update treatment plans and resolve scheduling conflicts.

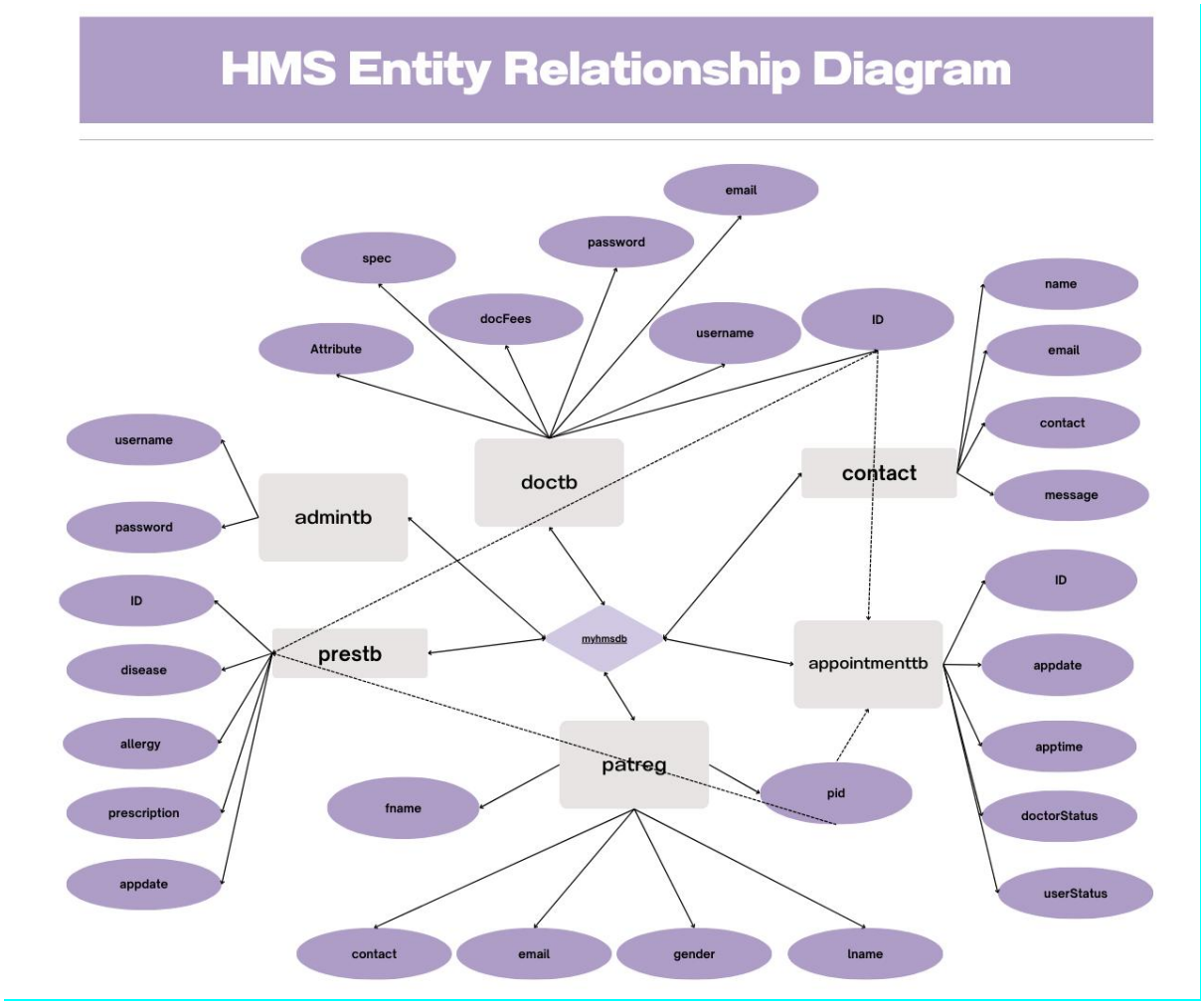
- **Administrators** manage user accounts, monitor activities across the system. They ensure that patient data is securely stored and accessed only by authorized users.
- HMS integrates with external services for email notifications and data security protocols to protect sensitive healthcare information.
- HMS supports real-time scheduling functionality, ensuring that any conflicts in doctor or patient availability are automatically resolved.

Real-World Constraints of AWS for HMS:

In AWS, deploying resources such as EC2 instances comes with its own set of challenges. By default, EC2 instances are assigned a public IPv4 address, but configuring the public IPv4 DNS address requires manual intervention through the AWS Management Console. This process can be complex and time-consuming, particularly for developers who are less familiar with AWS's configuration settings.

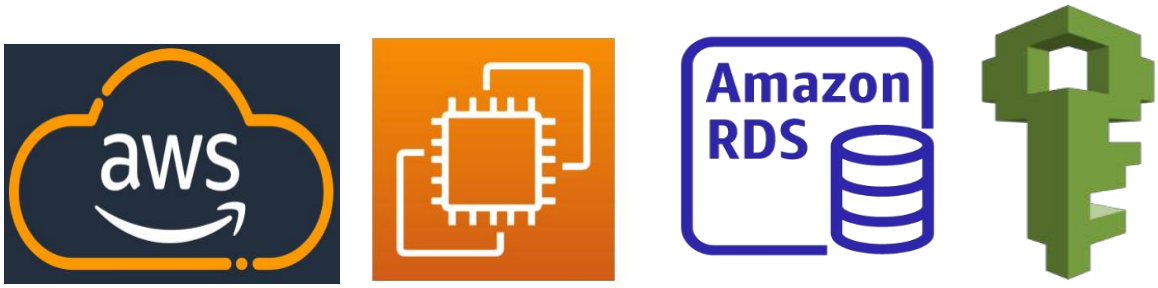
Additionally, managing instance-level settings, such as securing communication channels and configuring custom domains, may involve multiple steps that require a deep understanding of networking and AWS services. For example, while AWS provides Elastic Load Balancing and security groups for managing traffic and access, properly setting them up without disrupting the application's availability requires careful attention.

We believe that automating certain configurations, such as enabling DNS by default or offering simplified templates for common use cases, could make AWS more accessible for developers. Such improvements would streamline deployment workflows, enhance usability, and reduce the risk of misconfigurations for healthcare management systems like HMS.



HMS's Entity Relationship Diagram

4. Used Cloud Services and its Interfaces



AWS EC2 Instance **AWS IAM**

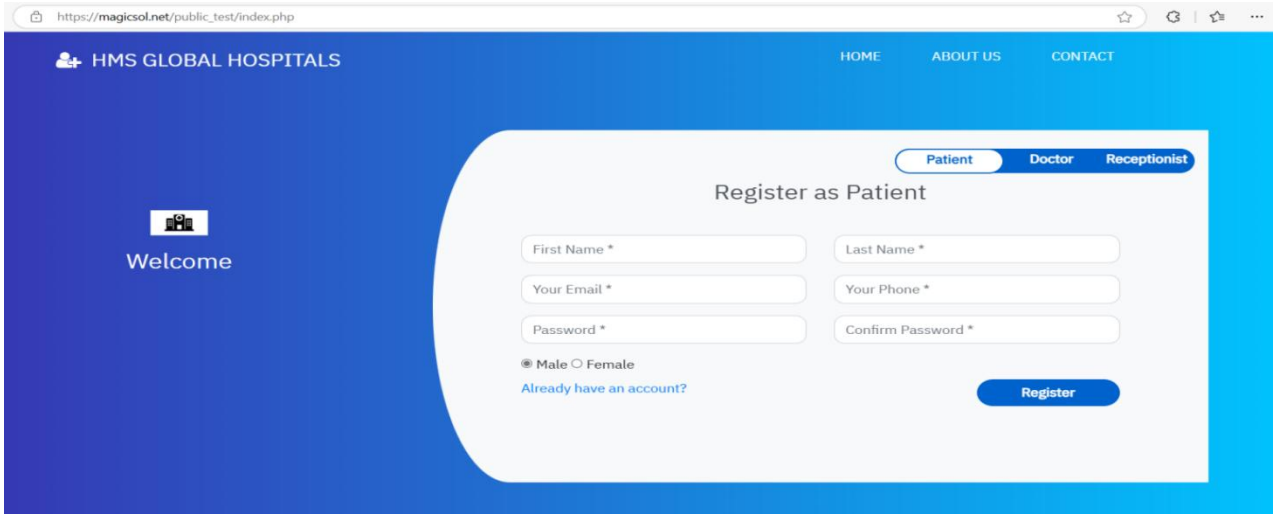
We utilized a range of AWS services to support the implementation and deployment of our system. These services included Amazon Relational Database Service (RDS), Amazon Elastic Compute Cloud (EC2) instances, security groups, and AWS Identity and Access Management (IAM). These services provided a reliable infrastructure for hosting, managing, and securing the application.

We leveraged the RDS service as our managed database solution, enabling us to store and access relational MySQL data efficiently and securely. The EC2 instances were employed to host the application, allowing us to deploy the components in a scalable and flexible environment that can handle varying workloads.

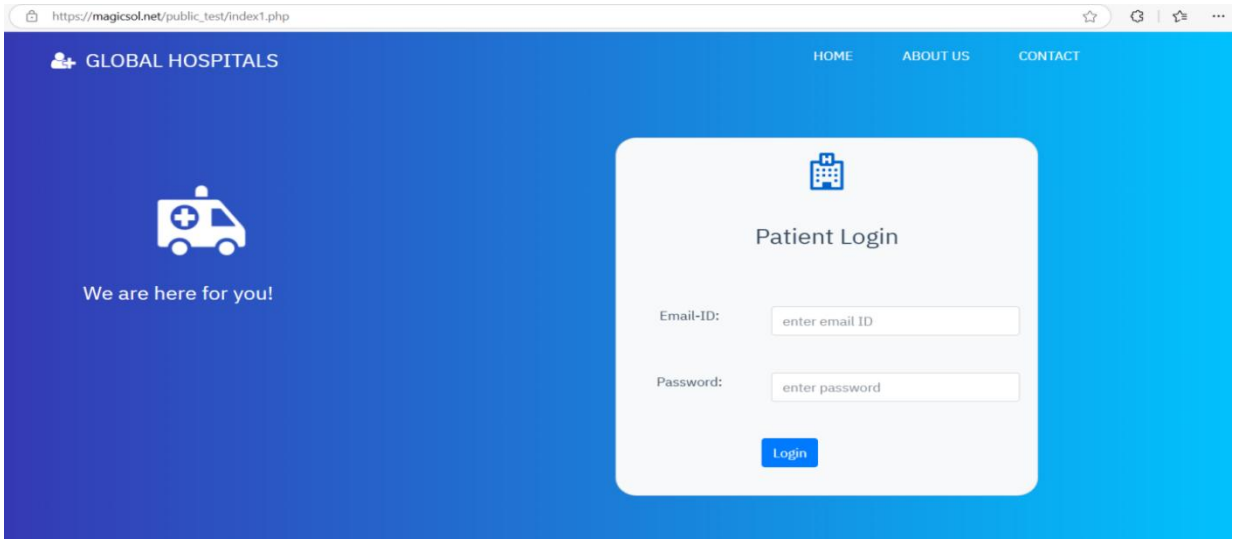
To ensure secure access, security groups were configured to manage inbound and outbound traffic, acting as virtual firewalls to control access to the instances and database. Additionally, AWS IAM was utilized to manage user roles and permissions, ensuring proper access control and authentication for various system users, including administrators and doctors.

These AWS services collectively provided a scalable, secure, and reliable foundation for deploying and managing our system in the cloud environment. This architecture ensures efficient operation, robust data handling, and seamless communication within the application.

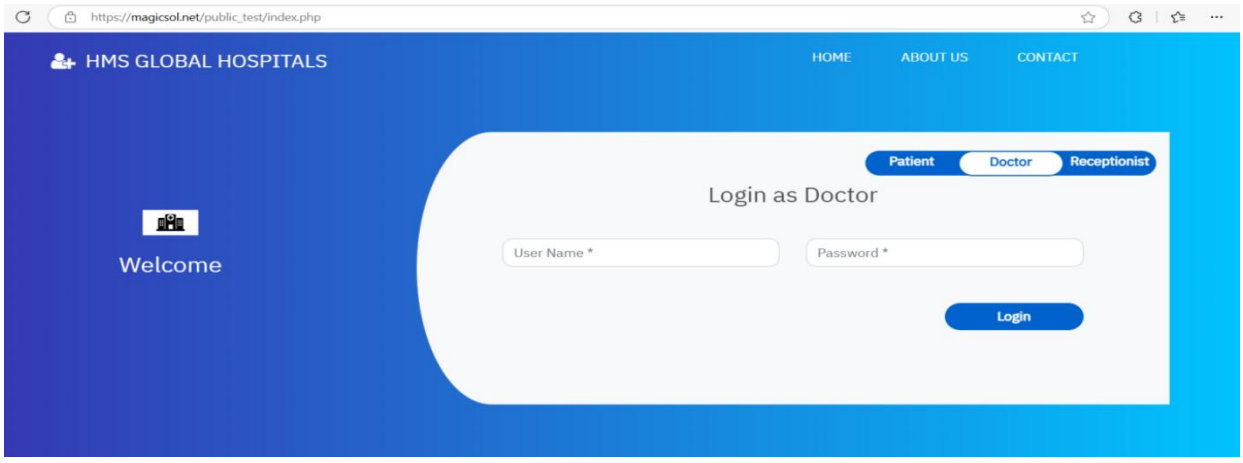
User Interfaces:



