Software Requirements Specification

for

Online Doctor Consultation System with Subscription Model

Prepared by PRITAM DEY

Organization: UEM Kolkata

12th August, 2025

Table of Contents

1. Introduction

1.1 Purpose

Patient Care Management

- o Patients can browse available doctors by specialization
- o Book appointments with preferred doctors
- o Communicate directly with healthcare providers

2. Doctor Practice Management
    - o Doctors can manage their patient appointments
    - o View scheduled consultations
    - o Respond to patient inquiries and messages
3. Healthcare Communication Hub
    - o Facilitates secure messaging between patients and doctors
    - o Eliminates phone tag and improves response times
    - o Creates a digital record of patient-doctor interactions

1.2 Document Conventions

The following are the list of conventions and acronyms used in this document and the project as well.

▪ Administrator: A system user with full privileges to manage doctors, patients, and appointments (future scope).

▪ Doctor: A registered medical professional using the system to manage their appointment schedule and communicate with patients.

▪ Patient: A registered user who can book appointments, send messages to doctors, and view their appointment history.

▪ UI Layer (User Interface Layer): The front-end components of the system where patients and doctors interact via web pages with responsive design.

▪ Application Logic Layer: The part of the system handling the core business logic (e.g., appointment booking, user authentication, message handling).

▪ Data Storage Layer: The MySQL database layer where patient, doctor, appointment, and message information is stored.

▪ SQL: Structured Query Language used for database operations and schema creation.

▪ PHP: Hypertext Preprocessor; the server-side scripting language used for implementing the system logic.

▪ MySQLi: MySQL Improved extension used for secure database connectivity with prepared statements.

▪ Primary Key: A unique identifier for each record in the database (e.g., patient ID, doctor ID, appointment ID).

▪ Foreign Key: A field that creates a link between two tables (e.g., patient_id in appointments table references patients table).

▪ Glass-morphism: Modern UI design technique using translucent backgrounds with blur effects for sophisticated appearance.

▪ Responsive Design: Web design approach ensuring optimal viewing experience across desktop and mobile devices.

▪ XAMPP: Cross-platform web server solution stack package containing Apache, MySQL, PHP for local development environment.

1.3 Intended Audience and Reading Suggestions

- Developers: Refer to Sections 3 and 4 for detailed feature requirements.

- Project Managers: Review Sections 1 and 2 for scope and objectives.

- Testers: Focus on functional requirements (Section 4) and non-functional requirements (Section 5).

- Marketing Team: Check subscription models in Section 4.2.

1.4 Product Scope

Current Scope: Basic hospital management with doctor/patient registration, appointment booking, messaging system, and specialization-based doctor filtering using PHP/MySQL web platform.

Future Scope: Can expand to include medical records management, telemedicine, payment integration, prescription systems, admin panels, mobile apps, and multi-hospital network support.

Target Market: Small to medium hospitals, clinics, and individual medical practitioners seeking digital transformation of their patient management processes.

Scalability: Designed for single hospital use but can scale to multi-location healthcare chains and regional healthcare networks with enhanced features and cloud deployment.

1.5 References

- ISO/IEC/IEEE 29148:2018 – Systems and software engineering — Life cycle processes — Requirements engineering.

- W3C WebRTC Specification for Real-Time Communication in Web Applications.

2. Overall Description

2.1 Product Perspective

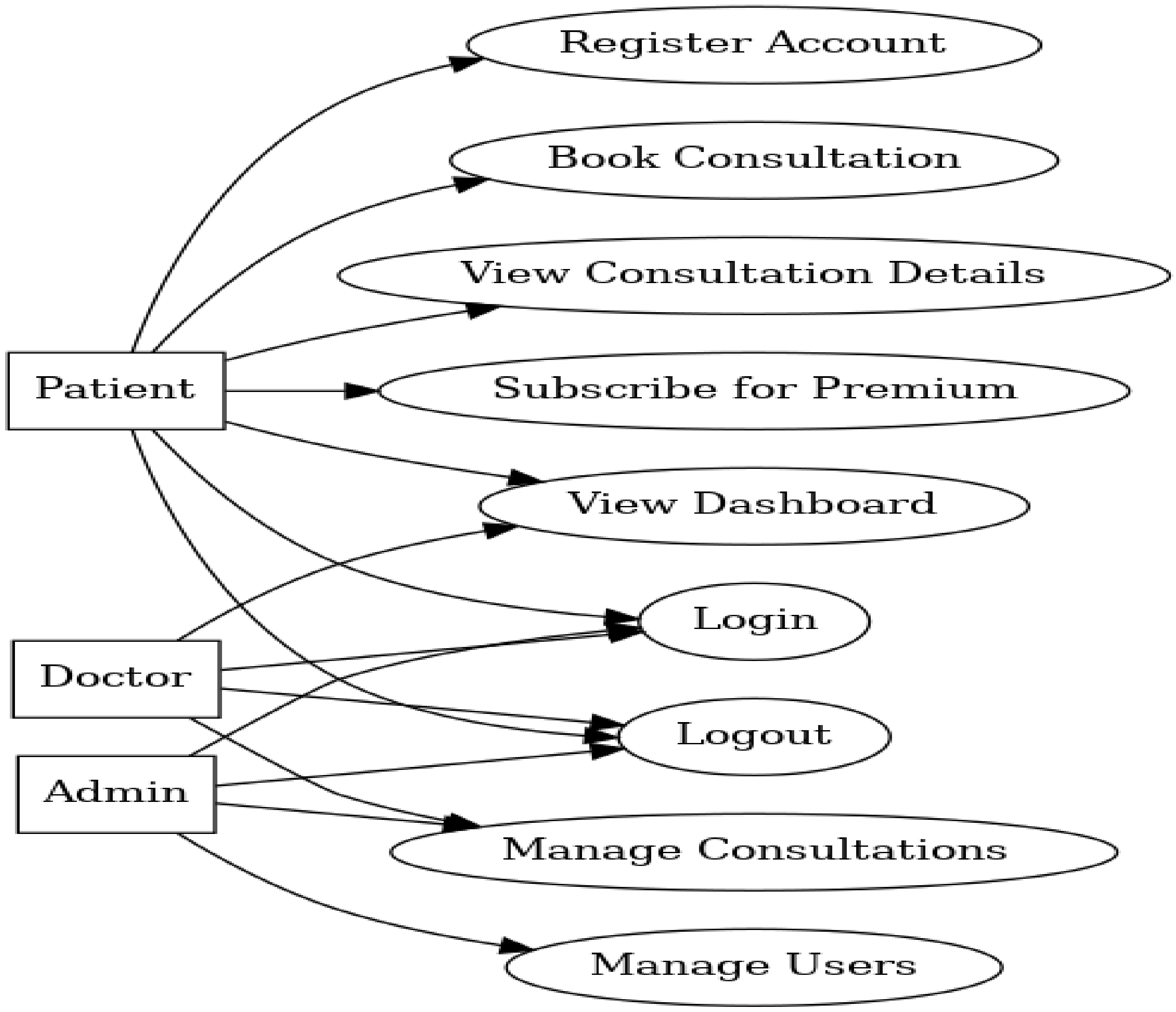
The Doctor Appointment Booking System is a standalone, web-based platform developed to modernize and streamline the process of scheduling medical appointments. It serves as an alternative or complement to traditional manual and phone-based booking methods used in healthcare facilities. While it functions independently, the system can be seamlessly integrated with hospital management software, electronic patient record systems, or third-party medical databases when needed.

2.2 Product Functions

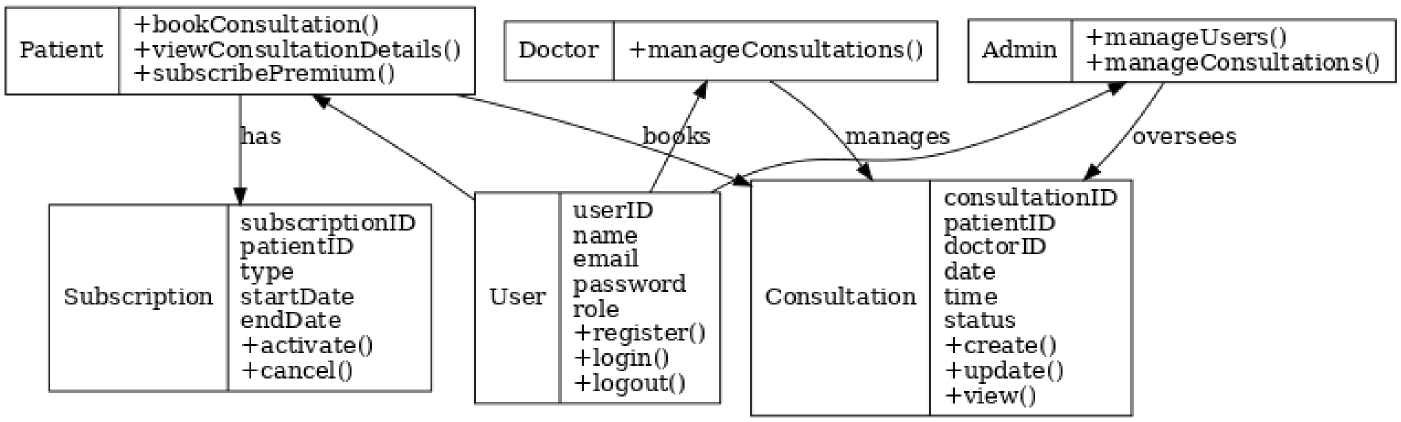
- Patient registration and login with profile management.
- Doctor registration with specialization verification.
- Appointment booking with date/time selection.
- Bidirectional messaging system between patients and doctors.
- Doctor search and filtering by specialization.
- Appointment history and status tracking.
- Session-based authentication and security.

2.3 User Classes and Characteristics

- Patients: General public seeking medical appointments with basic computer knowledge.
- Doctors: Medical professionals managing appointments and patient communication.
- System Users: Both groups require internet access and basic web navigation skills.



