1. INTRODUCTION

1.1 Problem Statement

Designing a comprehensive Doctor Appointment system that integrates user-friendly interfaces for patients and practitioners, automates scheduling processes, manages patient records securely, and provides real-time availability updates to optimize healthcare delivery. The system aims to enhance efficiency in appointment management, reduce wait times, and improve patient satisfaction while facilitating seamless communication between healthcare providers and patients.

1.2 Proposed System

In the envisioned Doctor Appointment system, doctors will be empowered to manage their availability efficiently by updating their timeslots as per their schedules. This functionality will ensure that their time is utilized optimally while providing patients with a range of options to choose from. Patients will have a user-friendly interface at their disposal, allowing them to easily browse through available timeslots and book appointments that align with their preferences and needs. Additionally, the system will integrate a secure payment gateway powered by Stripe, offering a seamless and reliable transaction experience for both patients and healthcare providers. This feature will eliminate any friction in the payment process, ensuring smooth transactions and enhancing the overall user experience.

Furthermore, the system will offer a premium upgrade option for patients, granting them access to exclusive insights such as popularly booked doctors and personalized recommendations. This upgrade will enhance the patient experience by providing valuable insights and additional features tailored to their preferences. Additionally, a chat feature will facilitate seamless communication between patients and healthcare professionals, enabling them to address any queries or concerns effectively. Moreover, the system will incorporate robust email notification capabilities, keeping patients informed about their appointments, updates, and any relevant information. Through these advanced features, the proposed Doctor Appointment system aims to revolutionize the appointment booking process, offering convenience, transparency, and enhanced functionality to individuals seeking healthcare services.

1.3 Features of the Proposed System

The features of the proposed system are:

- **Doctor Availability Management**
- **Patient Appointment Booking**
- **Secure Payment Gateway (Stripe**
- **Premium Upgrade Option**
- **Insights and Analytics**
- **Chat Feature**
- **Email Notifications**

2. FUNCTIONAL REQUIREMENTS

The functional requirements of the system are:

- ***Doctor Availability Management:*** Doctors can update their timeslots according to their schedules, ensuring efficient management of their availability within the system.
- ***Patient Appointment Booking:*** Patients have access to a user-friendly interface where they can browse available timeslots and book appointments with their preferred doctors, offering convenience and flexibility in scheduling healthcare appointments.
- ***Secure Payment Gateway (Stripe):*** Integration of a secure payment gateway powered by Stripe ensures smooth and reliable transactions for both patients and healthcare providers, enhancing the overall payment experience and security.
- ***Premium Upgrade Option:*** Patients can opt for a premium upgrade, unlocking exclusive insights such as popularly booked doctors and personalized recommendations, providing added value and enhanced user experience for those seeking additional features.
- ***Insights and Analytics:*** Premium users gain access to insights such as the most booked doctors, enabling them to make informed decisions about their healthcare provider choices and enhancing their overall experience within the system.
- ***Chat Feature:*** A chat feature facilitates seamless communication between patients and healthcare professionals, allowing them to address queries, concerns, or discuss appointments in real-time, fostering better engagement and support.
- ***Email Notifications:*** Robust email notification capabilities keep patients informed about their appointments, updates, and any relevant information, ensuring timely communication and reducing the risk of missed appointments or misunderstandings.
- ***Provide Prescriptions:*** The system allows healthcare professionals to provide prescriptions electronically to patients, facilitating seamless access to necessary medications and treatment plans.

- ***Review:*** Users can provide and access reviews to evaluate the quality of services provided by healthcare professionals, fostering transparency and informed decision-making within the system.

3. NON-FUNCTIONAL REQUIREMENTS

- **Performance:**

  The system should be able to handle a large volume of concurrent users without experiencing significant latency or downtime, ensuring smooth and uninterrupted streaming experiences.

- **Scalability:**

  The architecture should be designed to scale horizontally and vertically to accommodate increasing numbers of users and content without compromising performance.

- **Reliability:**

  The service should be highly reliable, with minimal downtime and robust failover mechanisms in place to ensure continuous availability.