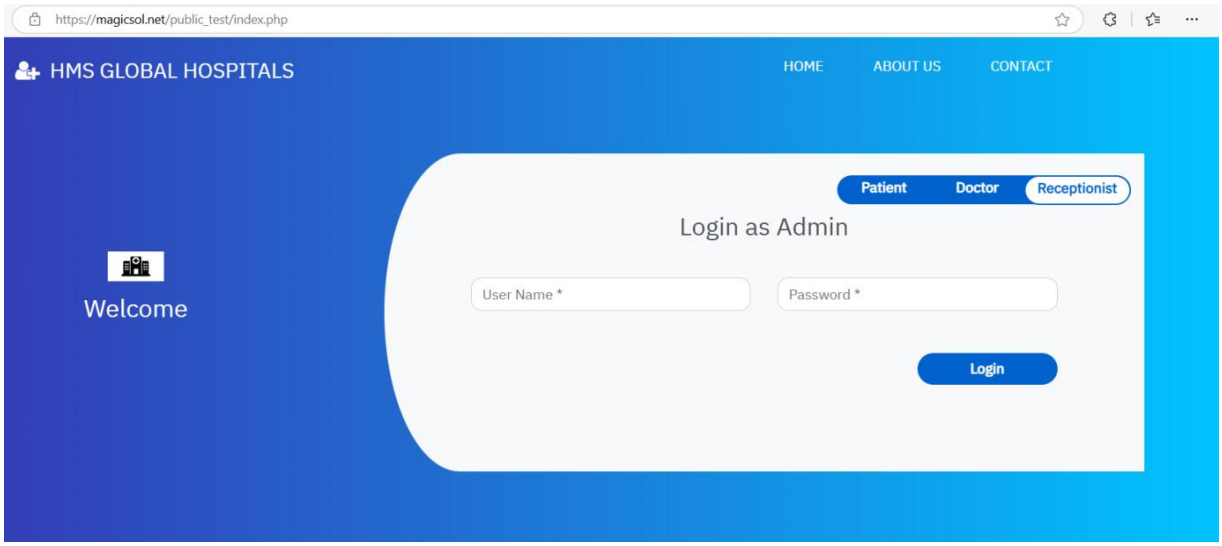
The Register Interface for Patient



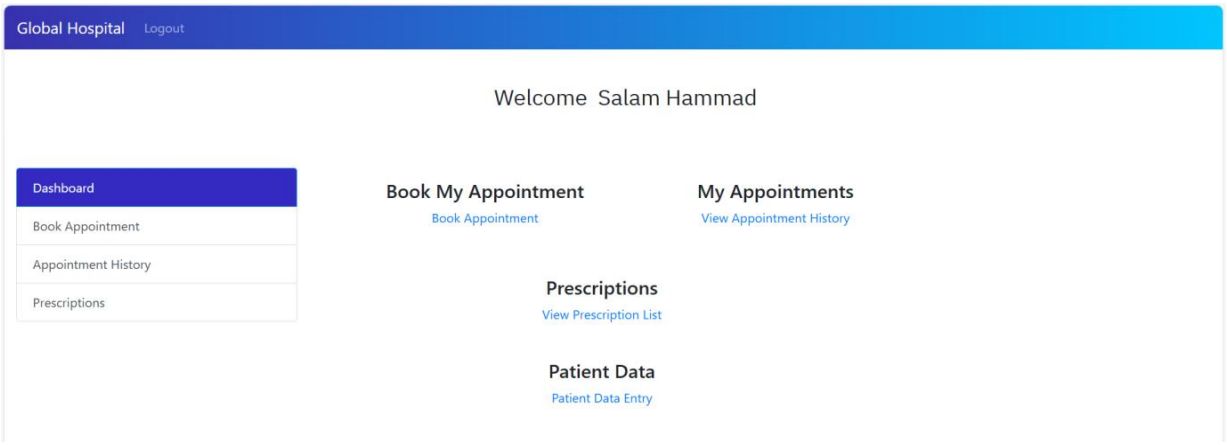
The Login Interface for Patient



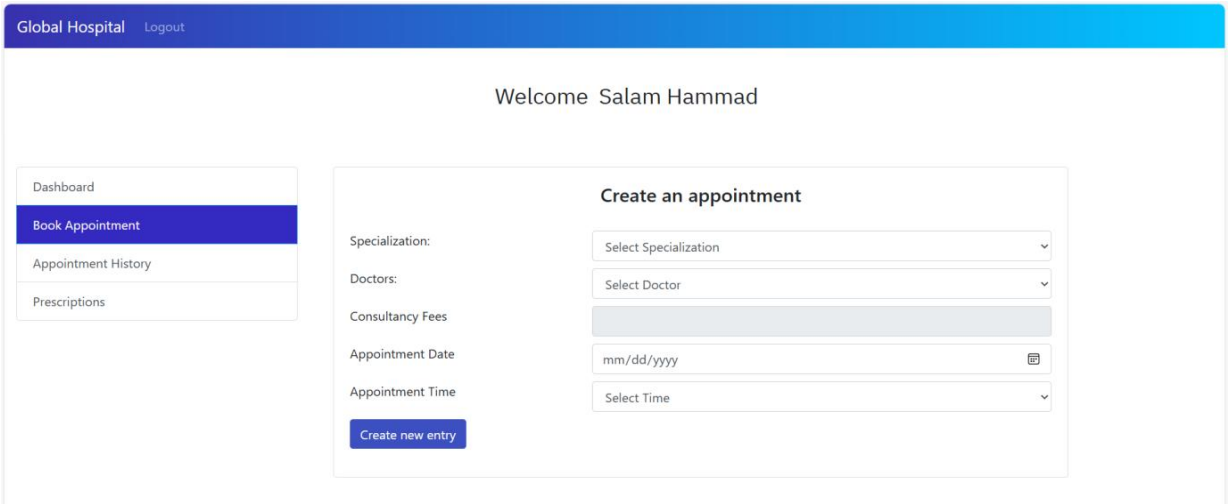
The Login Interface for Doctor



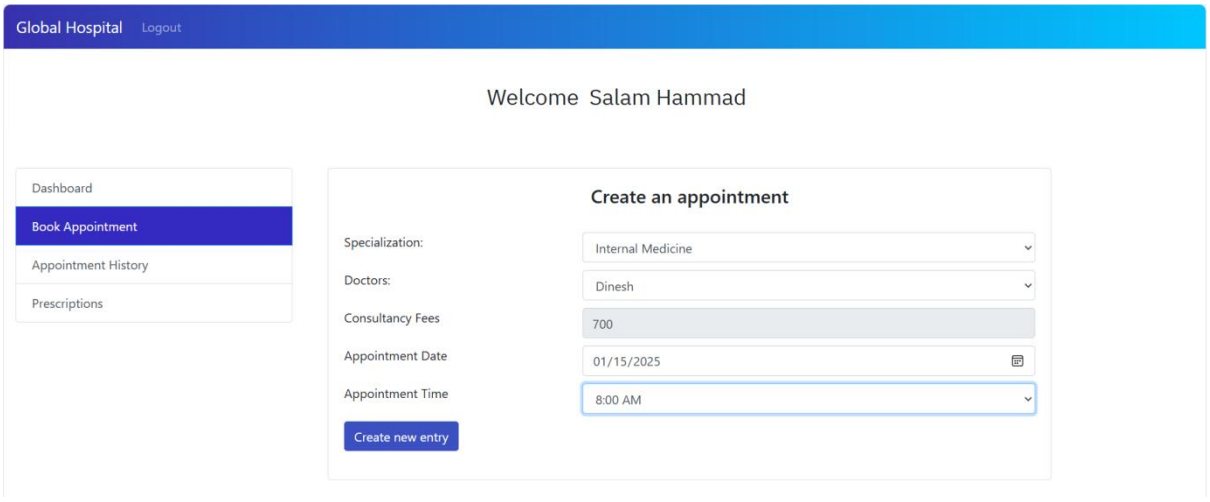
The Login Interface for Admin

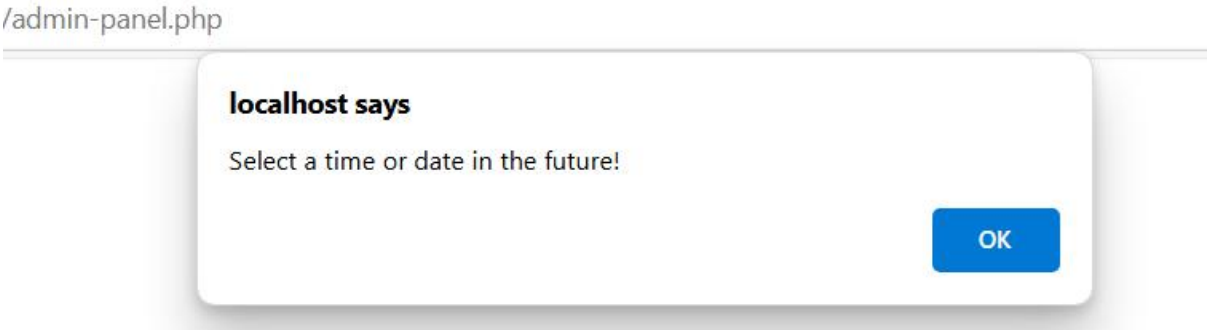


Patient Dashboard

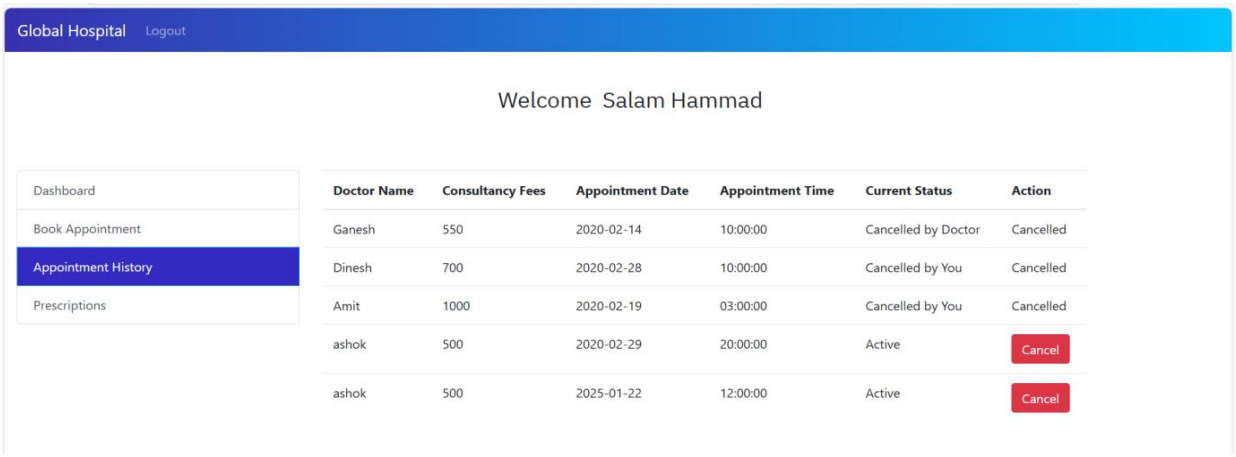


Patient Appointment Booking Page

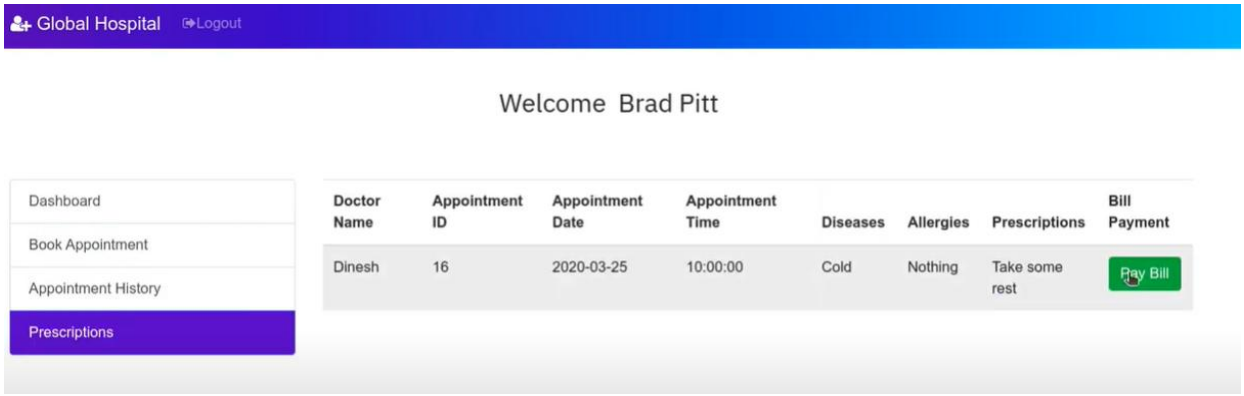




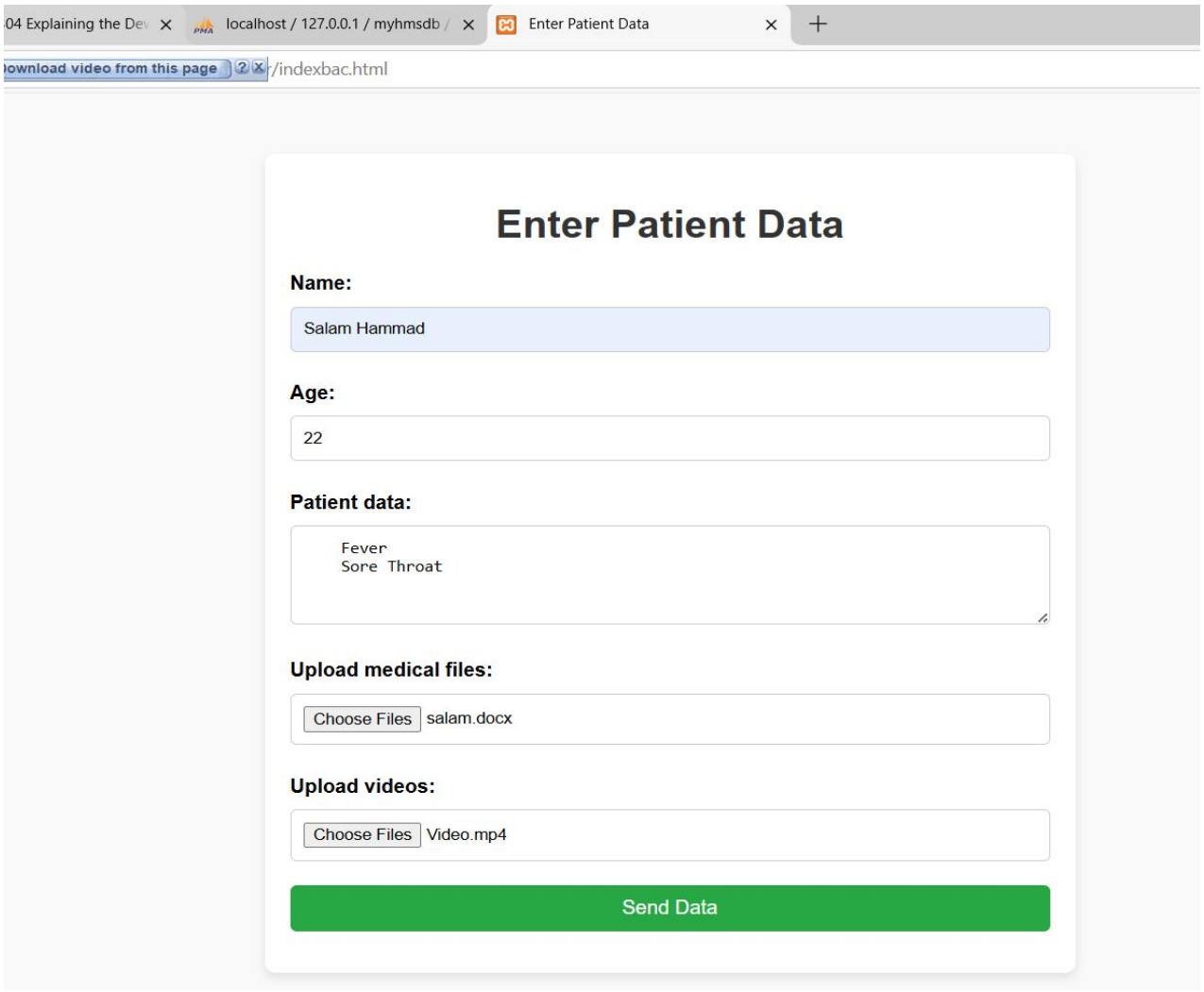
Select a time or date in the future!