Register Account
Book Consultation
View Consultation Details
Subscribe for Premium
View Dashboard
Login
Logout
Manage Consultations
Manage Users
Patient
Doctor
Admin



Patient
+bookConsultation()
+viewConsultationDetails()
+subscribePremium()
Doctor
+manageConsultations()
Admin
+manageUsers()
+manageConsultations()
has
books
manages
oversees
Subscription
subscriptionID
patientID
type
startDate
endDate
+activate()
+cancel()
User
userID
name
email
password
role
+register()
+login()
+logout()
Consultation
consultationID
patientID
doctorID
date
time
status
+create()
+update()
+view()

2.4 Operating Environment

- Web Browsers: Chrome, Firefox, Edge, Safari (modern versions).
- Server: Apache/Nginx with PHP 7.4+ support.
- Database: MySQL 5.7+ or MariaDB.
- Platform: Cross-platform web application accessible on desktop and mobile browsers.

2.5 Design and Implementation Constraints

- Must use MySQLi (not PDO) for database connectivity.
- Password security using PHP's built-in hashing functions.
- Responsive design for mobile compatibility.
- Session-based authentication without external dependencies.

2.6 User Documentation

- README.md with setup instructions and system overview. Inline help text and user-friendly error messages
- Database schema documentation for administrators.

2.7 Assumptions and Dependencies

- Users have stable internet connection for web access., MySQL server is properly configured and accessible.
- PHP environment supports MySQLi extension. XAMPP or similar local development environment for testing.

3. External Interface Requirements

3.1 User Interfaces

- Web Application: Responsive design with intuitive navigation for both desktop and mobile browsers.
- Patient Dashboard: Doctor browsing, appointment booking, and messaging interface.
- Doctor Dashboard: Appointment management and patient communication portal.
- Authentication Pages: Modern login/registration forms with glass-morphism design.

Patient Interface

• ***Registration Page*** *– Allows patients to create an account by providing username, password, full name, email, and phone number with secure password hashing.*

• ***Login Page*** *– Secure authentication system with username/password validation and session management for system access.*

• ***Patient Dashboard*** *– Displays available doctors with specialization filtering, appointment booking forms, appointment history, and messaging interface.*

• ***Doctor Browse Section*** *– View all registered doctors with their specializations, contact details, and real-time appointment booking capability.*

• ***Book Appointment Form*** *– Select preferred doctor, appointment date, and time with instant booking confirmation and database storage.*

• ***Appointment History*** *– Shows all booked appointments with doctor names, specializations, dates, times, and current status (pending/confirmed).*

• ***Messaging System*** *– Send messages directly to doctors with text input forms and view message history with timestamps.*

• ***Logout Option*** *– Securely ends the patient session and redirects to home page.*

Doctor Interface

• ***Registration Page*** *– Allows doctors to create accounts with username, password, full name, email, phone number, and medical specialization selection.*

• ***Login Page*** *– Secure doctor authentication with credential validation and specialized doctor session management.*

• ***Doctor Dashboard*** *– Shows scheduled patient appointments with patient details, appointment times, and comprehensive messaging interface.*

• ***Patient Appointments Section*** *– View all booked appointments with patient names, contact information, appointment dates/times, and booking timestamps.*

• ***Message Management*** *– View messages received from patients with sender details and reply functionality through integrated messaging forms.*

• ***Reply System*** *– Respond to patient messages with text input forms and maintain conversation threads with timestamps.*

• ***Appointment Overview*** *– Complete list of scheduled consultations with patient contact details and appointment status tracking.*

• ***Logout Option*** *– Securely ends the doctor session and returns to main portal selection page.*

3.2Software Interfaces

- MySQL Database: MySQLi connectivity for data storage and retrieval.and PHP Server: Apache/Nginx web server with PHP 7.4+ support.

- Email Services: Potential integration for appointment confirmations (future scope). Web Browser APIs: HTML5 form validation and responsive design features.

3.3 Communications Interfaces

- HTTP/HTTPS: Standard web protocols for client-server communication.

- MySQL Protocol: Database connectivity via MySQLi extension.

- Session Management: PHP session handling for user authentication. Form Data: POST/GET methods for secure data transmission between client and server.

4. System Features

4.1 Appointment Booking

4.1.1 Description and Priority

The Appointment Booking feature enables patients to schedule consultations with available doctors through a simple web interface. Patients can browse doctors by specialization, select preferred date/time slots, and book appointments directly.

Priority: High

Component Ratings:
• Benefit: 9 (Core functionality of the hospital management system)
• Penalty: 9 (System would be ineffective without this feature)
• Cost: 4 (Simple form-based implementation with database storage)
• Risk: 3 (Low risk with proper validation and database constraints)

4.1.2 Stimulus/Response Sequences

• Stimulus: Patient logs in and views the doctor list on dashboard.
Response: System displays available doctors with specialization, contact details, and appointment booking forms.

• Stimulus: Patient selects a doctor, enters appointment date and time.
Response: System validates the input and stores the appointment in the database.

• Stimulus: Patient submits appointment booking form.
Response: Appointment is saved with "pending" status, and patient is redirected to dashboard with confirmation.

• Stimulus: Doctor views their dashboard.
Response: System displays all scheduled appointments with patient details and timing.

4.1.3 Functional Requirements

• REQ-1: The system shall allow patients to view all registered doctors with their specializations.
• REQ-2: The system shall provide date and time input fields for appointment scheduling.
• REQ-3: The system shall allow patients to filter doctors by specialization using dropdown selection.
• REQ-4: The system shall store appointment details in the database with patient-doctor relationship.
• REQ-5: The system shall display appointment history for both patients and doctors.
• REQ-6: The system shall validate that appointment date is not in the past.
• REQ-7: The system shall maintain appointment status tracking (pending/confirmed/completed).

4.2 Messaging System

4.2.1 Description and Priority

The Messaging System enables bidirectional communication between patients and doctors through a simple text-based interface. Patients can send queries to doctors, and doctors can respond directly through their dashboard.

Priority: High

Component Ratings:
• Benefit: 8 (Enhances patient-doctor communication and engagement)
• Penalty: 6 (System remains functional but less interactive without messaging)
• Cost: 3 (Simple form-based messaging with database storage)
• Risk: 2 (Very low risk with basic text message handling)

4.2.2 Stimulus/Response Sequences

• Stimulus: Patient types message in doctor's message form and submits.
Response: System stores message in database and displays confirmation to patient.

• Stimulus: Doctor logs into dashboard.
Response: System displays all received messages from patients with sender details.

• Stimulus: Doctor types reply in patient message form and submits.
Response: System stores reply in database and displays confirmation to doctor.

• Stimulus: Patient views dashboard messages section.
Response: System displays all sent messages and any replies received from doctors.

4.2.3 Functional Requirements

• REQ-10: The system shall allow patients to send text messages to any registered doctor.
• REQ-11: The system shall allow doctors to reply to patient messages through their dashboard.
• REQ-12: The system shall store all messages with sender/receiver identification and timestamps.
• REQ-13: The system shall display message history for both patients and doctors.
• REQ-14: The system shall validate that message content is not empty before submission.
• REQ-15: The system shall maintain message threading between specific patient-doctor pairs.
• REQ-16: The system shall display messages in chronological order with clear sender identification.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Support at least 100 concurrent users with response time under 3 seconds
- Database queries should execute within 2 seconds for standard operations
- Page load time should not exceed 5 seconds on standard broadband connections

5.2 Safety Requirements

- Password hashing using PHP's built-in security functions
- Secure session management to prevent unauthorized access
- Input validation to prevent data corruption and system errors

5.3 Security Requirements

- MySQLi prepared statements to prevent SQL injection attacks
- Session-based authentication with automatic logout on inactivity
- Secure password storage using password_hash() and password_verify() functions
- Form validation on both client-side and server-side

5.4 Software Quality Attributes

- Availability: 95% uptime during business hours with proper error handling
- Usability: Intuitive web interface with responsive design for mobile compatibility
- Maintainability: Clean code structure with separate files for different functionalities
- Reliability: Consistent database operations with proper error messages for users
- Scalability: Modular design allowing future feature additions without major restructuring

5.5 Scalability Requirement

- *The system architecture must allow scaling to support* ***more users and features*** *without major redesign.*
- *Database design must accommodate additional tables for future features (e.g., prescription storage, video consultations).*

6. Other Requirements

6.1 Database

The system shall use MySQL relational database to store user profiles (doctors and patients), appointment data, messaging records, and authentication information. Database shall maintain referential integrity through foreign key constraints and support concurrent access through proper indexing.