- **Security:**

  The platform should implement stringent security measures to protect user data, including encryption of sensitive information, secure authentication mechanisms, and safeguards against unauthorized access and data breaches.

- **Usability:**

  The user interface should be intuitive and easy to navigate, catering to users of varying technical proficiency and ensuring a positive user experience.

- **Availability:**

  Since this application is a web-based application, it will be available to users 24x7 with the exception during server maintenance or upon site updates.

- **Interoperability:**

  The system should be designed to seamlessly integrate with external systems and services, such as payment gateways and content delivery networks, to provide a seamless user experience and optimize performance.

- **Maintainability:**

  The codebase and infrastructure should be well-organized and documented, facilitating ease of maintenance, updates, and enhancements over time.

4. UML DIAGRAMS

4.1 Use Cases

- **Use Case for MediCare SignUp**

Use case name:	Sign Up
Created by:	User
Date created:	25-01-2024
Description:	Allows Guest users to create a new account on Medicare.
Primary actor:	Guest User
Secondary actor:	None
Precondition:	The user is on the Medicare homepage and does not have an existing account.
Postcondition:	The user successfully creates a new Medicare account.
Main flow:	1. The user selects the "Sign Up" option. 2. The system prompts for necessary information (username,email,password,usertype,gender,image). 3. The user provides the required information and submits the registration form. 4. The system creates a new user account.

	5. The use case ends.

- **Use Case for Medicare Log In**

Use case name:	Log In
Created by:	User
Date created:	25-01-2024
Description:	Allows Guest users to log in to existing account on MediCare.
Primary actor:	Guest User
Secondary actor:	None
Precondition:	The user is on the Medicare homepage and does have an existing account.
Postcondition:	The user successfully logs in to his Medicare account.
Main flow:	1. The user selects the "Logs in " option. 2. The system prompts for necessary information (email, password). 3. The user provides the required information and submits the registration form. 4. The user enters into his account.

- **Use Case for Doctor Appointment**

Use case name:	Book Appointment
Created by:	User
Date created:	25-01-2024
Description:	Enables registered users to schedule appointments with doctors based on their preferred time slots from the available options.
Primary actor:	User (Registered User)
Secondary actor:	None
Precondition:	The user is on the Medicare homepage.
Postcondition:	The user successfully logs in to his Medicare account.
Main flow:	1. The user selects the "Doctors" option. 2. The system retrieves and displays a list of available doctors. 3. The user explores the doctor. 4. Select the booking date. 5. Select the booking time. 6. Redirect to payment(Stripe). 7. Complete the payment. 8. Use case ends.

- **Use Case for Upgrade to Premium**

Use case name:	Upgrade to premium
Created by:	User
Date created:	25-01-2024
Description:	Allows Registered User to upgrade their accounts to premium
Primary actor:	User (Registered User)
Secondary actor:	None
Precondition:	The user is on the Medicare homepage.
Postcondition:	The user will be upgraded to premium Account.
Main flow:	1. The user navigates to the user dashboard. 2. They click on the "Upgrade to Premium" option. 3. They proceed to complete the payment process. 4. The use case ends.