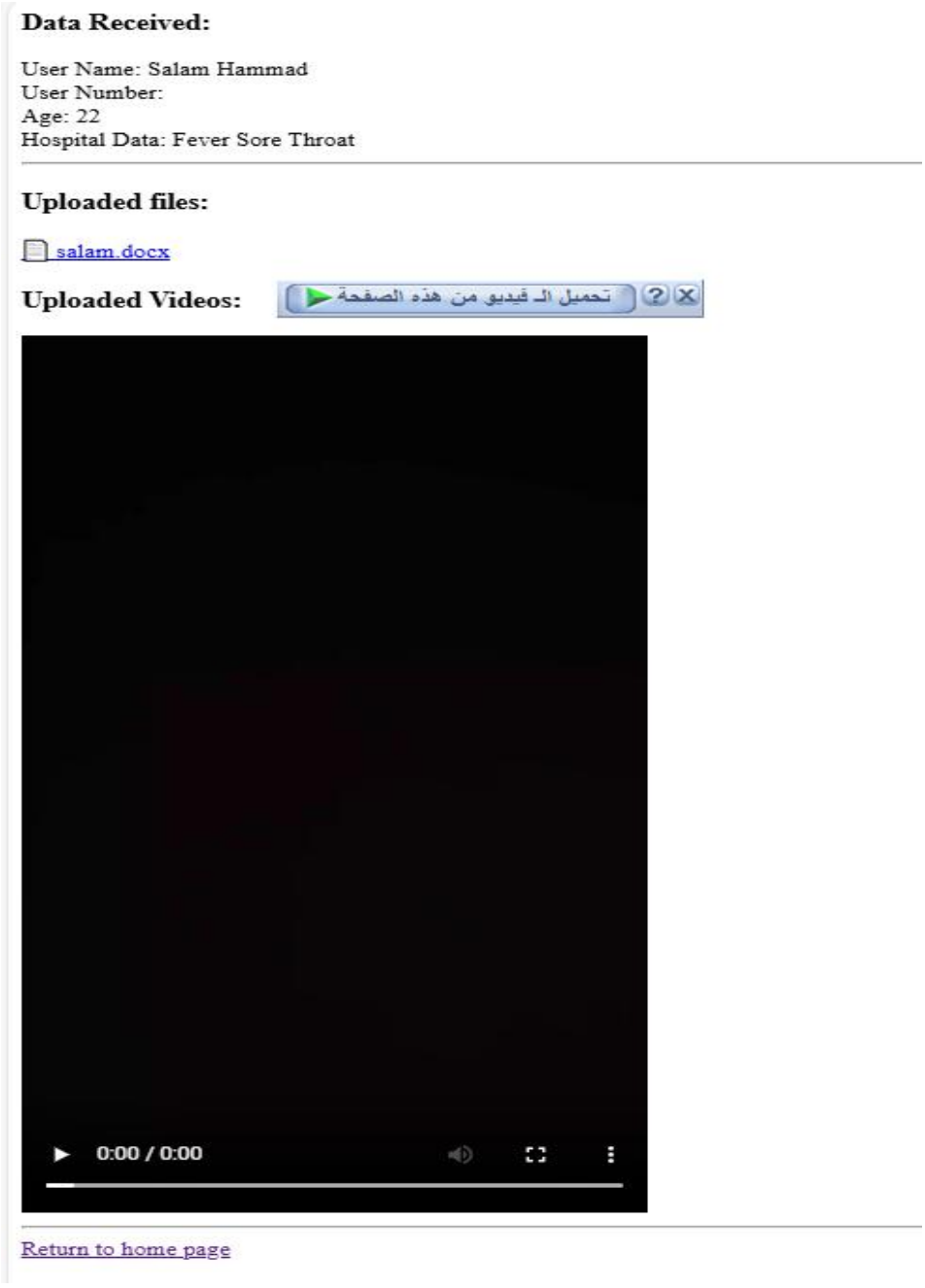
Patient Appointment History



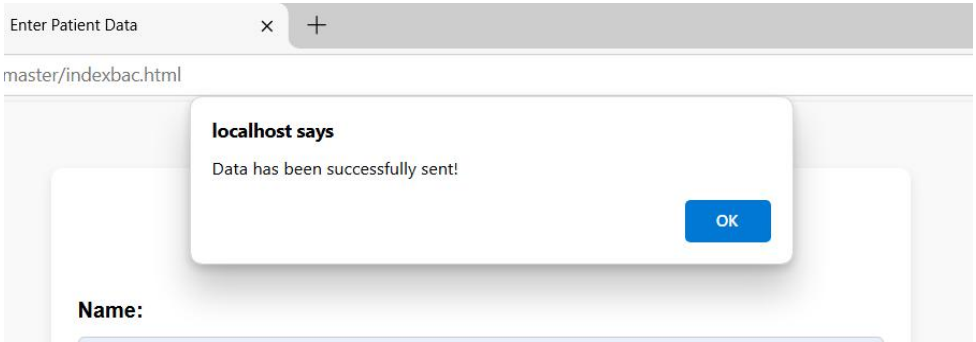
Prescription for the patient



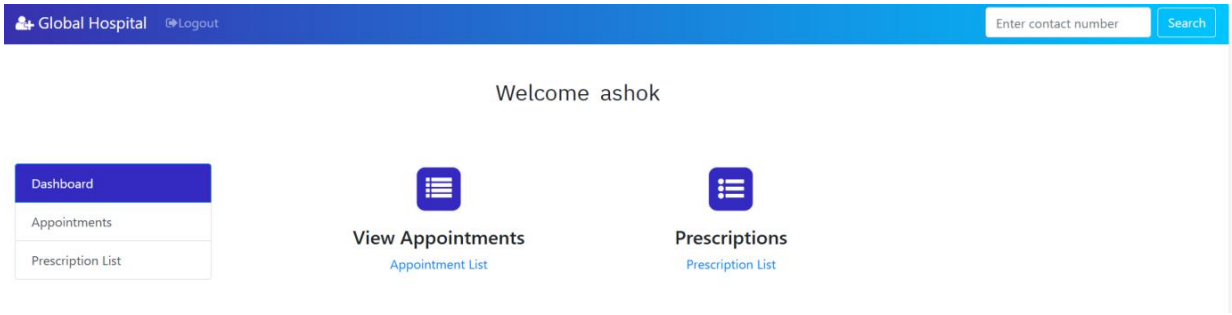
Patient Information Entry Interface



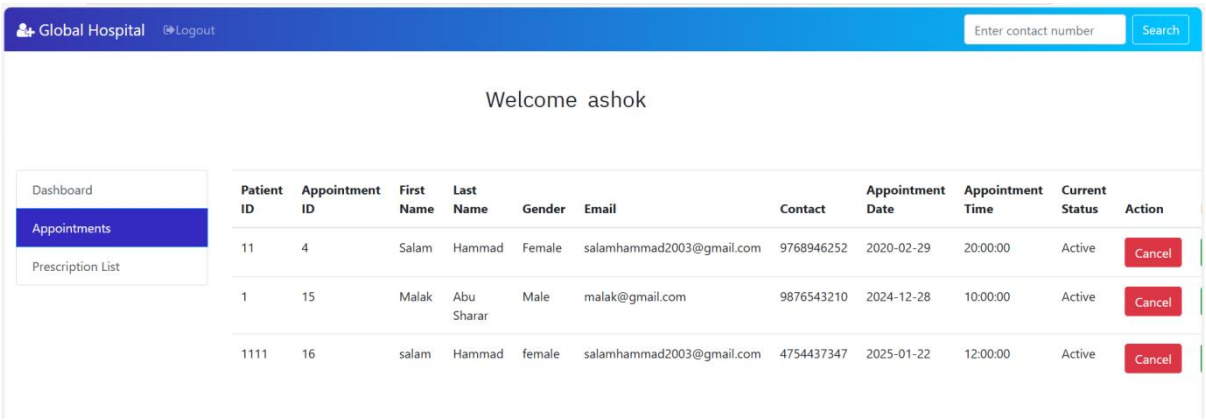
The interface after entering the patient's data



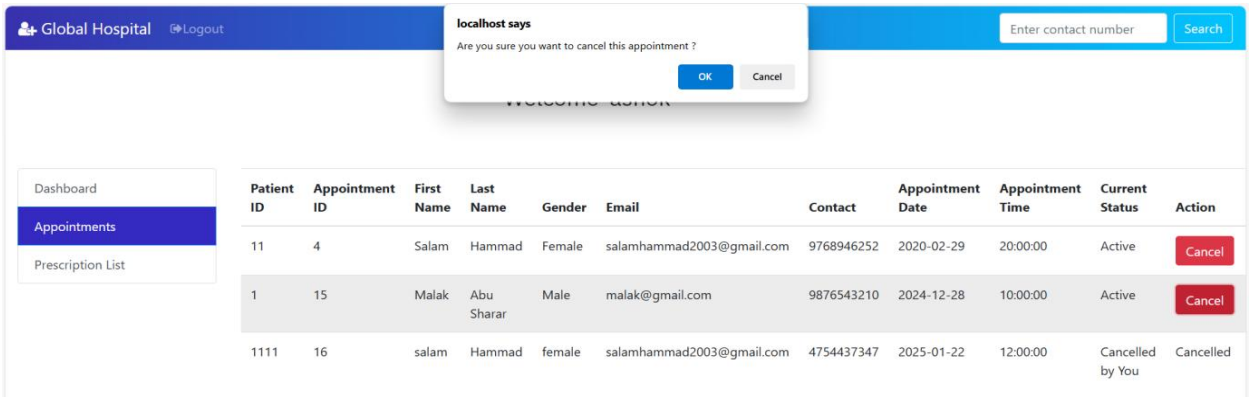
Patient information has been successfully saved



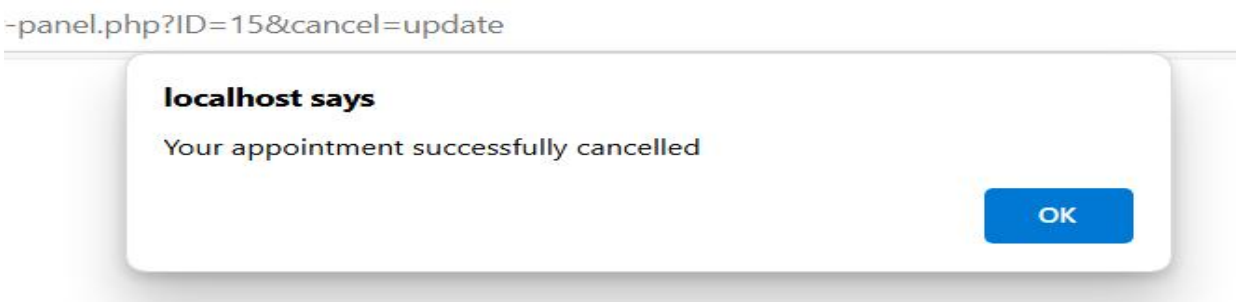
Doctor Dashboard



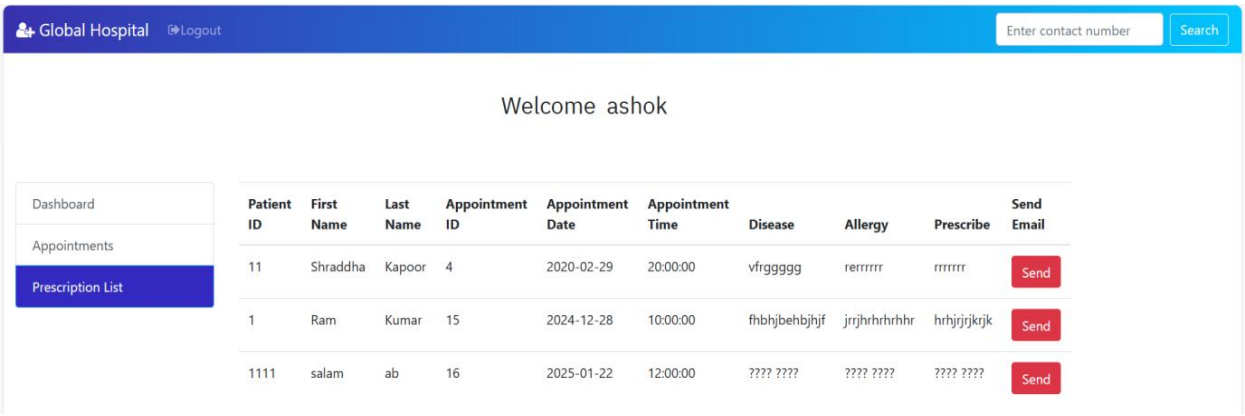
Doctor's Appointment Management Interface



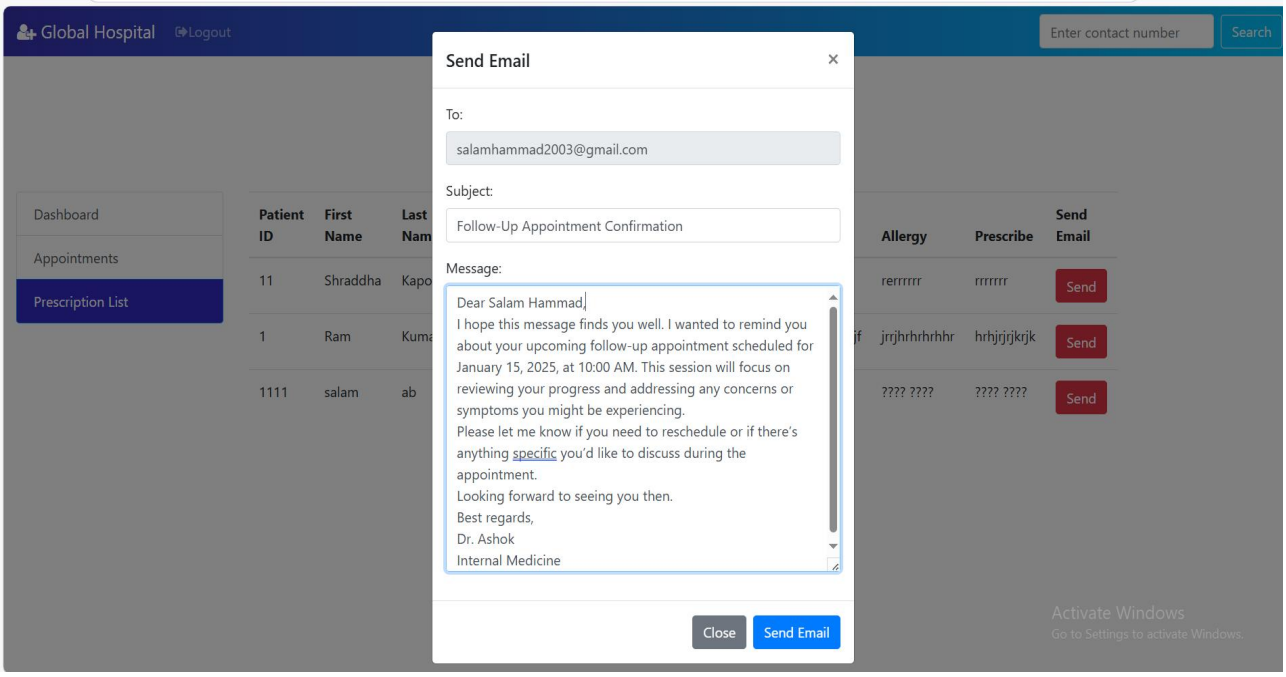
Appointment Cancellation Confirmation



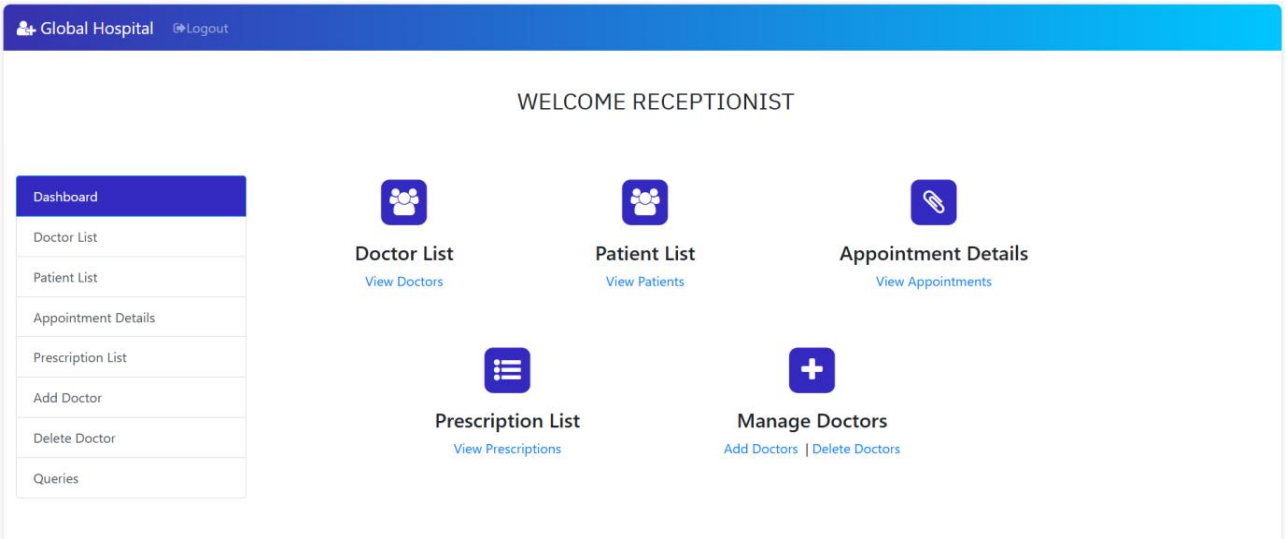
This message confirms the successful cancellation of the selected appointment, providing reassurance to the user that the action has been completed