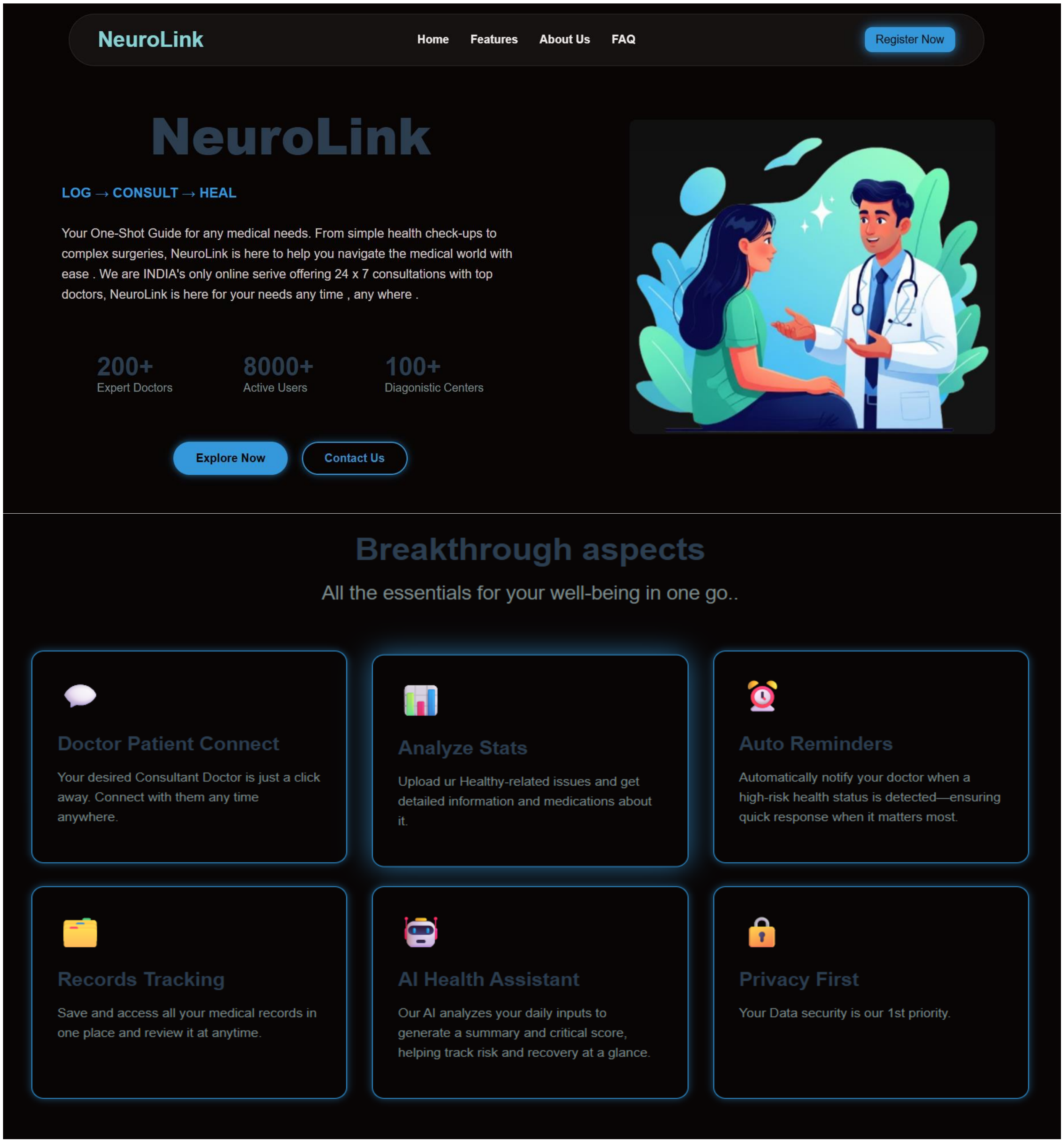
Appendix A: Glossary

• HMS: Hospital Management System – web-based platform connecting patients and doctors.
• MySQLi: MySQL Improved extension – PHP interface for accessing MySQL databases.
• Session Management: Server-side storage of user authentication state during web browsing.
• Prepared Statements: Pre-compiled SQL statements that prevent injection attacks.
• Responsive Design: Web design approach ensuring optimal viewing across different devices.
• Glass-morphism: Modern UI design technique using translucent backgrounds with blur effects.
• XAMPP: Cross-platform web server solution stack package for local development.

Appendix B: Analysis Models

• Data Flow Diagram (DFD): Shows data flow between patients, doctors, appointment booking system, messaging system, and MySQL database.
• Entity-Relationship Diagram (ERD): Models relationships between entities: Patient, Doctor, Appointment, and Message tables with their attributes and foreign key relationships.
• Use Case Diagrams: Depict scenarios such as patient registration, doctor login, appointment booking, message sending, and dashboard viewing.

HOME PAGE



NeuroLink

LOG → CONSULT → HEAL

Your One-Shot Guide for any medical needs. From simple health check-ups to complex surgeries, NeuroLink is here to help you navigate the medical world with ease . We are INDIA's only online serive offering 24 x 7 consultations with top doctors, NeuroLink is here for your needs any time , any where .

200+	8000+	100+
Expert Doctors	Active Users	Diagonistic Centers

Breakthrough aspects

All the essentials for your well-being in one go..

Doctor Patient Connect

Your desired Consultant Doctor is just a click away. Connect with them any time anywhere.

Analyze Stats

Upload ur Healthy-related issues and get detailed information and medications about it.

Auto Reminders

Automatically notify your doctor when a high-risk health status is detected—ensuring quick response when it matters most.

Records Tracking

Save and access all your medical records in one place and review it at anytime.

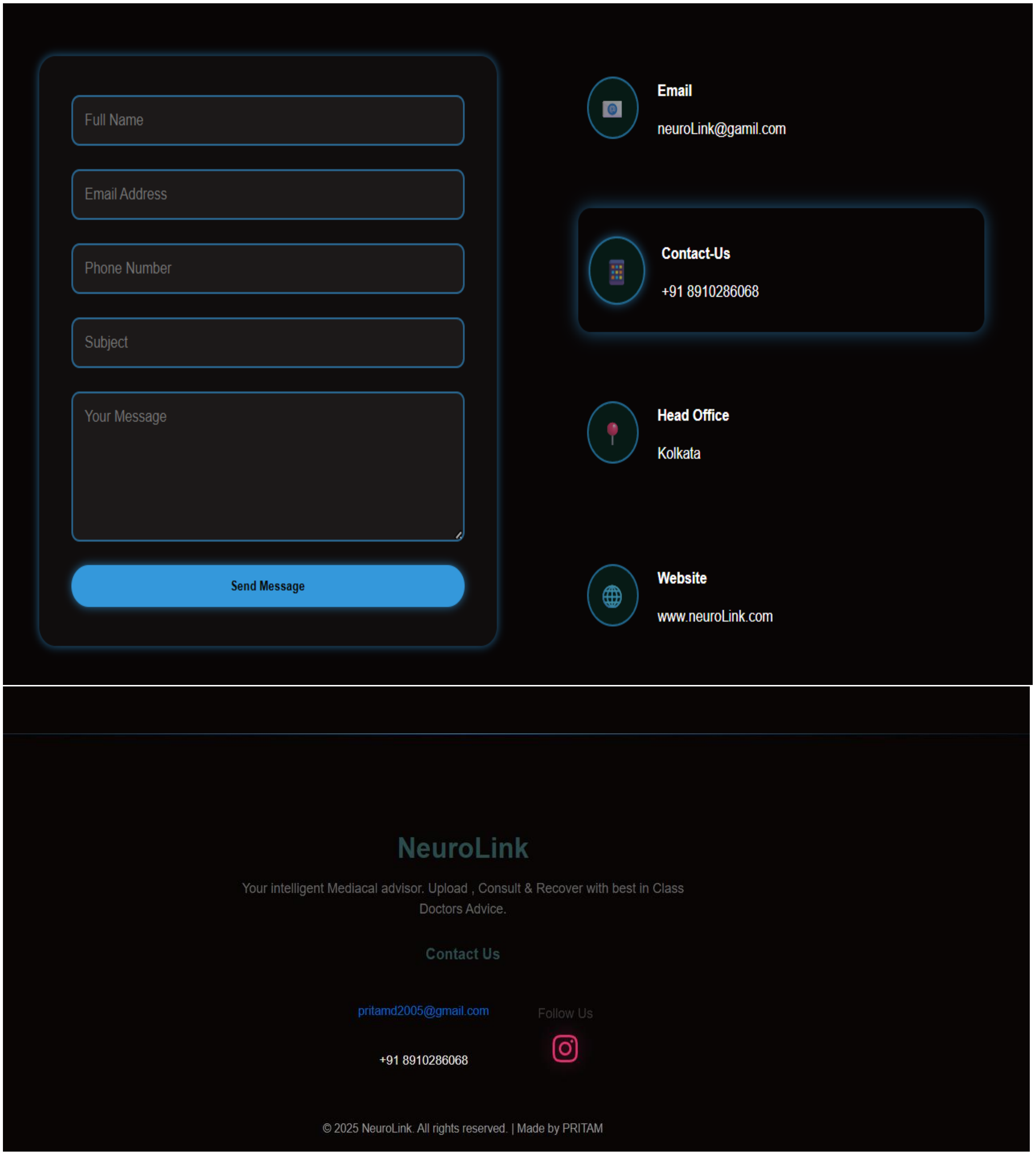
AI Health Assistant

Our AI analyzes your daily inputs to generate a summary and critical score, helping track risk and recovery at a glance.

Privacy First

Your Data security is our 1st priority.

REGISTRATION PAGE


Full Name

Email Address

Phone Number

Subject

Your Message

Send Message

Email

neuroLink@gamil.com

Contact-Us

+91 8910286068

Head Office

Kolkata

Website

www.neuroLink.com

NeuroLink

Your intelligent Mediacal advisor. Upload , Consult & Recover with best in Class Doctors Advice.