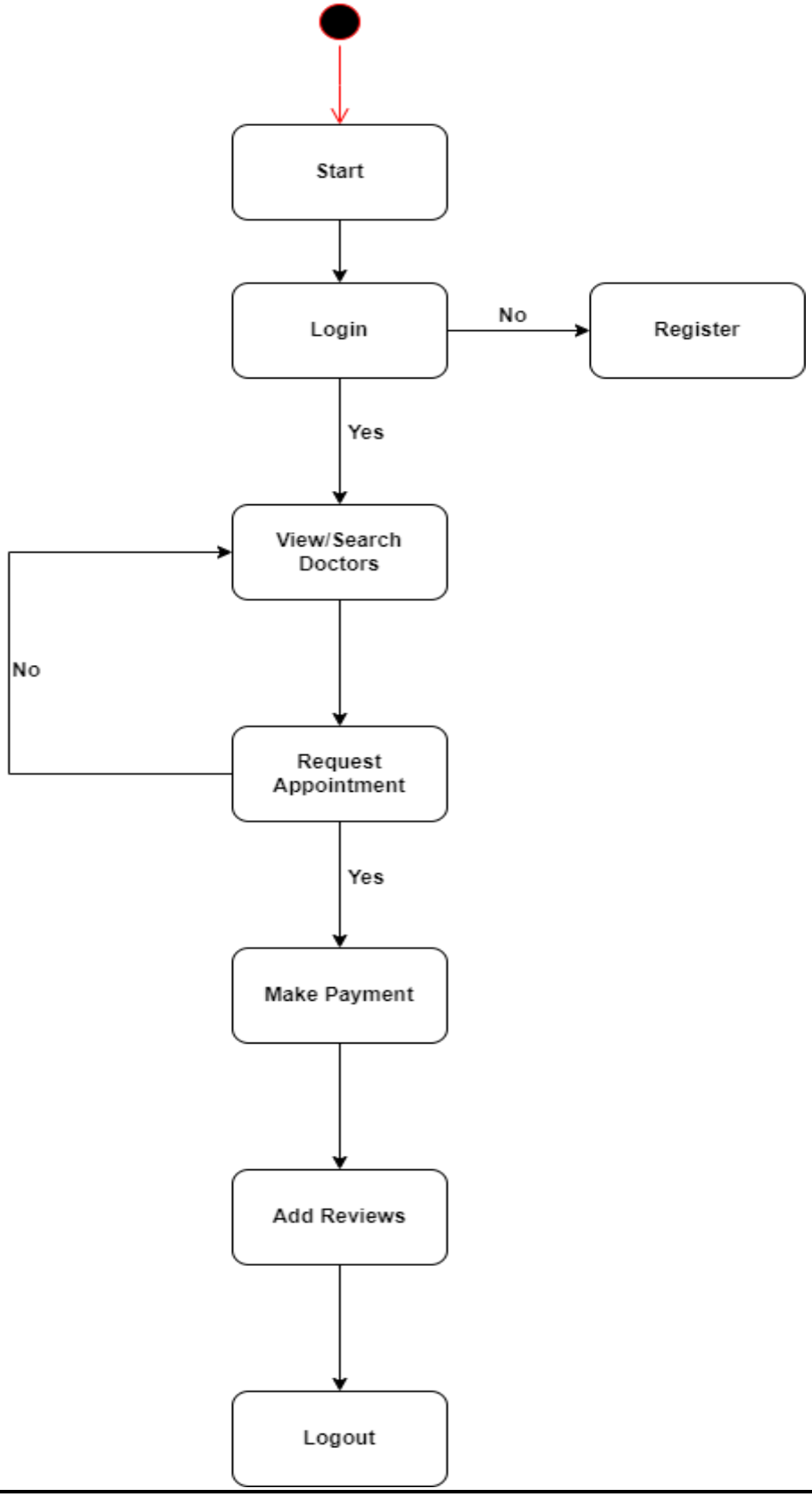
- **Use Case for Verifying Doctors**

Use case name:	Verify Doctors
Created by:	User
Date created:	25-01-2024
Description:	Verifying Registered Doctors to verify their account.
Primary actor:	Medicare Admin
Secondary actor:	None
Precondition:	The admin is logged in and is on their dashboard.
Postcondition:	The admin verifies the newly registered doctor.
Main flow:	1. The admin selects the "Verify Doctors" option. 2. The system displays options to view their account details 3. The admin makes the desired changes and saves. 4. The system updates the doctor's account settings. 5. The use case ends.