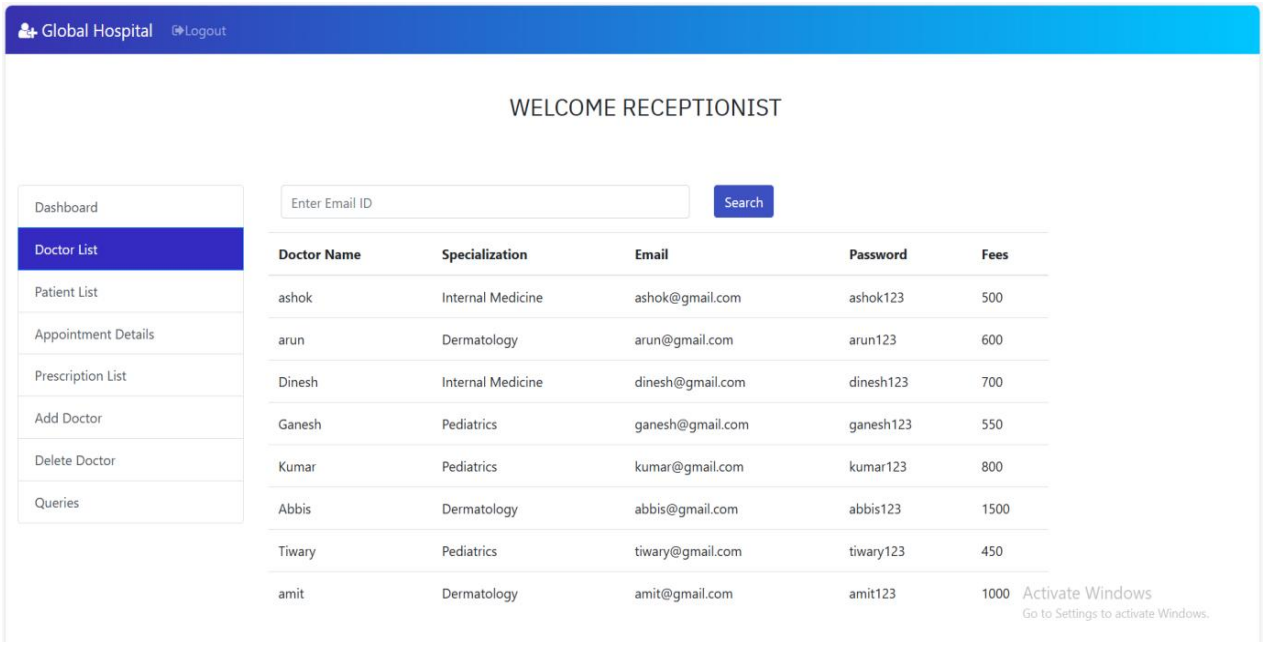
Doctor's Prescription Management Interface



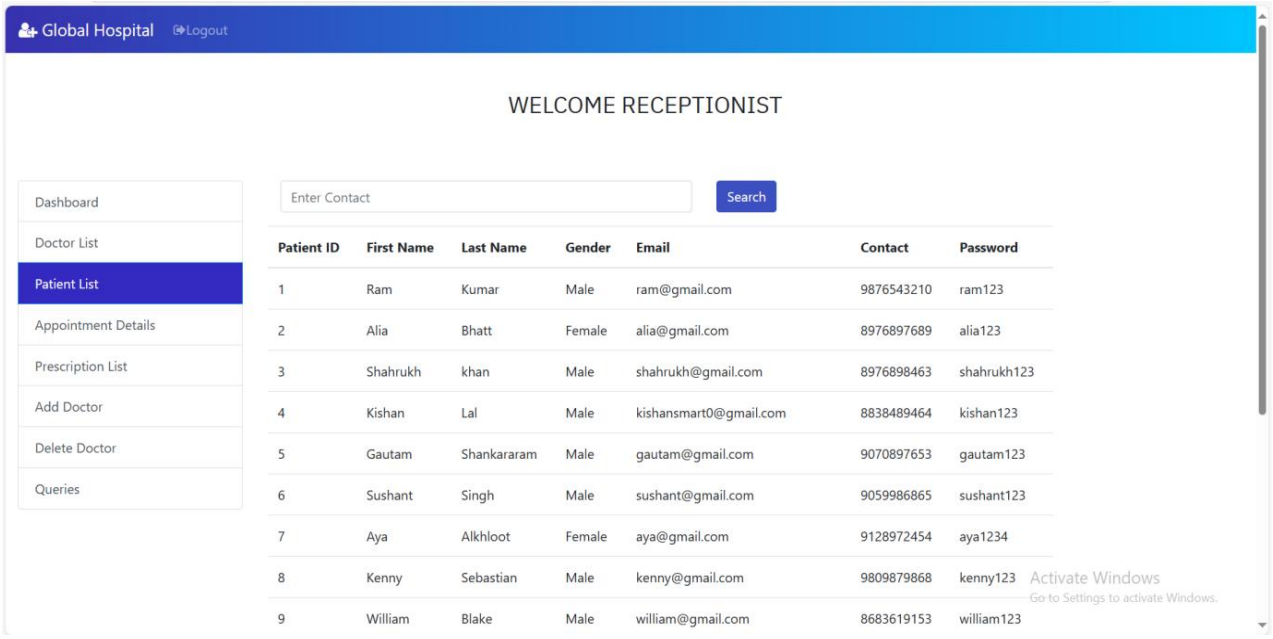
Send Email Feature



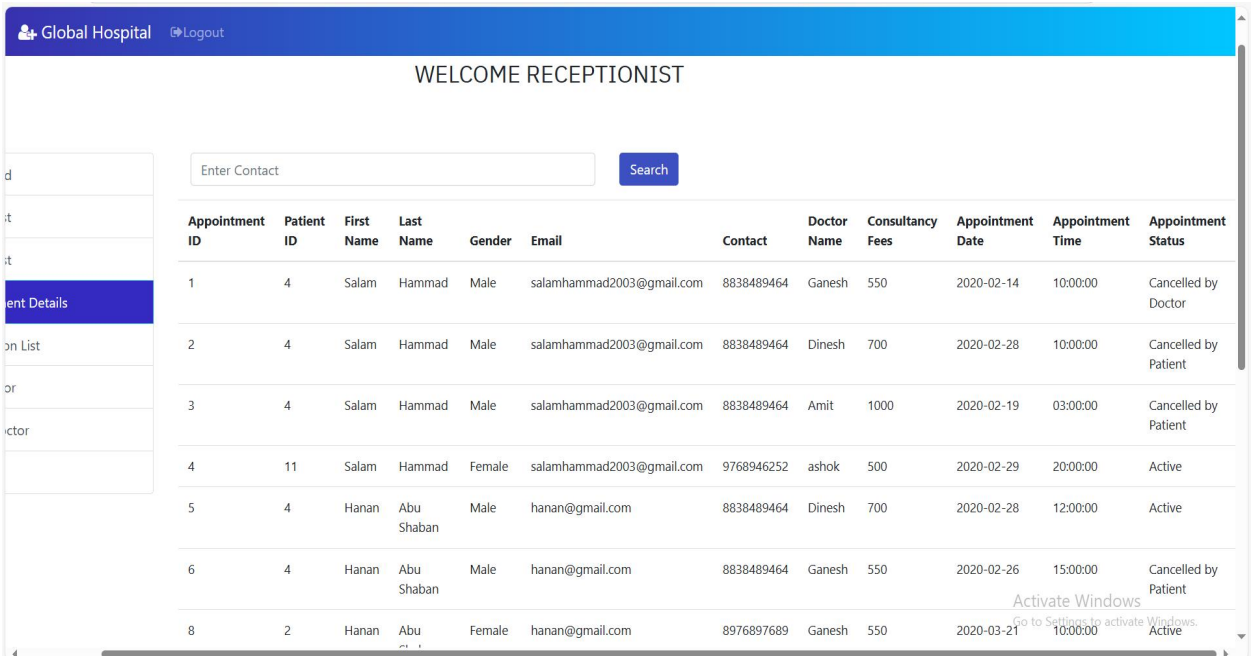
Admin Dashboard



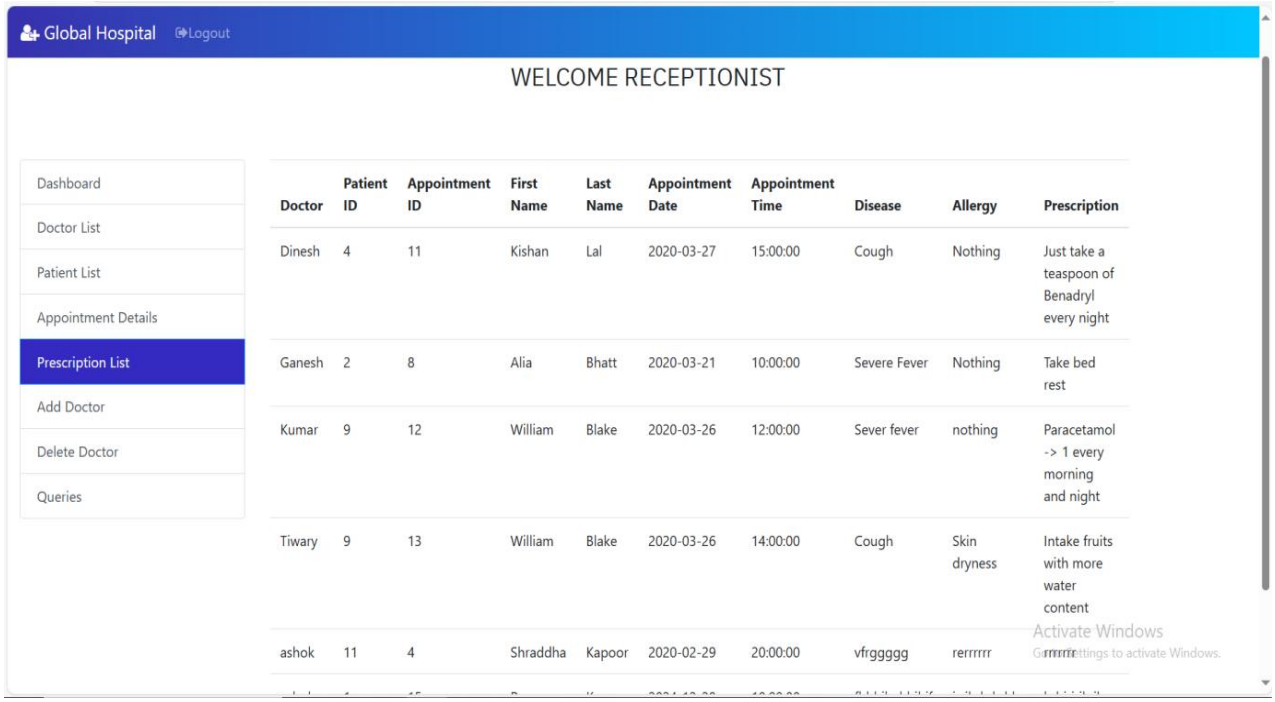
This interface provides the receptionist with a comprehensive list of all doctors and their details



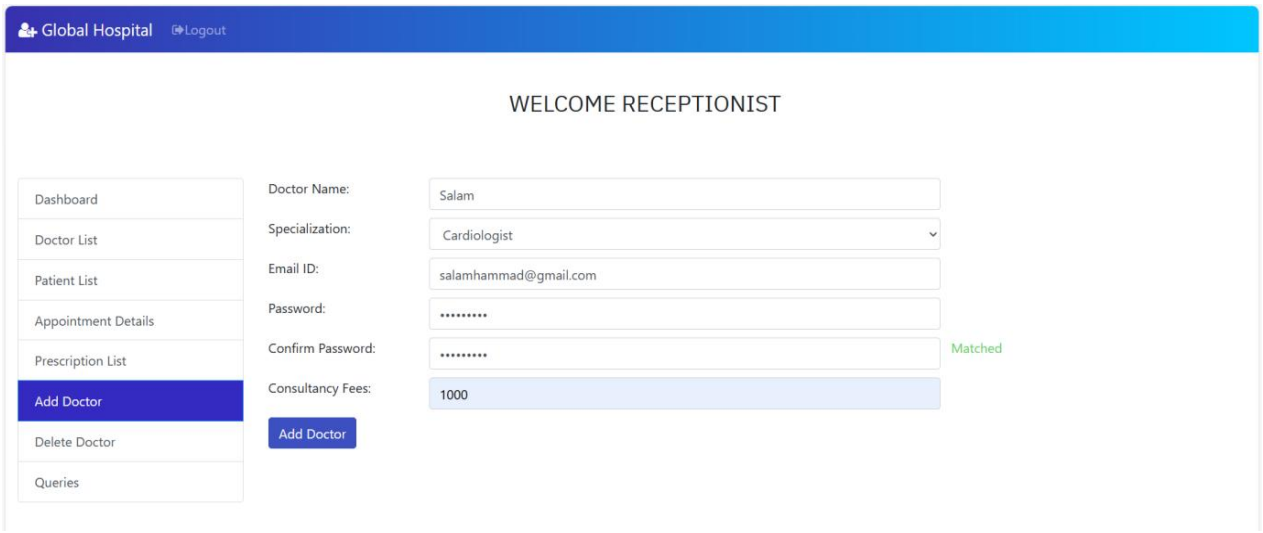
This interface provides the receptionist with a comprehensive list of all patients and their details



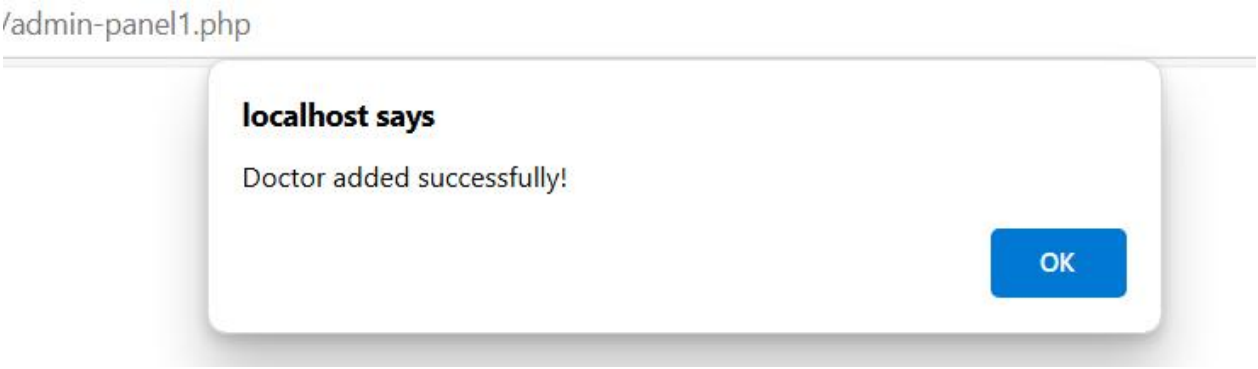
This interface provides the receptionist with a detailed overview of all patient appointments