Contact Us

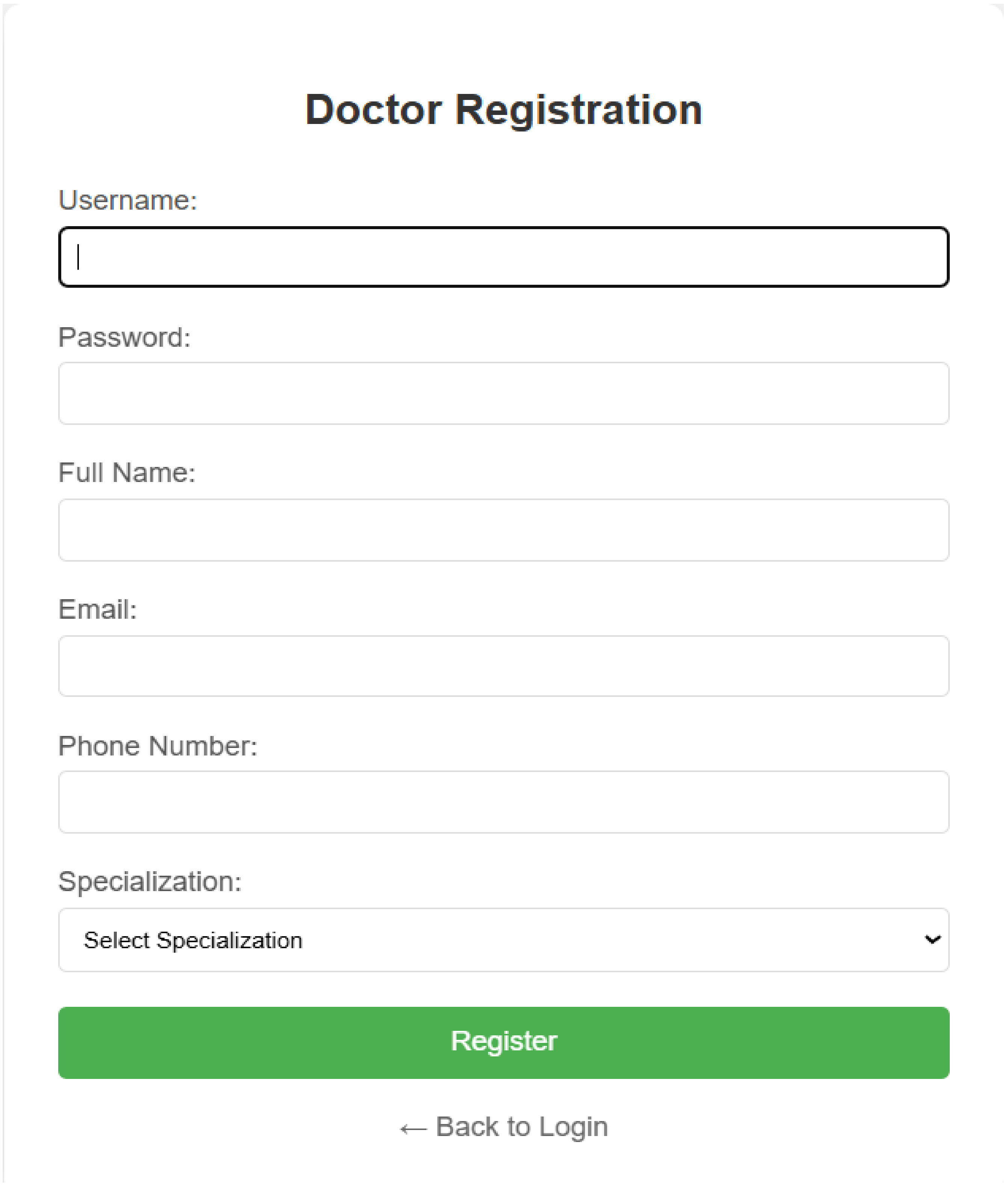
pritamd2005@gmail.com

+91 8910286068

Follow Us

© 2025 NeuroLink. All rights reserved. | Made by PRITAM

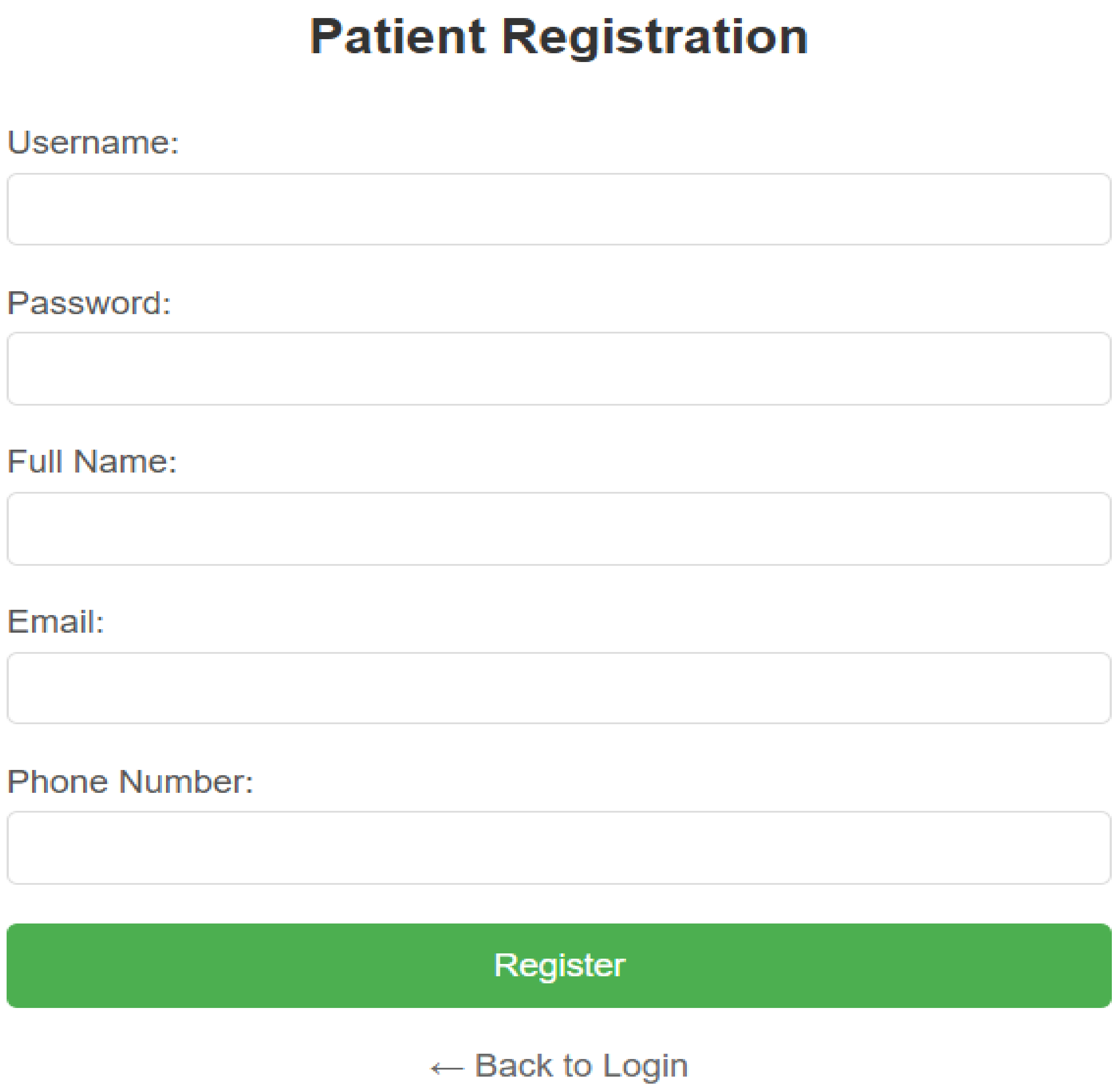
DOCTOR LOGIN



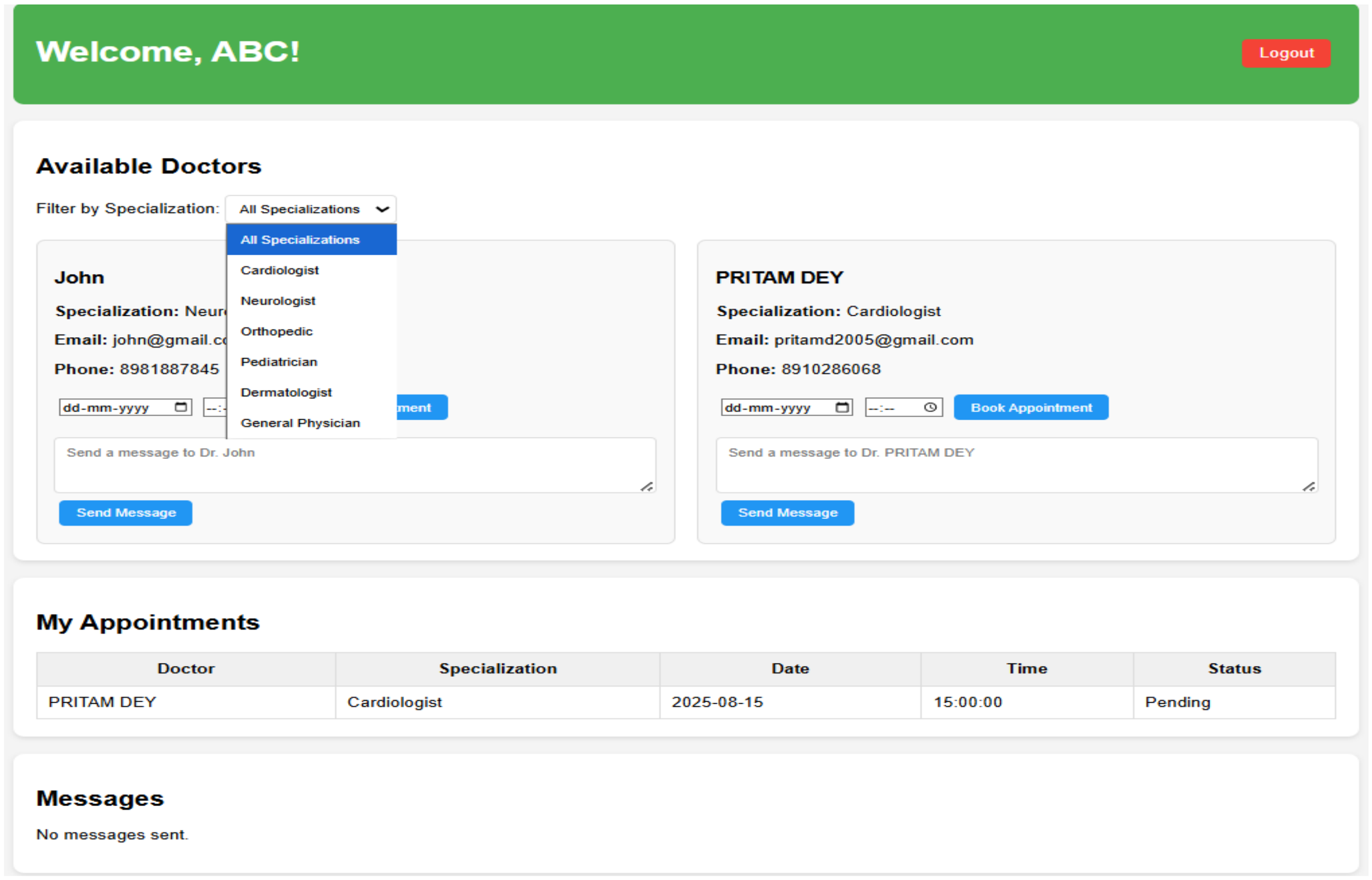
Doctor Registration

Username:	
Password:	
Full Name:	
Email:	
Phone Number:	
Specialization:	Select Specialization

PATIENT LOGIN



PATIENT DASHBOARD



Welcome, ABC!