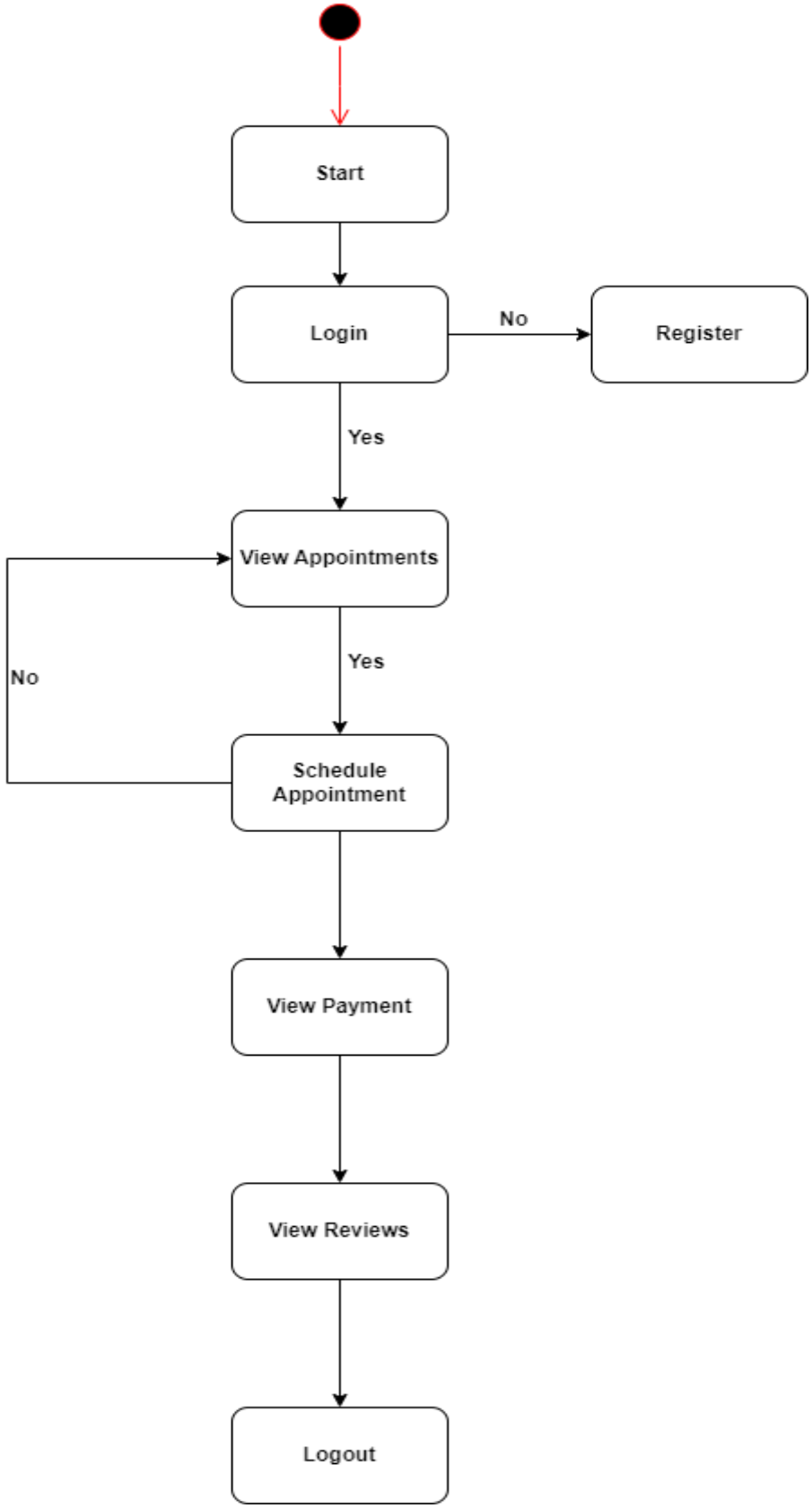
4.2 Use Cases Diagram

4.3 Activity Diagram

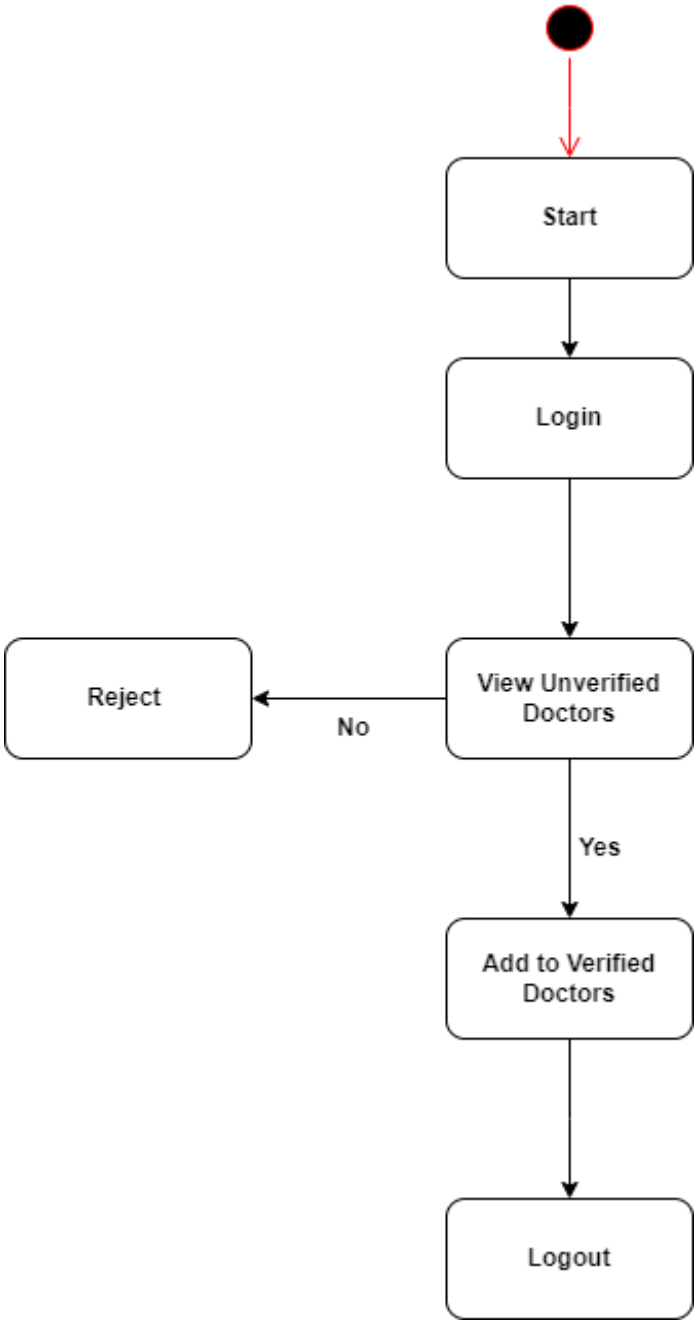
4.3.1 User



4.3.2 Doctor


Start
Login
No
Register
Yes
View Appointments
Yes
No
Schedule Appointment
View Payment
View Reviews
Logout

4.3.3 ADMIN


Start
Login
View Unverified Doctors
Reject
No
Yes
Add to Verified Doctors
Logout

5. DATABASE DESIGN

- **Table:** user

```
const UserSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  name: { type: String, required: true },
  phone: { type: Number },
  photo: { type: String },
  role: {
    type: String,
    enum: ["patient", "admin"],
    default: "patient",
  },
  gender: { type: String, enum: ["male", "female", "other"] },
  bloodType: { type: String },
  appointments: [{ type: mongoose.Types.ObjectId, ref: "Appointment" }],
  resetPasswordToken: { type: String },
  resetPasswordExpires: { type: Date },
  isPremiumUser: { type: Boolean, default: false },
  emailVerified: { type: Boolean, default: false },
});
```

- **Table:** doctor

```
const DoctorSchema = new mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  name: { type: String, required: true },
  phone: { type: Number },
  photo: { type: String },
  ticketPrice: { type: Number },
  role: {
    type: String,
  },
  specialization: { type: String },
  qualifications: {
    type: Array,
  },
  experiences: {
    type: Array,
  },
  emailVerified: { type: Boolean, default: false },
  bio: { type: String, maxLength: 50 },
  about: { type: String },
  timeSlots: { type: Array },
  reviews: [{ type: mongoose.Types.ObjectId, ref: "Review" }],
  averageRating: {
    type: Number,
    default: 0,
  },
  totalRating: {
    type: Number,
    default: 0,
  },
  isApproved: {
    type: String,
    enum: ["pending", "approved", "cancelled"],
    default: "pending",
  },
  appointments: [{ type: mongoose.Types.ObjectId, ref: "Appointment" }],
});
```