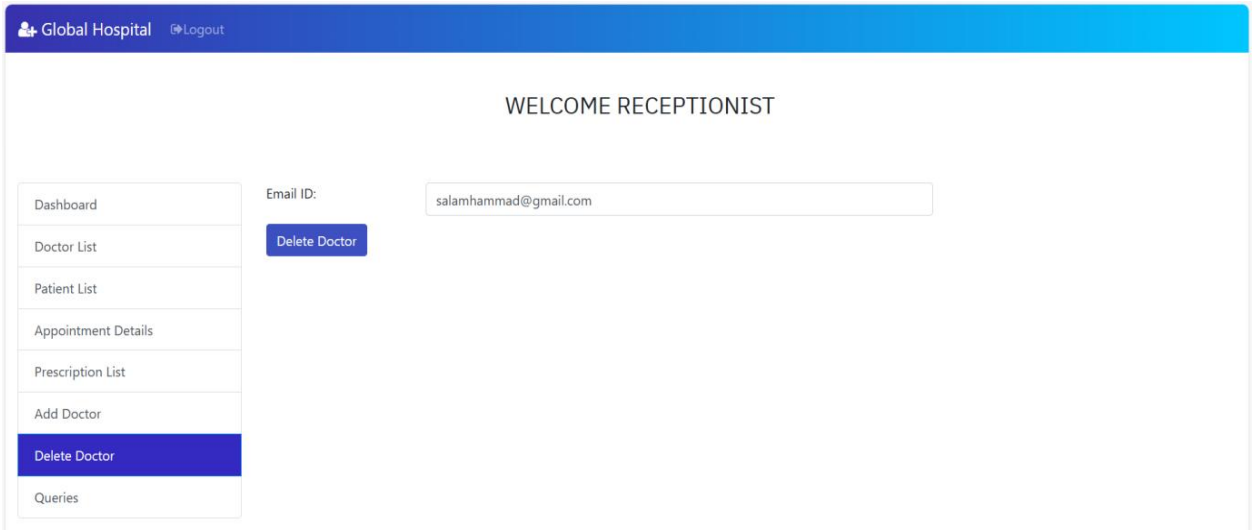
This interface provides the receptionist with a comprehensive list of patient prescriptions



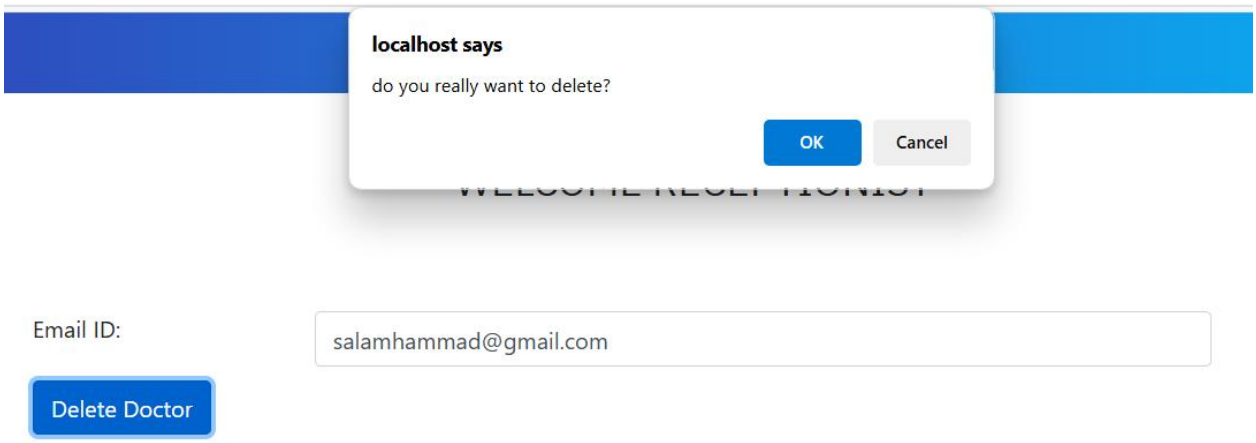
This interface allows the receptionist to add a new doctor by entering their details



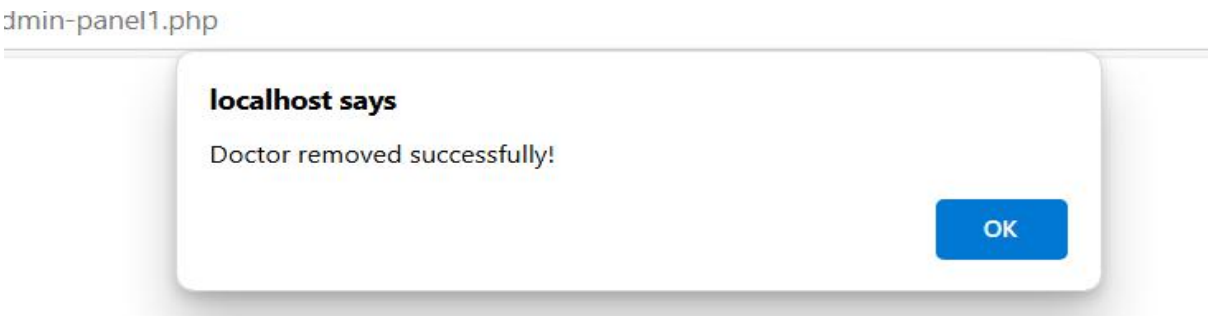
Doctor successfully added by the administrator!



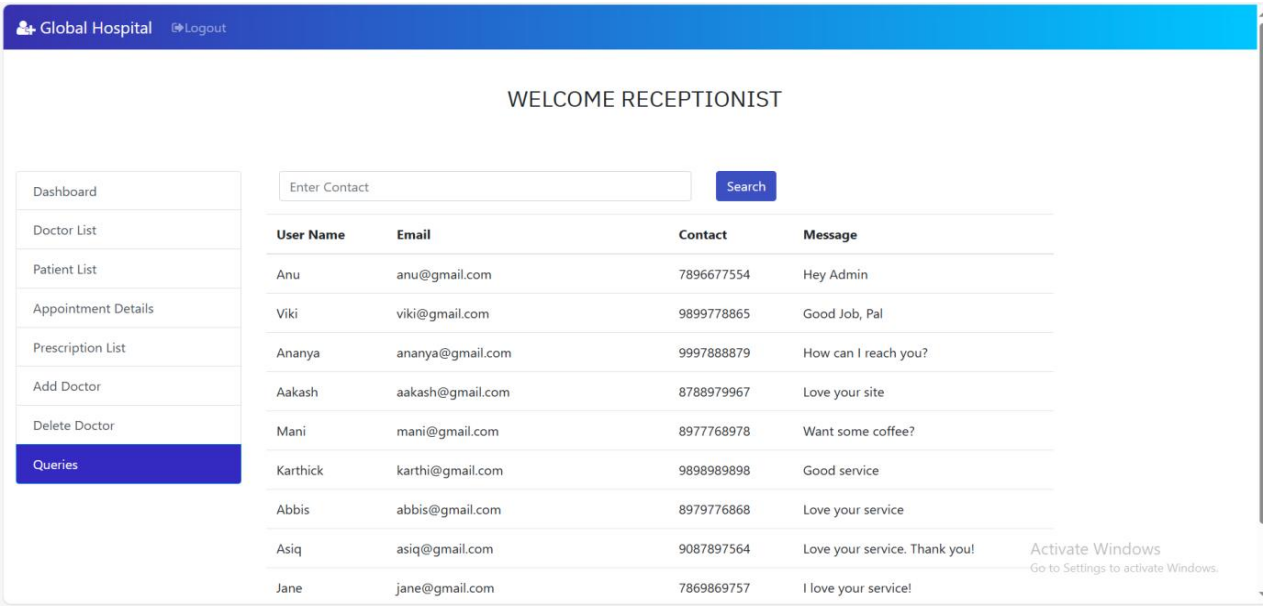
This interface allows the receptionist to delete a doctor's record by entering their email ID



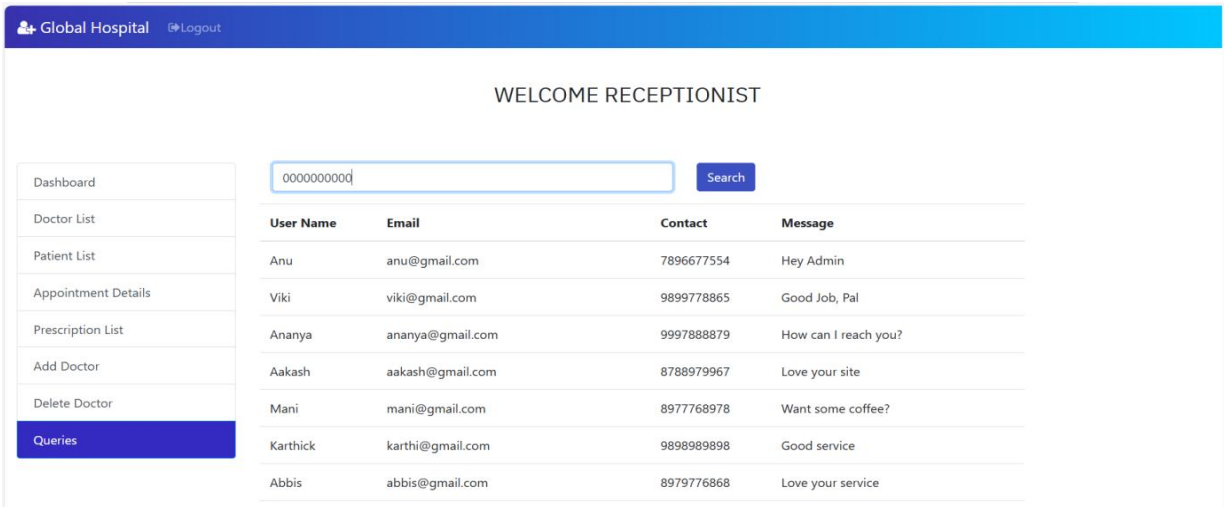
This alert prompts the receptionist to confirm the deletion of a doctor's record before proceeding

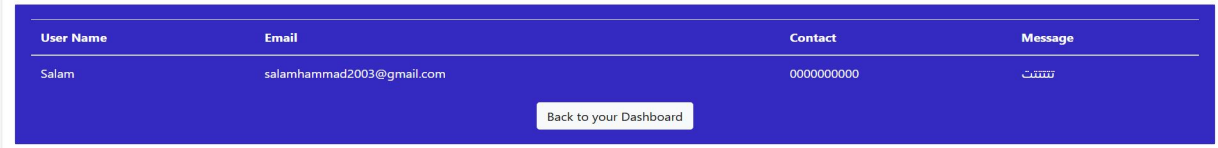


Doctor successfully removed by the administrator!

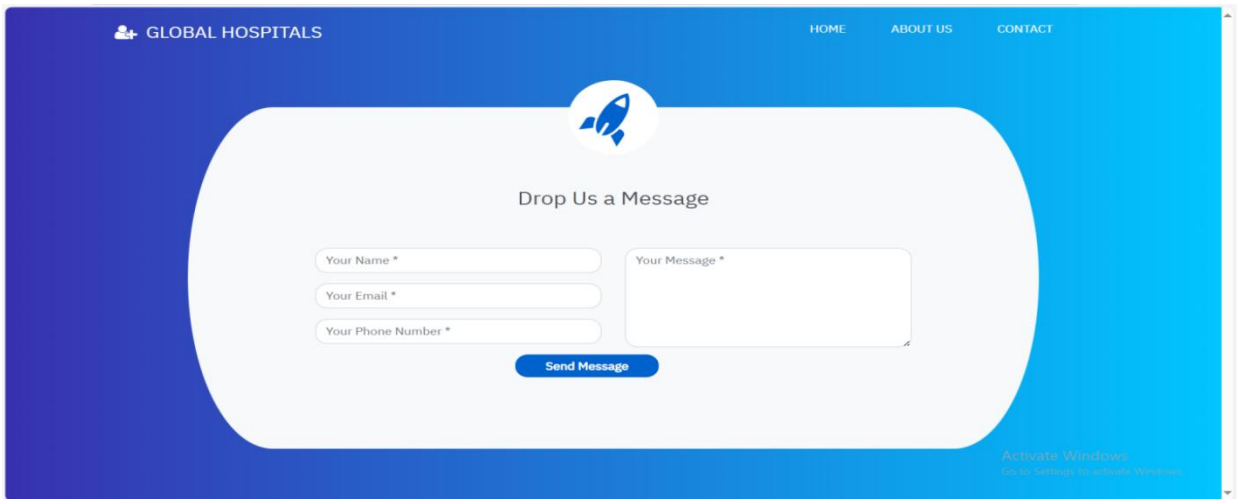


This interface displays a list of queries submitted by users, including their names, contact details, and messages, allowing the receptionist to review and respond effectively

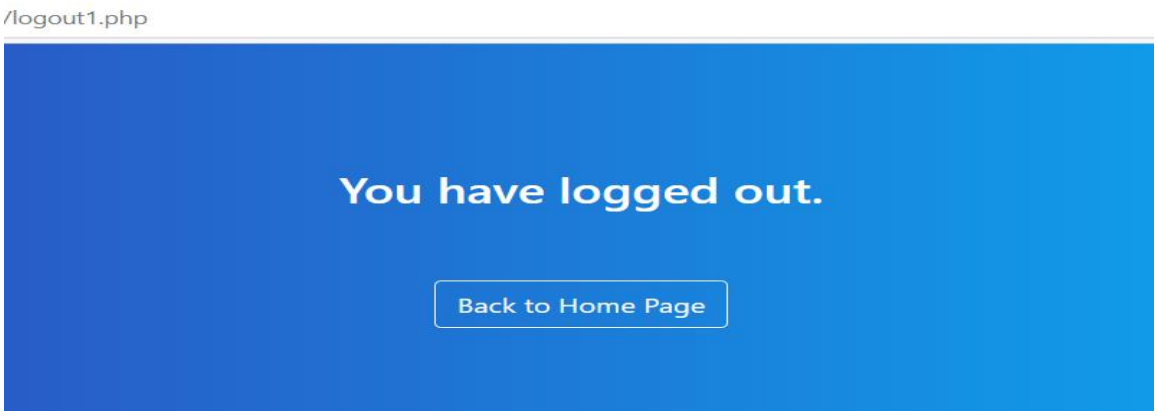




This interface displays the search result related to the user whose details were entered



Contact Page



Logout Page

5. Implementation

In this section, we will preview some of the most important code and the logic behind.