Available Doctors

Filter by Specialization: All Specializations

John

Specialization: Neur

Email: john@gmail.c

Phone: 8981887845

PRITAM DEY

Specialization: Cardiologist

Email: pritamd2005@gmail.com

Phone: 8910286068

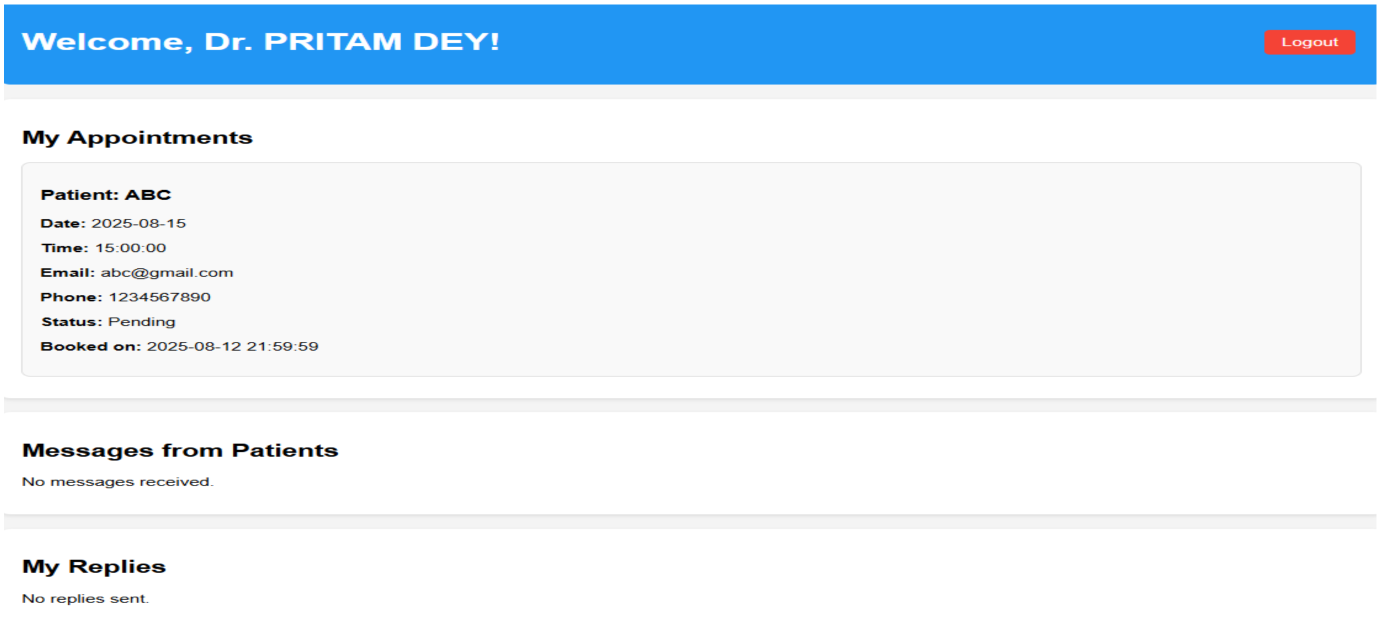
My Appointments

Doctor	Specialization	Date	Time	Status
PRITAM DEY	Cardiologist	2025-08-15	15:00:00	Pending

Messages

No messages sent.

DOCTOR'S DASHBOARD



Welcome, Dr. PRITAM DEY!

My Appointments

Patient: ABC
Date: 2025-08-15
Time: 15:00:00
Email: abc@gmail.com
Phone: 1234567890
Status: Pending
Booked on: 2025-08-12 21:59:59

Messages from Patients

No messages received.

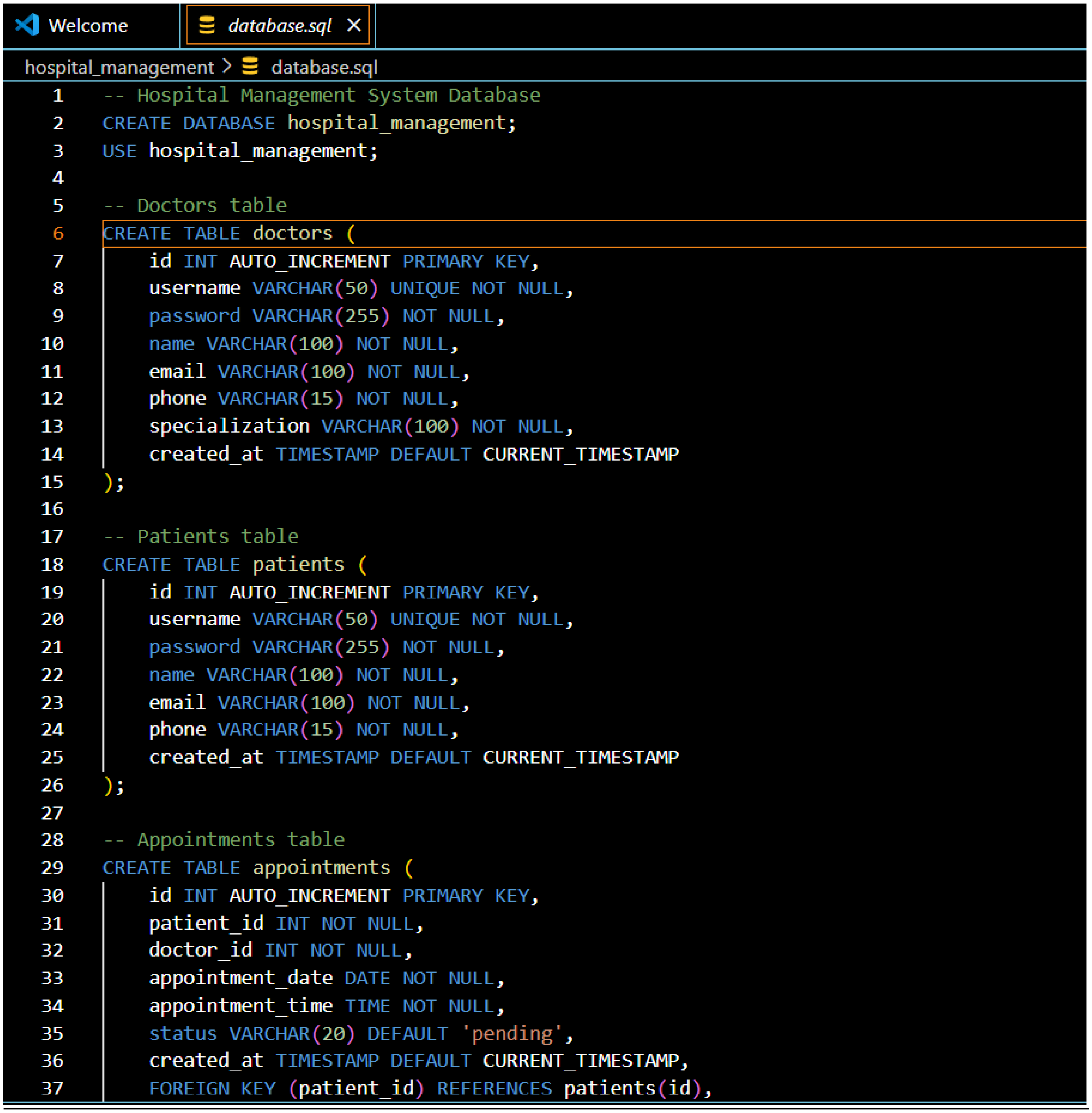
My Replies

No replies sent.

Appendix C: Code Snippets

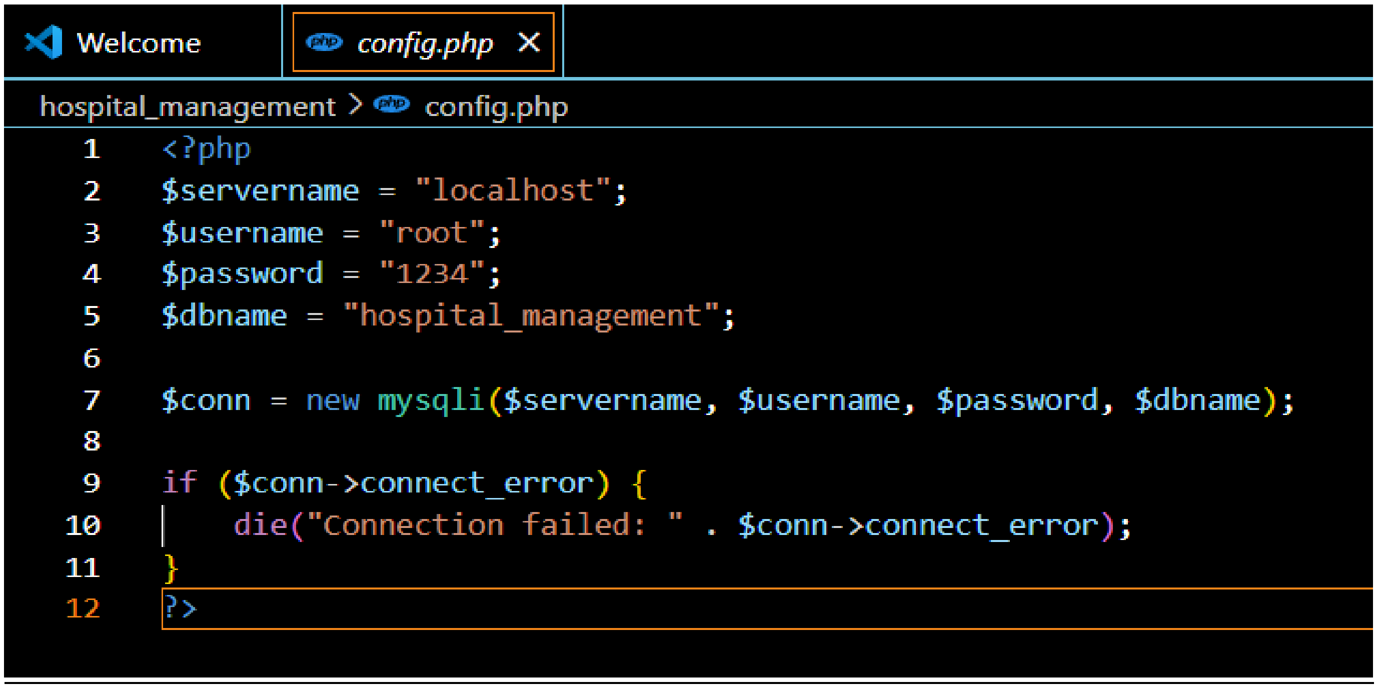
Here are the code snippets for the website present above:

SQL SNIPPETS



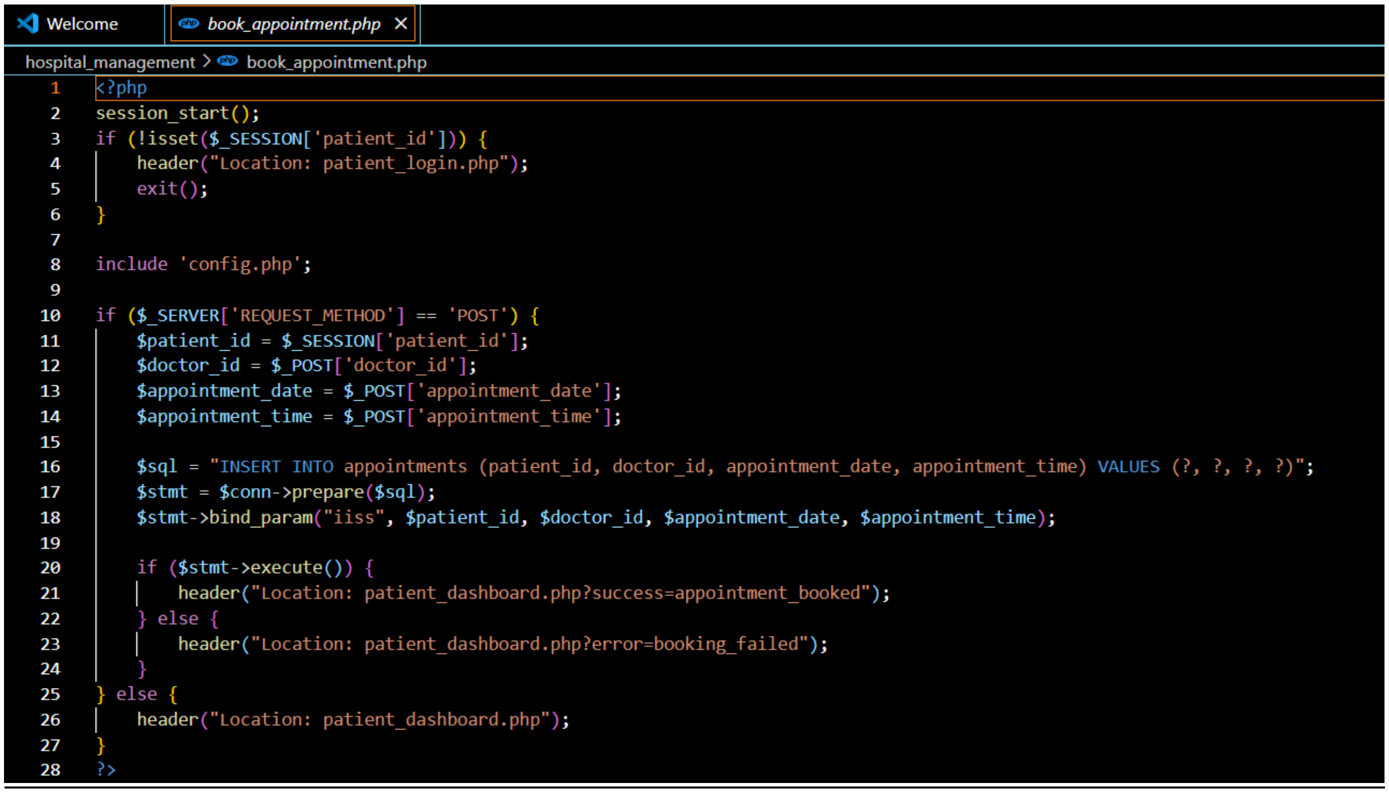
```sql
-- Hospital Management System Database
CREATE DATABASE hospital_management;
USE hospital_management;

-- Doctors table
CREATE TABLE doctors (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    specialization VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Patients table
CREATE TABLE patients (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Appointments table
CREATE TABLE appointments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT NOT NULL,
    doctor_id INT NOT NULL,
    appointment_date DATE NOT NULL,
    appointment_time TIME NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (patient_id) REFERENCES patients(id),
```

CONFIG.PHP



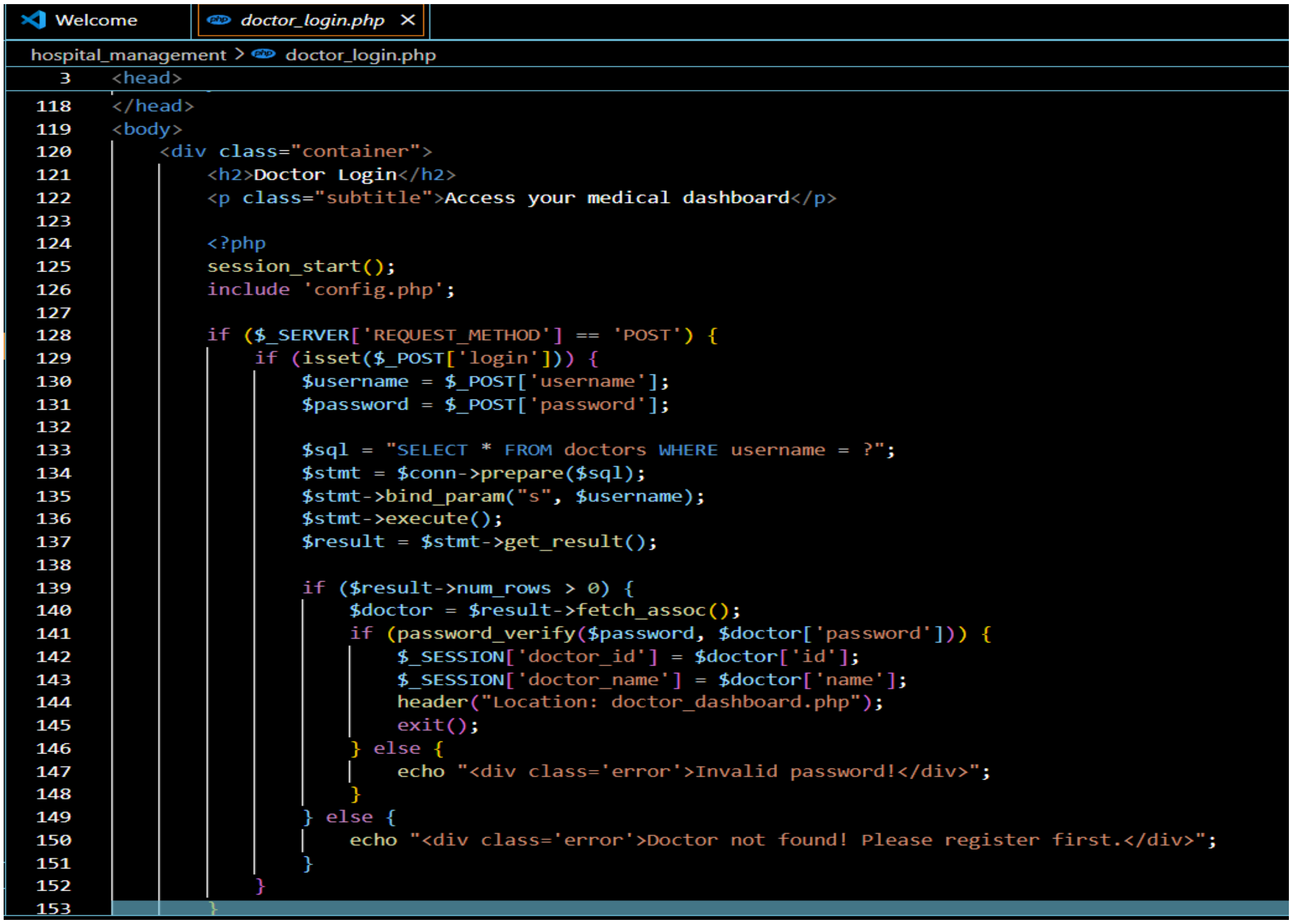
```php
<?php
$servername = "localhost";
$username = "root";
$password = "1234";
$dbname = "hospital_management";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

BOOK_APPOINTMENT.PHP

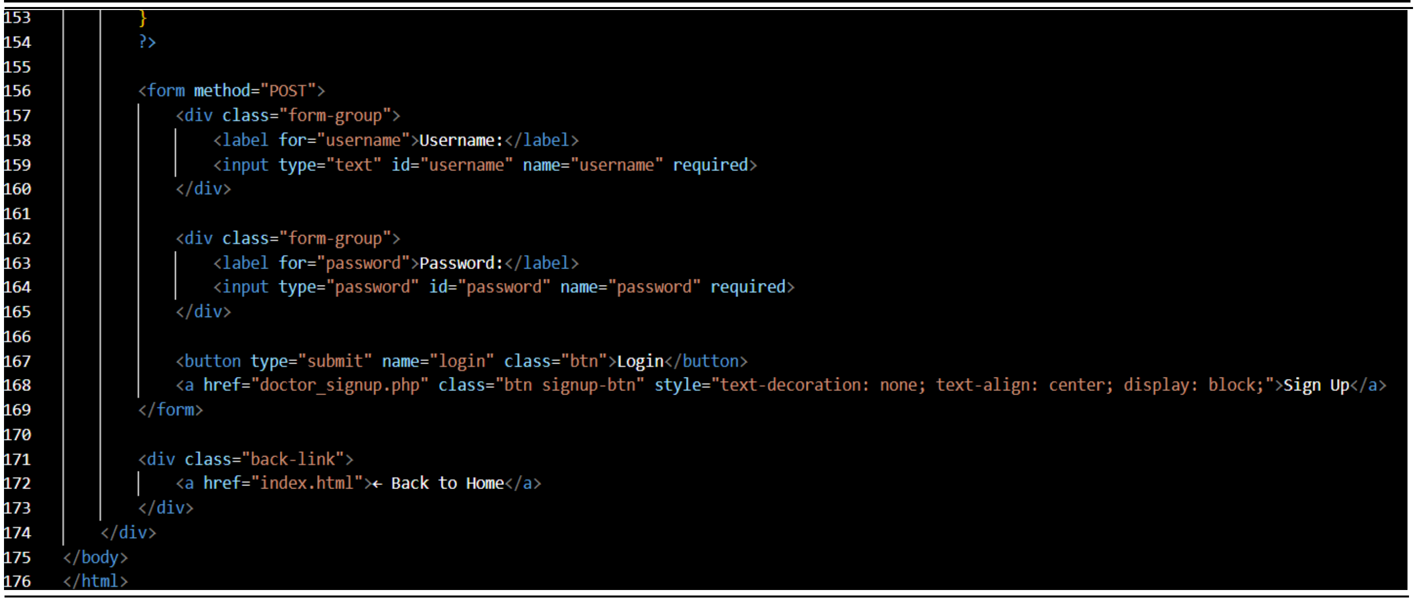