Please feel free to check out all the source code of TMS through the provided GitHub link in the references.

admin-panel.php | token.php | backend.php | indexbac.html | admin-panel1.php

token.php > TokenHandler > generateToken

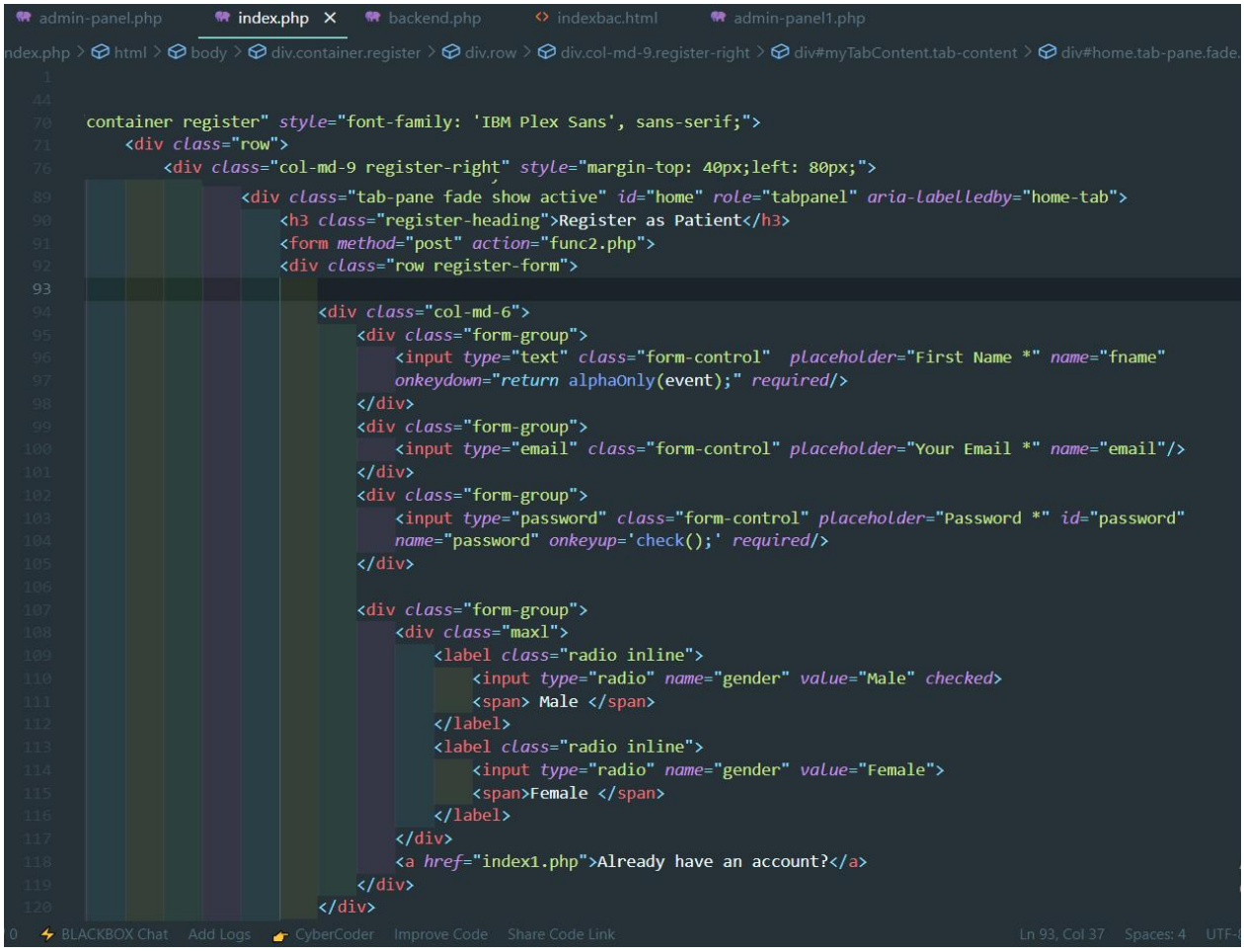
```php
}
// Generating a token
$userData = ['id' => 1, 'name' => 'Salam Hammad', 'role' => 'patient'];
$token = TokenHandler::generateToken($userData);
echo "Generated Token: " . $token . "\n";

// Verifying the token
$decodedData = TokenHandler::verifyToken($token);
if ($decodedData) {
    echo "Token is valid. Decoded Data: ";
    print_r($decodedData);
} else {
    echo "Invalid Token.";
}

// Authorizing a user
if (TokenHandler::authorizeUser($token)) {
    echo "Access granted.";
} else {
    echo "Access denied.";
}

?>
```

admin-panel.php | token.php | backend.php | indexbac.html | admin-panel1.php

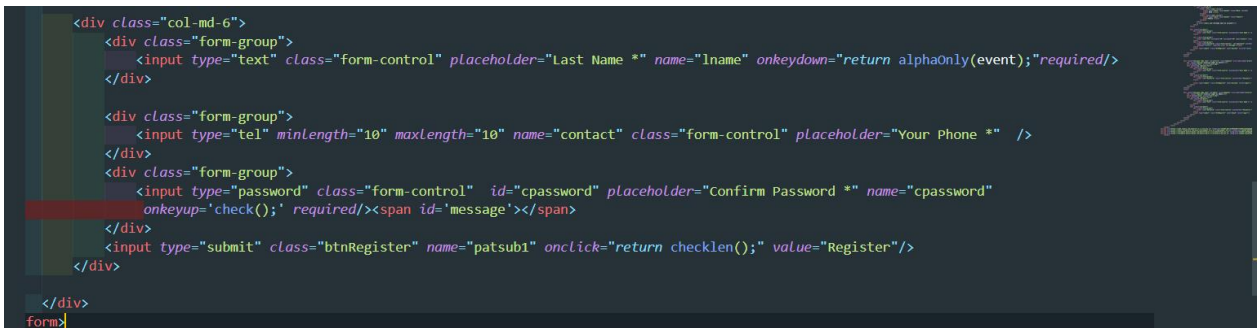
token.php > TokenHandler > generateToken

```php
class TokenHandler
    public static function verifyToken($token)
        }
    }
    /**
     * Authorize user based on token.
     * @param string $token
     * @return bool
     */
    public static function authorizeUser($token)
    {
        if (!$token) {
            // No token provided
            echo "Unauthorized access, no token.";
            return false;
        }

        // Verify and decode the token
        $decoded = self::verifyToken($token);

        if ($decoded) {
            return true;
        } else {
            echo "User is not authorized.";
            return false;
        }
    }
}
```

Token Generation, Verification & Authorization



```
admin-panel.php  index.php  backend.php  indexbac.html  admin-panel1.php
index.php > html > body > div.container.register > div.row > div.col-md-9.register-right > div#myTabContent.tab-content > div#home.tab-pane.fade

container register" style="font-family: 'IBM Plex Sans', sans-serif;">
    <div class="row">
        <div class="col-md-9 register-right" style="margin-top: 40px;left: 80px;">
            <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">
                <h3 class="register-heading">Register as Patient</h3>
                <form method="post" action="func2.php">
                <div class="row register-form">

                    <div class="col-md-6">
                        <div class="form-group">
                            <input type="text" class="form-control"  placeholder="First Name *" name="fname"
                            onkeydown="return alphaOnly(event);" required/>
                        </div>
                        <div class="form-group">
                            <input type="email" class="form-control" placeholder="Your Email *" name="email"/>
                        </div>
                        <div class="form-group">
                            <input type="password" class="form-control" placeholder="Password *" id="password"
                            name="password" onkeyup='check();' required/>
                        </div>

                        <div class="form-group">
                            <div class="maxl">
                                <label class="radio inline">
                                    <input type="radio" name="gender" value="Male" checked>
                                    <span> Male </span>
                                </label>
                                <label class="radio inline">
                                    <input type="radio" name="gender" value="Female">
                                    <span>Female </span>
                                </label>
                            </div>
                            <a href="index1.php">Already have an account?</a>
                        </div>
                    </div>

BLACKBOX Chat  Add Logs  CyberCoder  Improve Code  Share Code Link  Ln 93, Col 37  Spaces: 4  UTF-
```



```
    <div class="col-md-6">
        <div class="form-group">
            <input type="text" class="form-control" placeholder="Last Name *" name="lname" onkeydown="return alphaOnly(event);"required/>
        </div>

        <div class="form-group">
            <input type="tel" minlength="10" maxlength="10" name="contact" class="form-control" placeholder="Your Phone *"  />
        </div>
        <div class="form-group">
            <input type="password" class="form-control"  id="cpassword" placeholder="Confirm Password *" name="cpassword"
            onkeyup='check();' required/><span id='message'></span>
        </div>
        <input type="submit" class="btnRegister" name="patsub1" onclick="return checklen();" value="Register"/>
    </div>

  </div>
form>
```

```
                <div class="tab-pane fade show" id="profile" role="tabpanel" aria-labelledby="profile-tab">
                    <h3  class="register-heading">Login as Doctor</h3>
                    <form method="post" action="func1.php">
                    <div class="row register-form">
                        <div class="col-md-6">
                            <div class="form-group">
                                <input type="text" class="form-control" placeholder="User Name *" name="username3"
                                onkeydown="return alphaOnly(event);" required/>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="form-group">
                                <input type="password" class="form-control" placeholder="Password *" name="password3" required/>
                            </div>

                            <input type="submit" class="btnRegister" name="docsub1" value="Login"/>
                        </div>
                    </div>
                </form>
                </div>
```

```
<div class="tab-pane fade show" id="admin" role="tabpanel" aria-labelledby="profile-tab">
    <h3  class="register-heading">Login as Admin</h3>
    <form method="post" action="func3.php">
    <div class="row register-form">
        <div class="col-md-6">
            <div class="form-group">
                <input type="text" class="form-control" placeholder="User Name *" name="username1"
                onkeydown="return alphaOnly(event);" required/>
            </div>
        </div>
        <div class="col-md-6">
            <div class="form-group">
                <input type="password" class="form-control" placeholder="Password *" name="password2" required/>
            </div>
            <input type="submit" class="btnRegister" name="adsub" value="Login"/>
        </div>
    </div>
</form>
</div>
</div>
</div>
</div>
```

```
admin-panel.php  index.php  index1.php  backend.php  indexbac.html  admin-panel1.php
index1.php > html > body > div.container-fluid
<html lang="en">
  <body style="background: -webkit-linear-gradient(left, #3931af, #00c6ff); background-size: cover;">
    <div class="container-fluid" style="margin-top:60px;margin-bottom:60px;color:#34495E;">
      <div class="row">
        <div class="col-md-4" style="margin-top: 5%;right: 8%">
          <br>
          <h3 style="margin-top: 10%">Patient Login</h3><br>
          <form class="form-group" method="POST" action="func.php">
            <div class="row" style="margin-top: 10%">
              <div class="col-md-4"><label>Email-ID: </label></div>
              <div class="col-md-8"><input type="text" name="email" class="form-control" placeholder="enter email ID"
              required/></div><br><br>
              <div class="col-md-4" style="margin-top: 8%"><label>Password: </label></div>
              <div class="col-md-8" style="margin-top: 8%"><input type="password" class="form-control" name="password2"
              placeholder="enter password" required/></div><br><br><br>
            </div>
            <div class="row">
              <div class="col-md-4"  style="padding-left: 160px;margin-top: 10%">
                <center><input type="submit" id="inputbtn" name="patsub" value="Login" class="btn btn-primary"></center></div>
            </div>
          </form>
        </center>
        </div>
```

The login implementation for the Doctor, Administrator, and Patient in the Healthcare Management System (HMS)

admin-panel.php | backend.php | indexbac.html | admin-panel1.php

admin-panel.php > ...

```php
if (isset($_POST['app-submit'])) {
    $pid = $_SESSION['pid'];
    $username = $_SESSION['username'];
    $email = $_SESSION['email'];
    $fname = $_SESSION['fname'];
    $lname = $_SESSION['lname'];
    $gender = $_SESSION['gender'];
    $contact = $_SESSION['contact'];
    $doctor = $_POST['doctor'];
    $email = $_SESSION['email'];
    $docFees = $_POST['docFees'];

    $appdate = $_POST['appdate'];
    $apptime = $_POST['apptime'];
    $cur_date = date("Y-m-d");
    date_default_timezone_set('Asia/Kolkata');
    $cur_time = date("H:i:s");
    $apptime1 = strtotime($apptime);
    $appdate1 = strtotime($appdate);
```

admin-panel.php | backend.php | indexbac.html | admin-panel1.php

admin-panel.php > ...

```php
if (date("Y-m-d", $appdate1) >= $cur_date) {
  if ((date("Y-m-d", $appdate1) == $cur_date and date("H:i:s", $apptime1) > $cur_time) or date("Y-m-d", $appdate1) > $cur_date) {
    $check_query = mysqli_query($con, "select apptime from appointmenttb where doctor='$doctor'
    and appdate='$appdate' and apptime='$apptime'");

    if (mysqli_num_rows($check_query) == 0) {
      $query = mysqli_query($con, "insert into appointmenttb(pid,fname,lname,gender,email,contact,doctor,
      docFees,appdate,apptime,userStatus,doctorStatus)
      values($pid,'$fname','$lname','$gender','$email','$contact','$doctor','$docFees','$appdate','$apptime','1','1')");

      if ($query) {
        echo "<script>alert('Your appointment successfully booked');</script>";
      } else {
        echo "<script>alert('Unable to process your request. Please try again!');</script>";
      }
    } else {
      echo "<script>alert('We are sorry to inform that the doctor is not available in this time or date.
      Please choose different time or date!');</script>";
    }
  } else {
    echo "<script>alert('Select a time or date in the future!');</script>";
  }
} else {
  echo "<script>alert('Select a time or date in the future!');</script>";
}
```

Booking Appointments allows patients to efficiently schedule doctor consultations.

admin-panel.php

admin-panel.php > ...

```php
if (isset($_GET['cancel'])) {
  $query = mysqli_query($con, "update appointmenttb set userStatus='0' where ID = '" . $_GET['ID'] . "'");
  if ($query) {
    echo "<script>alert('Your appointment successfully cancelled');</script>";
  }
}
```

Cancel Appointments .

admin-panel.php

```php
function generate_bill()
{
  $con = mysqli_connect("localhost", "root", "", "myhmsdb");
  $pid = $_SESSION['pid'];
  $output = '';
  $query = mysqli_query($con, "select p.pid,p.ID,p.fname,p.lname,p.doctor,p.appdate,p.apptime,p.disease,p.allergy,
  p.prescription,a.docFees from prestb p inner join appointmenttb a on p.ID=a.ID and p.pid = '$pid' and p.ID = '" . $_GET['ID'] . "'");
  while ($row = mysqli_fetch_array($query)) {
    $output .= '
    <label> Patient ID : </label>' . $row["pid"] . '<br/><br/>
    <label> Appointment ID : </label>' . $row["ID"] . '<br/><br/>
    <label> Patient Name : </label>' . $row["fname"] . ' ' . $row["lname"] . '<br/><br/>
    <label> Doctor Name : </label>' . $row["doctor"] . '<br/><br/>
    <label> Appointment Date : </label>' . $row["appdate"] . '<br/><br/>
    <label> Appointment Time : </label>' . $row["apptime"] . '<br/><br/>
    <label> Disease : </label>' . $row["disease"] . '<br/><br/>
    <label> Allergies : </label>' . $row["allergy"] . '<br/><br/>
    <label> Prescription : </label>' . $row["prescription"] . '<br/><br/>
    <label> Fees Paid : </label>' . $row["docFees"] . '<br/>

    ';
  }

  return $output;
}
```

```php
if (isset($_GET["generate_bill"])) {
  require_once("TCPDF/tcpdf.php");
  $obj_pdf = new TCPDF('P', PDF_UNIT, PDF_PAGE_FORMAT, true, 'UTF-8', false);
  $obj_pdf->SetCreator(PDF_CREATOR);
  $obj_pdf->SetTitle("Generate Bill");
  $obj_pdf->SetHeaderData('', '', PDF_HEADER_TITLE, PDF_HEADER_STRING);
  $obj_pdf->SetHeaderFont(array(PDF_FONT_NAME_MAIN, '', PDF_FONT_SIZE_MAIN));
  $obj_pdf->SetFooterFont(array(PDF_FONT_NAME_MAIN, '', PDF_FONT_SIZE_MAIN));
  $obj_pdf->SetDefaultMonospacedFont('helvetica');
  $obj_pdf->SetFooterMargin(PDF_MARGIN_FOOTER);
  $obj_pdf->SetMargins(PDF_MARGIN_LEFT, '5', PDF_MARGIN_RIGHT);
  $obj_pdf->SetPrintHeader(false);
  $obj_pdf->SetPrintFooter(false);
  $obj_pdf->SetAutoPageBreak(TRUE, 10);
  $obj_pdf->SetFont('helvetica', '', 12);
  $obj_pdf->AddPage();

  $content = '';
  $content .= '<br/>
      <h2 align ="center"> Global Hospitals</h2></br>
      <h3 align ="center"> Bill</h3>';
  $content .= generate_bill();
  $obj_pdf->writeHTML($content);
  ob_end_clean();
  $obj_pdf->Output("bill.pdf", 'I');
}
```

Generate Bill.

admin-panel.php | indexbac.html | appsearch.php | doctor-panel.php | doctorsearch.php | newfunc.php

```php
<?php
$con=mysqli_connect("localhost","root","","myhmsdb");
if(isset($_POST['update_data']))
{
 $contact=$_POST['contact'];
 $status=$_POST['status'];
 $query="update appointmenttb set payment='$status' where contact='$contact';";
 $result=mysqli_query($con,$query);
 if($result)
  header("Location:updated.php");
}
```

Update appointment payment status by contact.

```
<script>
  document.getElementById('spec').onchange = function foo() {
    let spec = this.value;
    console.log(spec)
    let docs = [...document.getElementById('doctor').options];

    docs.forEach((el, ind, arr) => {
      arr[ind].setAttribute("style", "");
      if (el.getAttribute("data-spec") != spec) {
        arr[ind].setAttribute("style", "display: none");
      }
    });
  };
</script>
```

Display Doctors by Specialization.

```
<script>
  document.getElementById('doctor').onchange = function updateFees(e) {
    var selection = document.querySelector(`[value=${this.value}]`).getAttribute('data-value');
    document.getElementById('docFees').value = selection;
  };
</script>
```

Update Fees on Doctor Selection.

```
<div class="tab-pane fade" id="app-hist" role="tabpanel" aria-labelledby="list-pat-list">
  <table class="table table-hover">
    <thead>
      <tr>
        <th scope="col">Doctor Name</th>
        <th scope="col">Consultancy Fees</th>
        <th scope="col">Appointment Date</th>
        <th scope="col">Appointment Time</th>
        <th scope="col">Current Status</th>
        <th scope="col">Action</th>
      </tr>
    </thead>
    <tbody>
      <?php

      $con = mysqli_connect("localhost", "root", "", "myhmsdb");
      global $con;

      $query = "select ID,doctor,docFees,appdate,apptime,userStatus,doctorStatus from appointmenttb
      where fname ='$fname' and lname='$lname';";
      $result = mysqli_query($con, $query);
      while ($row = mysqli_fetch_array($result)) {
      ?>
        <tr>
          <td><?php echo $row['doctor']; ?></td>
          <td><?php echo $row['docFees']; ?></td>
          <td><?php echo $row['appdate']; ?></td>
          <td><?php echo $row['apptime']; ?></td>
          <td>
            <?php if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 1)) {
              echo "Active";
```

```
                    }
                    if (($row['userStatus'] == 0) && ($row['doctorStatus'] == 1)) {
                      echo "Cancelled by You";
                    }
                    if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 0)) {
                      echo "Cancelled by Doctor";
                    }
                    ?></td>
                  <td>
                    <?php if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 1)) { ?>
                      <a href="admin-panel.php?ID=<?php echo $row['ID'] ?>&cancel=update"
                        onClick="return confirm('Are you sure you want to cancel this appointment ?')"
                        title="Cancel Appointment" tooltip-placement="top" tooltip="Remove"><button
                        class="btn btn-danger">Cancel</button></a>
                    <?php } else {
                      echo "Cancelled";
                    } ?>
                  </td>
                </tr>
              <?php } ?>
            </tbody>
          </table>
          <br>
        </div>
```

View Appointment History.

```
<div class="tab-pane fade" id="list-pres" role="tabpanel" aria-labelledby="list-pres-list">
  <table class="table table-hover">
    <thead>
      <tr>
        <th scope="col">Doctor Name</th>
        <th scope="col">Appointment ID</th>
        <th scope="col">Appointment Date</th>
        <th scope="col">Appointment Time</th>
        <th scope="col">Diseases</th>
        <th scope="col">Allergies</th>
        <th scope="col">Prescriptions</th>
        <th scope="col">Bill Payment</th>
      </tr>
    </thead>
    <tbody>
      <?php
      $con = mysqli_connect("localhost", "root", "", "myhmsdb");
      global $con;
      $query = "select doctor,ID,appdate,apptime,disease,allergy,prescription from prestb where pid='$pid';";
      $result = mysqli_query($con, $query);
      if (!$result) {
        echo mysqli_error($con);
      }
```

```
      while ($row = mysqli_fetch_array($result)) {
      ?>
        <tr>
          <td><?php echo $row['doctor']; ?></td>
          <td><?php echo $row['ID']; ?></td>
          <td><?php echo $row['appdate']; ?></td>
          <td><?php echo $row['apptime']; ?></td>
          <td><?php echo $row['disease']; ?></td>
          <td><?php echo $row['allergy']; ?></td>
          <td><?php echo $row['prescription']; ?></td>
          <td>
            <form method="get">
              <a href="admin-panel.php?ID=<?php echo $row['ID'] ?>">
                <input type="hidden" name="ID" value="<?php echo $row['ID'] ?>" />
                <input type="submit" onclick="alert('Bill Paid Successfully');" name="generate_bill"
                class="btn btn-success" value="Pay Bill" />
              </a>
          </td>
          </form>
        </tr>
      <?php }
      ?>
    </tbody>
  </table>
  <br>
</div>
```

View Prescriptions.

```php
<div class="tab-pane fade" id="list-messages" role="tabpanel" aria-labelledby="list-messages-list">...</div>
<div class="tab-pane fade" id="list-settings" role="tabpanel" aria-labelledby="list-settings-list">
  <form class="form-group" method="post" action="func.php">
    <label>Doctors name: </label>
    <input type="text" name="name" placeholder="Enter doctors name" class="form-control">
    <br>
    <input type="submit" name="doc_sub" value="Add Doctor" class="btn btn-primary">
  </form>
</div>
```

admin-panel.php | indexbac.html | admin-panel1.php

admin-panel1.php > ...

```php
$con = mysqli_connect("localhost", "root", "", "myhmsdb");
include('newfunc.php');
if (isset($_POST['docsub'])) {
  $doctor = $_POST['doctor'];
  $dpassword = $_POST['dpassword'];
  $demail = $_POST['demail'];
  $spec = $_POST['special'];
  $docFees = $_POST['docFees'];
  $query = "insert into doctb(username,password,email,spec,docFees)values('$doctor','$dpassword','$demail','$spec','$docFees')";
  $result = mysqli_query($con, $query);
  if ($result) {
    echo "<script>alert('Doctor added successfully!');</script>";
  }
}
```

Add Doctor.

```php
if (isset($_POST['docsub1'])) {
  $demail = $_POST['demail'];
  $query = "delete from doctb where email='$demail';";
  $result = mysqli_query($con, $query);
  if ($result) {
    echo "<script>alert('Doctor removed successfully!');</script>";
  } else {
    echo "<script>alert('Unable to delete!');</script>";
  }
}
?>
```

Delete Doctor.

```php
<?php
$con = mysqli_connect("localhost", "root", "", "myhmsdb");
global $con;
$query = "select * from doctb";
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
  $username = $row['username'];
  $spec = $row['spec'];
  $email = $row['email'];
  $password = $row['password'];
  $docFees = $row['docFees'];

  echo "<tr>
        <td>$username</td>
        <td>$spec</td>
        <td>$email</td>
        <td>$password</td>
        <td>$docFees</td>
      </tr>";
}
?>
</tbody>
```

View Doctor List.

```php
$con = mysqli_connect("localhost", "root", "", "myhmsdb");
global $con;
$query = "select * from patreg";
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
  $pid = $row['pid'];
  $fname = $row['fname'];
  $lname = $row['lname'];
  $gender = $row['gender'];
  $email = $row['email'];
  $contact = $row['contact'];
  $password = $row['password'];

  echo "<tr>
        <td>$pid</td>
        <td>$fname</td>
        <td>$lname</td>
        <td>$gender</td>
        <td>$email</td>
        <td>$contact</td>
        <td>$password</td>
      </tr>";
}
?>
```

View Patient List.

```php
$query = "select * from appointmenttb;";
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
?>
  <tr>
    <td><?php echo $row['ID']; ?></td>
    <td><?php echo $row['pid']; ?></td>
    <td><?php echo $row['fname']; ?></td>
    <td><?php echo $row['lname']; ?></td>
    <td><?php echo $row['gender']; ?></td>
    <td><?php echo $row['email']; ?></td>
    <td><?php echo $row['contact']; ?></td>
    <td><?php echo $row['doctor']; ?></td>
    <td><?php echo $row['docFees']; ?></td>
    <td><?php echo $row['appdate']; ?></td>
    <td><?php echo $row['apptime']; ?></td>
    <td>
      <?php if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 1)) {
        echo "Active";
      }
      if (($row['userStatus'] == 0) && ($row['doctorStatus'] == 1)) {
        echo "Cancelled by Patient";
      }

      if (($row['userStatus'] == 1) && ($row['doctorStatus'] == 0)) {
        echo "Cancelled by Doctor";
      }
      ?></td>
  </tr>
<?php } ?>
```

View Appointment Details.

```php
<?php
$con = mysqli_connect("localhost", "root", "", "myhmsdb");
global $con;
$query = "select * from prestb";
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
  $doctor = $row['doctor'];
  $pid = $row['pid'];
  $ID = $row['ID'];
  $fname = $row['fname'];
  $lname = $row['lname'];
  $appdate = $row['appdate'];
  $apptime = $row['apptime'];
  $disease = $row['disease'];
  $allergy = $row['allergy'];
  $pres = $row['prescription'];

  echo "<tr>
    <td>$doctor</td>
    <td>$pid</td>
    <td>$ID</td>
    <td>$fname</td>
    <td>$lname</td>
    <td>$appdate</td>
    <td>$apptime</td>
    <td>$disease</td>
    <td>$allergy</td>
    <td>$pres</td>
  </tr>";
}
?>
```

View Prescription List.

admin-panel.php | indexbac.html | appsearch.php

appsearch.php > html > body

```php
<html>
<body>
<?php
include("newfunc.php");
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 {
```

Search Appointments by Contact.

```php
<html lang="en">
<body>
    <div class="container">
        <form id="hospitalForm" action="backend.php" method="POST" enctype="multipart/form-data">
            <input type="text" id="username" name="username" placeholder="Enter your name" required>
            <label for="age">Age:</label>
            <input type="number" id="age" name="age" placeholder="Enter your age" required>
            <label for="hospitalData">Patient data:</label>
            <textarea id="hospitalData" name="hospitalData" rows="4" placeholder="Enter case details"></textarea>
            <label for="files">Upload medical files:</label>
            <input type="file" id="files" name="files[]" multiple>
            <label for="videos">Upload videos:</label>
            <input type="file" id="videos" name="videos[]" accept="video/*" multiple>
            <button type="submit">Send Data</button>
        </form>
    </div>
```

```php
    <script>
        const form = document.getElementById('hospitalForm');
        form.addEventListener('submit', async (event) => {
            event.preventDefault();
            const formData = new FormData(form);
            try {
                const response = await fetch('backend.php', {
                    method: 'POST',
                    body: formData,
                });
                if (response.ok) {
                    alert('Data has been successfully sent!');
                } else {
                    alert('There was an error sending the data. Please try again.');
                }
            } catch (error) {
                alert(`Error: ${error.message}`);
            }
        });
    </script>
</body>
</html>
```

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = $_POST['username'] ?? null;
    $usernumber = $_POST['usernumber'] ?? null;
    $age = $_POST['age'] ?? null;
    $hospitalData = $_POST['hospitalData'] ?? null;

    $files_dir = "uploads/files/";
    $videos_dir = "uploads/videos/";

    if (!is_dir($files_dir)) mkdir($files_dir, 0777, true);
    if (!is_dir($videos_dir)) mkdir($videos_dir, 0777, true);

    echo "<h3>Data Received:</h3>";
    echo "User Name: $username<br>";
    echo "User Number: $usernumber<br>";
    echo "Age: $age<br>";
    echo "Hospital Data: $hospitalData<br><hr>";

    if (isset($_FILES['files'])) {
        echo "<h3>Uploaded files:</h3>";
        foreach ($_FILES['files']['tmp_name'] as $key => $tmp_name) {
            $file_name = basename($_FILES['files']['name'][$key]);
            $file_path = $files_dir . time() . "_" . $file_name;

            if (move_uploaded_file($tmp_name, $file_path)) {
                echo "<a href='$file_path' target='_blank'>📄 $file_name</a><br>";
            } else {
                echo "Failed to upload file: $file_name<br>";
            }
        }
    }
```

Patient Data Entry and Receiving user data.

admin-panel.php indexbac.html appsearch.php doctor-panel.php doctorsearch.php send_email.php

send_email.php > ...

```php
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require 'phpmailer/PHPMailer.php';
require 'phpmailer/Exception.php';
require 'phpmailer/SMTP.php';
```

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $to = $_POST['email'];
    $subject = $_POST['subject'];
    $message = $_POST['message'];
    $mail = new PHPMailer(true);
    try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;
        $mail->Username = '235380@ppu.edu.ps';
        $mail->Password = '408365070';
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
        $mail->Port = 587;
        $mail->setFrom('235380@ppu.edu.ps', 'Hospital');
        $mail->addAddress($to);
        $mail->isHTML(true);
        $mail->Subject = $subject;
        $mail->Body = $message;
        $mail->send();
```

Sending an email using PHPMailer with SweetAlert2 notifications for success or failure.

admin-panel.php indexbac.html appsearch.php doctor-panel.php

doctor-panel.php > html > body > div#emailModal.modal.fade > div.modal-dialog > div.modal-content > form

```html
<html lang="en">
  <body style="padding-top:50px;">
   <div class="container-fluid" style="margin-top:50px;">
   </div>
   <!-- Modal لإرسال البريد -->
   <div class="modal fade" id="emailModal" tabindex="-1" role="dialog" aria-labelledby="emailModalLabel" aria-hidden="true">
       <div class="modal-dialog" role="document">
           <div class="modal-content">
           <form method="POST" action="send_email.php">
    <div class="modal-header">
       <h5 class="modal-title" id="emailModalLabel">Send Email</h5>
       <button type="button" class="close" data-dismiss="modal" aria-label="Close">
           <span aria-hidden="true">&times;</span>
       </button>
    </div>
    <div class="modal-body">
       <div class="form-group">
           <label for="email">To:</label>
           <input type="email" class="form-control" id="email" name="email" readonly>
       </div>
       <div class="form-group">
           <label for="subject">Subject:</label>
           <input type="text" class="form-control" id="subject" name="subject" required>
       </div>
       <div class="form-group">
           <label for="message">Message:</label>
           <textarea class="form-control" id="message" name="message" rows="5" required></textarea>
       </div>
    </div>
    <div class="modal-footer">
       <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
       <button type="submit" class="btn btn-primary">Send Email</button>
    </div>
</form>
```

Emailing enables doctors to share details with patients.

admin-panel.php indexbac.html appsearch.php doctor-panel.php doctorsearch.php

doctorsearch.php > html > body

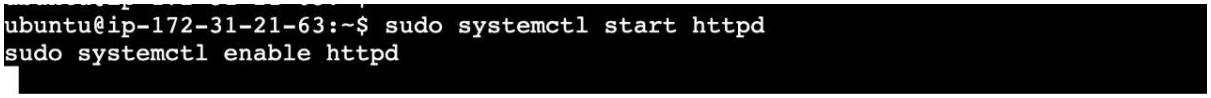
```
<html>

<body>
  <?php
  include("newfunc.php");
  if (isset($_POST['doctor_search_submit'])) {
    $contact = $_POST['doctor_contact'];
    $query = "select * from doctb where email= '$contact'";
    $result = mysqli_query($con, $query);
    $row = mysqli_fetch_array($result);
    if ($row['username'] == "" & $row['password'] == "" & $row['email'] == "" & $row['docFees'] == "") {
      echo "<script> alert('No entries found!');
          window.location.href = 'admin-panel1.php#list-doc';</script>";
    } else {
      echo "<div class='container-fluid' style='margin-top:50px;'>
  <div class ='card'>
  <div class='card-body' style='background-color:#342ac1;color:#ffffff;'>
<table class='table table-hover'>
  <thead>
    <tr>
      <th scope='col'>Username</th>
      <th scope='col'>Password</th>
      <th scope='col'>Email</th>
      <th scope='col'>Consultancy Fees</th>
    </tr>
  </thead>
  <tbody>";
```

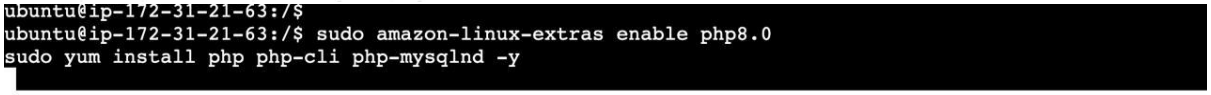
Search for doctors by email to view details or check availability.