```php
<?php
session_start();
if (!isset($_SESSION['patient_id'])) {
    header("Location: patient_login.php");
    exit();
}

include 'config.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $patient_id = $_SESSION['patient_id'];
    $doctor_id = $_POST['doctor_id'];
    $appointment_date = $_POST['appointment_date'];
    $appointment_time = $_POST['appointment_time'];

    $sql = "INSERT INTO appointments (patient_id, doctor_id, appointment_date, appointment_time) VALUES (?, ?, ?, ?)";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("iiss", $patient_id, $doctor_id, $appointment_date, $appointment_time);

    if ($stmt->execute()) {
        header("Location: patient_dashboard.php?success=appointment_booked");
    } else {
        header("Location: patient_dashboard.php?error=booking_failed");
    }
} else {
    header("Location: patient_dashboard.php");
}
?>
```

DOCTOR_LOGIN.PHP

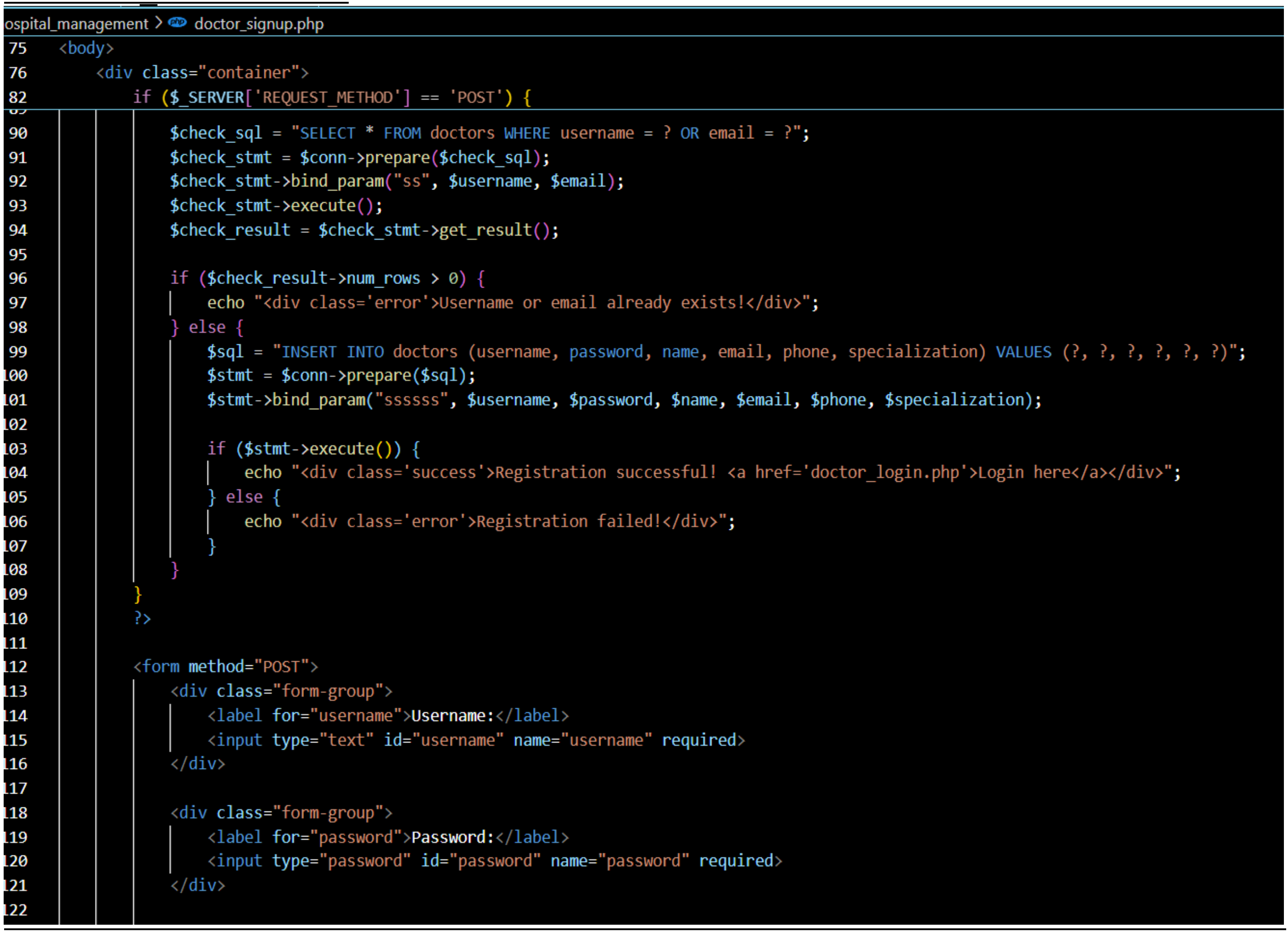


```
<head>
</head>
<body>
    <div class="container">
        <h2>Doctor Login</h2>
        <p class="subtitle">Access your medical dashboard</p>

        <?php
        session_start();
        include 'config.php';

        if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            if (isset($_POST['login'])) {
                $username = $_POST['username'];
                $password = $_POST['password'];

                $sql = "SELECT * FROM doctors WHERE username = ?";
                $stmt = $conn->prepare($sql);
                $stmt->bind_param("s", $username);
                $stmt->execute();
                $result = $stmt->get_result();

                if ($result->num_rows > 0) {
                    $doctor = $result->fetch_assoc();
                    if (password_verify($password, $doctor['password'])) {
                        $_SESSION['doctor_id'] = $doctor['id'];
                        $_SESSION['doctor_name'] = $doctor['name'];
                        header("Location: doctor_dashboard.php");
                        exit();
                    } else {
                        echo "<div class='error'>Invalid password!</div>";
                    }
                } else {
                    echo "<div class='error'>Doctor not found! Please register first.</div>";
                }
            }
```



```
        }
        ?>

        <form method="POST">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
            </div>

            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
            </div>

            <button type="submit" name="login" class="btn">Login</button>
            <a href="doctor_signup.php" class="btn signup-btn" style="text-decoration: none; text-align: center; display: block;">Sign Up</a>
        </form>

        <div class="back-link">
            <a href="index.html">← Back to Home</a>
        </div>
    </div>
</body>
</html>
```

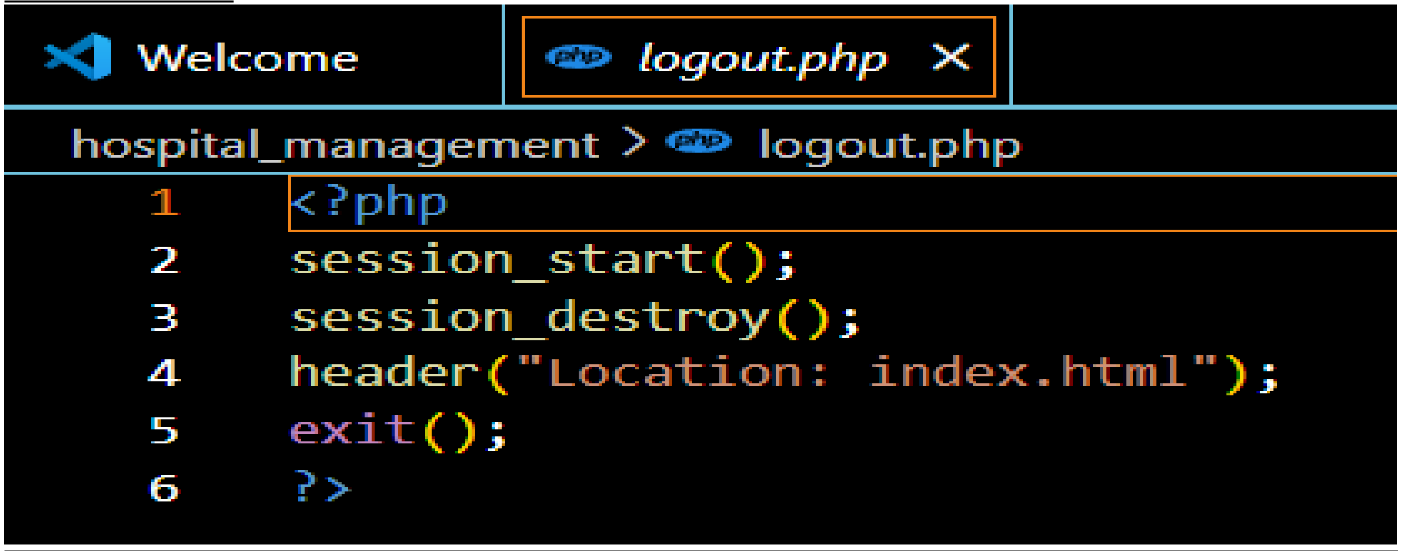
DOCTOR_SIGNUP.PHP



ospital_management > doctor_signup.php