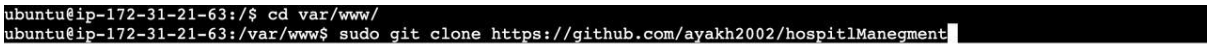
The same logic used in the doctor management component is also applied to the patient management and admin management features, ensuring a consistent and streamlined approach throughout the system. This includes handling user data, secure login, and role-specific functionalities.

For further insights on the code, you can freely check the GitHub repository provided in the references. It includes the complete file structure, detailed explanations, and line-by-line comments to clarify the implementation.

These steps outline the initial setup for deploying the application on AWS and integrating it with GitHub for version control and code management.

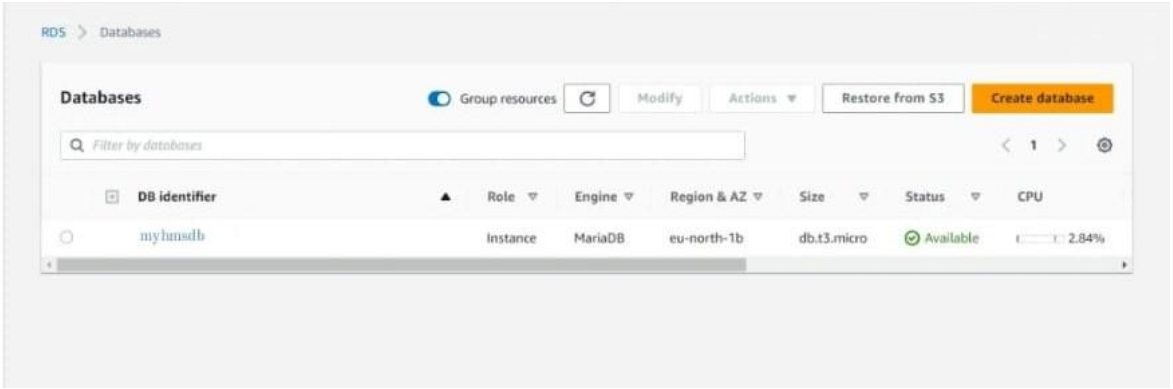






6. Data Model

he data model of our system is implemented using MySQL, which is hosted on Amazon RDS (Relational Database Service).



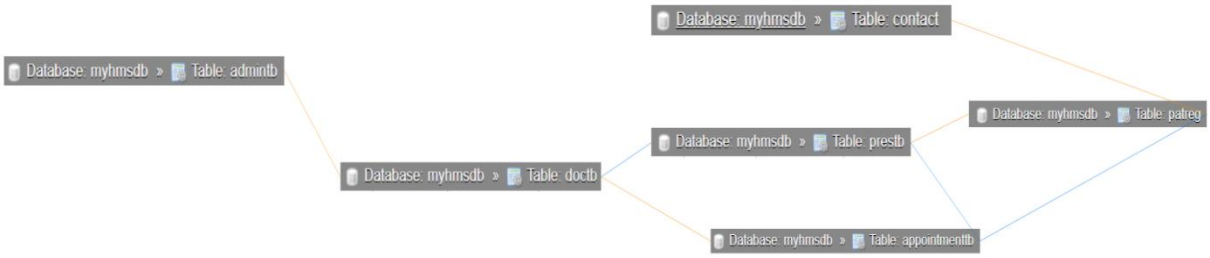
The Database Instance in RDS.

In RDS, the MySQL database is stored and its various entities and their related information are managed. This includes data related to doctors, patients, appointments, and other system-specific entities. The data model consists of tables, as illustrated in the previous figures, that represent different entities, and the relationships between these entities define the structure and organization of the data.

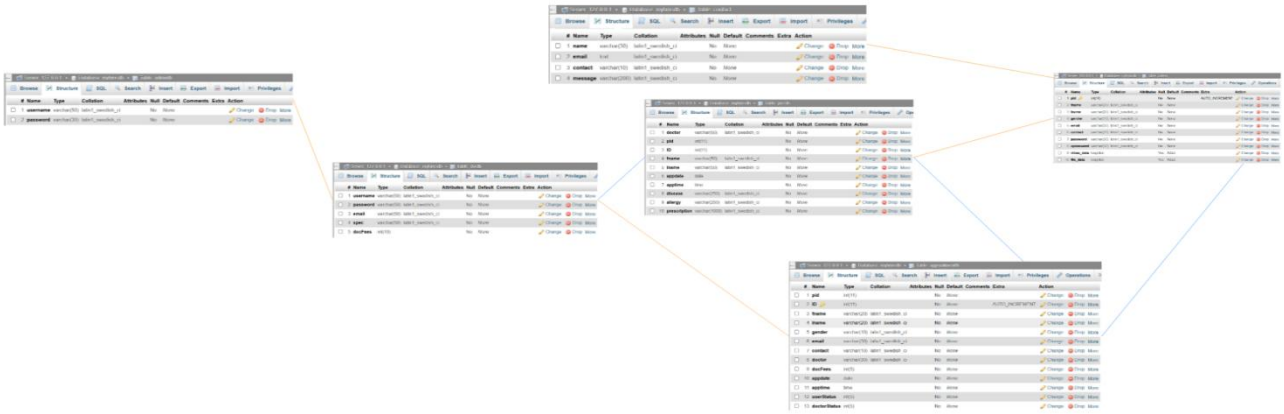
As shown in the previous figures, the data itself is stored within the MySQL database instance provided by Amazon RDS, a fully managed database service that simplifies the deployment, management, and scaling of databases in the cloud. It offers features such as automated backups, high availability, and performance monitoring.

By utilizing MySQL with Amazon RDS, the Hospital Management System (HMS) benefits from a reliable and scalable database solution that ensures data integrity, availability, and optimal performance. The data model is designed to meet the requirements of HMS, enabling efficient storage, retrieval, and manipulation of data to support the functionality and operations of the hospital management system.

The Database Schema:



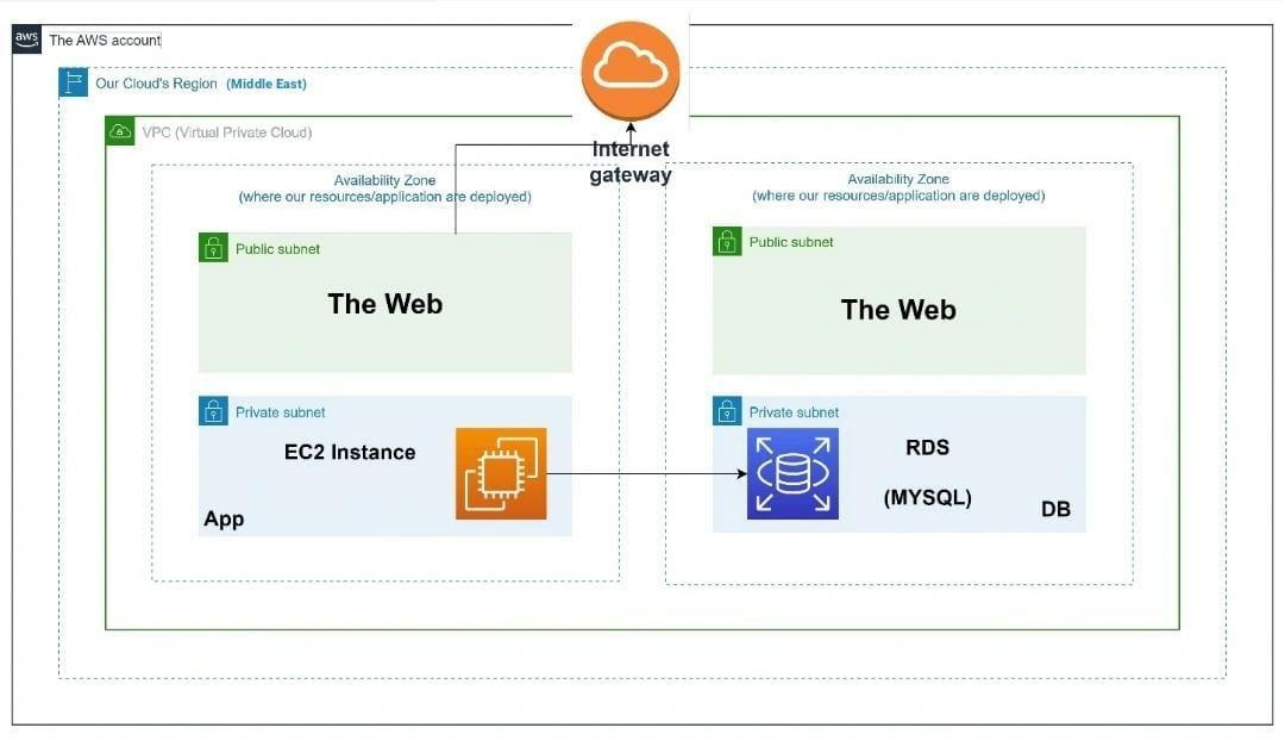
Database Schema Pt.1



Database Schema Pt.2 (Detailed)

7. Architecture of the AWS.

Amazon Web Services (AWS) is a leading and widely adopted cloud computing platform, offering a vast range of services designed to meet the needs of diverse industries.With its global infrastructure and advanced tools, AWS provides scalable, reliable, and secure solutions for businesses of all sizes, including hospitals and healthcare organizations. It is trusted by millions of users, ranging from innovative startups to large enterprises, to optimize costs, enhance operational efficiency, and accelerate innovation.



AWS is designed to provide one of the most secure and flexible cloud computing environments available today. Its robust infrastructure is built to meet the stringent security requirements of industries such as healthcare, finance, and government agencies. AWS ensures data protection through a comprehensive suite of security tools, offering over 300 security, compliance, and governance features. With support for 98 security standards and compliance certifications, AWS guarantees that data stored across its services, including those used in the HMS project, can be encrypted and safeguarded effectively. This ensures that patient and healthcare data remain confidential and secure while enabling scalability and reliability for the system.

Having said that, the AWS architecture for HMS is designed as follows:

1. **EC2 Instances:**
   HMS requires Amazon EC2 instances to host all its essential components. These instances provide scalable and reliable compute resources, enabling the system to handle varying workloads efficiently. The deployment ensures high availability and fault tolerance.

2. **RDS (Relational Database Service):**
   HMS uses Amazon RDS to store and manage its MySQL database. The RDS instance is configured with appropriate storage capacity, instance type, and automated backups to ensure data persistence, reliability, and seamless database management.

3. **IAM (Identity and Access Management):**

   AWS IAM is utilized to manage user access and permissions to AWS resources securely. It allows administrators to define fine-grained access controls, ensuring that only authorized users, such as administrators, doctors, and patients, can interact with the system's components.

These are the key components and services that are a part of the AWS architecture for HMS. They collectively ensure that the system operates in a secure, scalable, and reliable cloud environment, meeting the operational requirements of the healthcare management system.

8. Deployment on the Platform

Here are the steps in which we deployed HMS on AWS:

- **Launched the EC2 Instance:** Using the AWS Management Console, we navigated to the EC2 service and selected "Launch Instance" to create a virtual server for hosting the HMS application.

- **Chose Instance Type:** We selected an instance type that met the compute, memory, and storage requirements of HMS for optimal performance.

- **Configured the Instance:** Configured options such as the number of instances, network settings, and security groups to meet the application requirements.

- **Added Storage:** Defined the storage size and type to support the HMS application files efficiently.

- **Set Up RDS (Relational Database Service):** Configured Amazon RDS for managing the MySQL database. This included selecting the instance type, defining storage capacity, and setting up automated backups for secure database management.

- **Configured Security Groups:** Defined inbound and outbound rules to control network traffic, ensuring secure access to the EC2 instance and RDS.

- **Review and Launch:** Verified all configurations and launched the EC2 instance. The status and details of the instance were monitored using the AWS Management Console.

- **Key Pair Selection:** Generated and downloaded a key pair to securely connect to the instance from a local system for deployment and monitoring.

9. User Support in HMS

User service in HMS is designed to ensure a seamless experience for administrators, doctors, and patients. The application provides multiple touchpoints and interactions to deliver a high-quality user experience. Each of these interactions plays a vital role in shaping the user's perception of the system's efficiency and effectiveness.

1. Patient's Convenience and Accessibility:

Patients can book appointments with doctors through the application, providing them with the flexibility to select a doctor based on specialization, availability, and preferences. The system allows patients to view their appointment history, receive reminders for upcoming appointments, and access their medical records securely.

2. Doctor's Streamlined Management Tools:

Doctors can manage their schedules efficiently, including viewing upcoming appointments and their patients' basic medical history. They can also prescribe medications, add notes, and update patient records in the system, ensuring accurate and timely information for future consultations.

3. Administrator's Centralized Control:

The administrator has full control over managing the system. This includes registering new doctors, managing patient records, monitoring appointment bookings, and handling any disputes or conflicts that arise. Administrators ensure that the system operates securely by managing user roles and permissions.

4. Enhanced Data Security in the Cloud:

HMS leverages AWS services to securely store patient and doctor data in the cloud. Only authorized personnel, such as administrators and doctors, can access and modify the data. This ensures that sensitive medical records are protected from unauthorized access and tampering.

5. Automated Notification System:

HMS integrates notifications to keep users informed. Patients receive reminders for appointments and updates about their consultations, while doctors are notified of any changes in their schedules. This helps maintain effective communication between all stakeholders.

6. Doctor's Flexibility in Decision-Making:

Doctors have the autonomy to manage their interactions with patients, such as approving appointment requests, updating patient records, and prescribing medications. This ensures that each doctor has complete control over their patient management.

7. Patient's Peace of Mind with Data Management:

HMS eliminates the need for patients to worry about managing their health records manually. The cloud-based system securely stores patient records, schedules, and prescriptions, providing a comprehensive and reliable solution for their healthcare needs.

8. Simple and Intuitive Interface:

The application is designed with a user-friendly interface, making it easy for patients, doctors, and administrators to navigate and perform their tasks without technical expertise.

These features collectively ensure that HMS delivers a secure, efficient, and high-quality user experience for all its users, fostering trust and reliability in the healthcare management process.

10. Conclusion

HMS is a robust and comprehensive healthcare management system designed to streamline interactions between administrators, doctors, and patients. With features such as appointment booking, patient record management, and secure data handling, the system ensures a seamless and efficient user experience. By leveraging AWS services like EC2 instances, RDS, and IAM, HMS guarantees scalability, security, and reliable data storage. Its cloud-based architecture provides an efficient framework for managing medical records, scheduling appointments, and maintaining secure communication. Overall, HMS enhances collaboration among users, simplifies healthcare processes, and offers a user-friendly platform for delivering effective and reliable healthcare services.

References

The GitHub Report for HMS:

https://github.com/aya-alkahlot/Hospital-Management-System

https://magicsol.net/public_test/index.php