```
<body>
    <div class="container">
        if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            $check_sql = "SELECT * FROM doctors WHERE username = ? OR email = ?";
            $check_stmt = $conn->prepare($check_sql);
            $check_stmt->bind_param("ss", $username, $email);
            $check_stmt->execute();
            $check_result = $check_stmt->get_result();

            if ($check_result->num_rows > 0) {
                echo "<div class='error'>Username or email already exists!</div>";
            } else {
                $sql = "INSERT INTO doctors (username, password, name, email, phone, specialization) VALUES (?, ?, ?, ?, ?, ?)";
                $stmt = $conn->prepare($sql);
                $stmt->bind_param("ssssss", $username, $password, $name, $email, $phone, $specialization);

                if ($stmt->execute()) {
                    echo "<div class='success'>Registration successful! <a href='doctor_login.php'>Login here</a></div>";
                } else {
                    echo "<div class='error'>Registration failed!</div>";
                }
            }
        }
        ?>

        <form method="POST">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
            </div>

            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
            </div>

```

LOGOUT.PHP

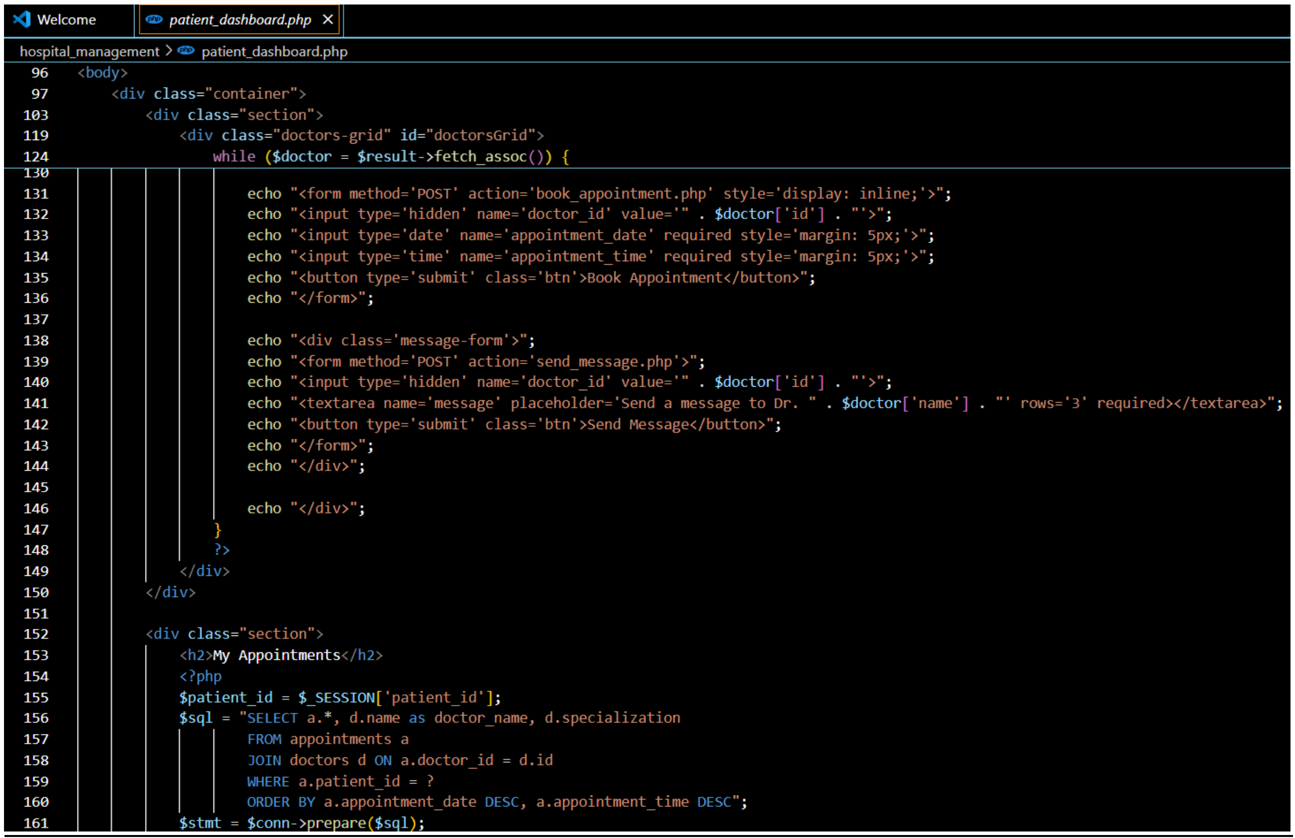


Welcome | logout.php

hospital_management > logout.php

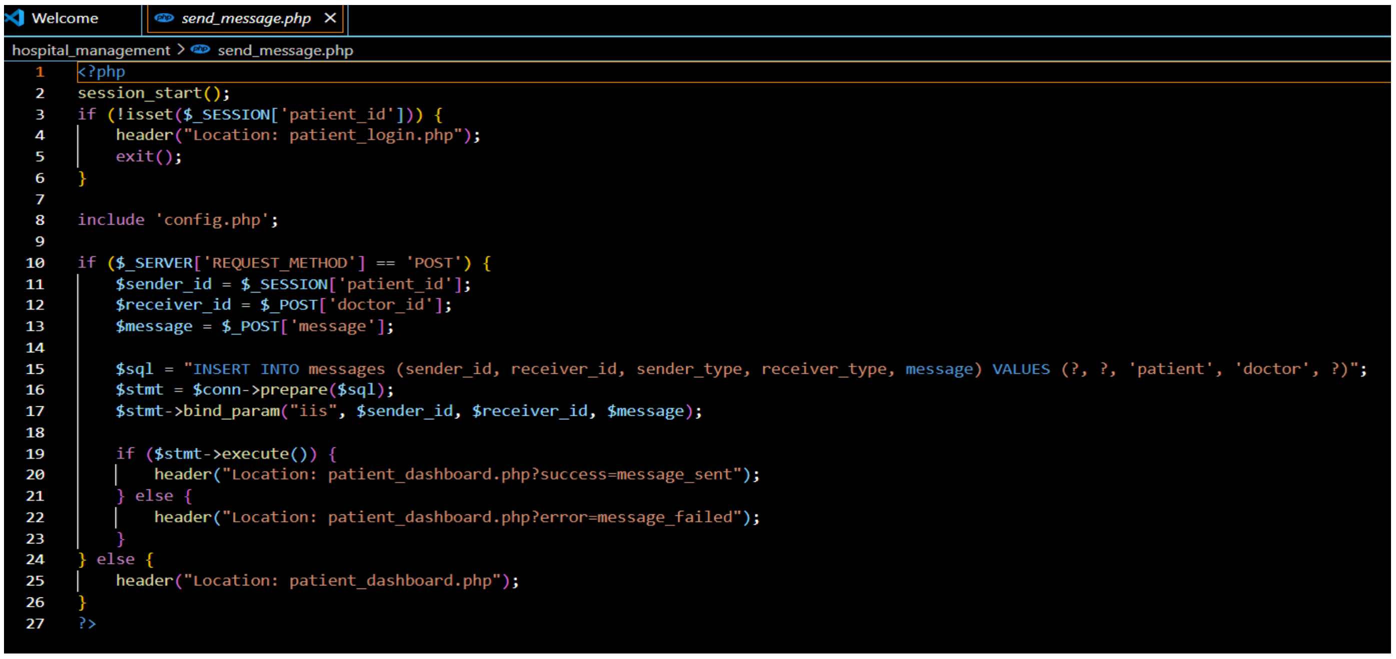
```
<?php
session_start();
session_destroy();
header("Location: index.html");
exit();
?>
```

PATIENT_DASHBOARD.PHP



```
<body>
    <div class="container">
        <div class="section">
            <div class="doctors-grid" id="doctorsGrid">
                while ($doctor = $result->fetch_assoc()) {
                    echo "<form method='POST' action='book_appointment.php' style='display: inline;'>";
                    echo "<input type='hidden' name='doctor_id' value='" . $doctor['id'] . "'>";
                    echo "<input type='date' name='appointment_date' required style='margin: 5px;'>";
                    echo "<input type='time' name='appointment_time' required style='margin: 5px;'>";
                    echo "<button type='submit' class='btn'>Book Appointment</button>";
                    echo "</form>";

                    echo "<div class='message-form'>";
                    echo "<form method='POST' action='send_message.php'>";
                    echo "<input type='hidden' name='doctor_id' value='" . $doctor['id'] . "'>";
                    echo "<textarea name='message' placeholder='Send a message to Dr. " . $doctor['name'] . "' rows='3' required></textarea>";
                    echo "<button type='submit' class='btn'>Send Message</button>";
                    echo "</form>";
                    echo "</div>";

                    echo "</div>";
                }
                ?>
            </div>
        </div>

        <div class="section">
            <h2>My Appointments</h2>
            <?php
            $patient_id = $_SESSION['patient_id'];
            $sql = "SELECT a.*, d.name as doctor_name, d.specialization
                    FROM appointments a
                    JOIN doctors d ON a.doctor_id = d.id
                    WHERE a.patient_id = ?
                    ORDER BY a.appointment_date DESC, a.appointment_time DESC";
            $stmt = $conn->prepare($sql);
```

SEND_MESSAGE.PHP



```
<?php
session_start();
if (!isset($_SESSION['patient_id'])) {
    header("Location: patient_login.php");
    exit();
}

include 'config.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $sender_id = $_SESSION['patient_id'];
    $receiver_id = $_POST['doctor_id'];
    $message = $_POST['message'];

    $sql = "INSERT INTO messages (sender_id, receiver_id, sender_type, receiver_type, message) VALUES (?, ?, 'patient', 'doctor', ?)";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("iis", $sender_id, $receiver_id, $message);

    if ($stmt->execute()) {
        header("Location: patient_dashboard.php?success=message_sent");
    } else {
        header("Location: patient_dashboard.php?error=message_failed");
    }
} else {
    header("Location: patient_dashboard.php");
}
?>
```

Appendix D: To Be Determined List

Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.

TBD-1 Implement email notification system for appointment confirmations
TBD-2 Add file upload capability for medical documents
TBD-3 Integrate payment gateway for consultation fees (future enhancement)
TBD-4 Determine hosting environment (local server vs cloud deployment)
TBD-5 Add appointment cancellation and rescheduling functionality
TBD-6 Implement admin panel for system management