- **Table:** booking

```
const bookingSchema = new mongoose.Schema(
  {
    doctor: {
      type: mongoose.Types.ObjectId,
      ref: "Doctor",
      required: true,
    },
    user: {
      type: mongoose.Types.ObjectId,
      ref: "User",
      required: true,
    },
    ticketPrice: { type: String, required: true },
    status: {
      type: String,
      enum: ["pending", "approved", "cancelled"],
      default: "pending",
    },
    isPaid: {
      type: Boolean,
      default: true,
    },
    appointmentDate: { type: Date, required: true },
    appointmentTime: { type: String, required: true },
  },
  { timestamps: true }
);
```

- **Table:** Review

```
const reviewSchema = new mongoose.Schema(
  {
    doctor: {
      type: mongoose.Types.ObjectId,
      ref: "Doctor",
    },
    user: {
      type: mongoose.Types.ObjectId,
      ref: "User",
    },
    reviewText: {
      type: String,
      required: true,
    },
    rating: {
      type: Number,
      required: true,
      min: 0,
      max: 5,
      default: 0,
    },
  },
  { timestamps: true }
);
```

- **Table:** chat

```
const MessageSchema = new mongoose.Schema(
  {
    sender: {
      type: mongoose.Types.ObjectId,
      required: true,
      refPath: "senderType",
    },
    senderType: {
      type: String,
      enum: ["Doctor", "User"],
      required: true,
    },
    receiver: {
      type: mongoose.Types.ObjectId,
      required: true,
      refPath: "receiverType",
    },
    receiverType: {
      type: String,
      enum: ["Doctor", "User"],
      required: true,
    },
    message: {
      type: String,
      required: true,
    },
    read: {
      type: Boolean,
      default: false,
    },
  },
  { timestamps: true }
);
```

- **Table:** prescription

```
const prescriptionSchema = new mongoose.Schema({
    patient: { type: mongoose.Types.ObjectId, ref: "Patient" },
    doctor: { type: mongoose.Types.ObjectId, ref: "Doctor" },
    prescriptions: [{
        medicine: { type: String, required: true },
        noofdays: { type: String },
        dosage: { type: String },
        frequency: { type: String }
    }]
});
```

6. TEST CASES

6.1 Test Case for Sign Up

Sr. No.	Test Case	Feature Description	Steps to Execute	Expected Results
1	TC-REG-001 Invalid Username	Registration Form	1. Enter an invalid username format. 2. Provide valid email and password. 3. Submit registration.	User should receive an error message indicating that the username format is invalid.
2	TC-REG-002 Weak Password	Registration Form	1. Provide valid username and email. 2. Enter a weak password. 3. Submit registration.	User should receive an error message indicating that the password is too weak.
3	TC-REG-003 Missing Email	Registration Form	1. Enter valid username and password. 2. Leave the email field blank. 3. Submit registration.	User should receive an error message indicating that the email field is required.
4	TC-REG-004 Existing Email	Registration Form	1. Provide valid username and password. 2. Enter an email that is already registered. 3. Submit registration.	User should receive an error message indicating that the email is already in use.
5	TC-REG-005 Successful Reg.	Registration Form	1. Provide valid username, email, and password. 2. Submit registration.	User should be successfully registered and redirected to the home page or login screen.
6	TC-REG-006 Password Match	Registration Form	1. Provide valid username, email, and password. 2. Confirm password with a mismatch. 3. Submit registration.	User should receive an error message indicating that the passwords do not match.

6.2 Test Case for Log In

Sr. No.	Test Case	Feature Description	Steps to Execute	Expected Results
1	TC-LOGIN-001 Missing Fields	Login Form	1. Attempt to log in with blank username and password. 2. Click on Login button.	User should receive an error message indicating that both username and password fields are required.
2	TC-LOGIN-002 Invalid Email	Login Form	1. Enter an invalid email format. 2. Provide valid password. 3. Click on Login button.	User should receive an error message indicating that the email format is invalid.
3	TC-LOGIN-003 Incorrect Pass	Login Form	1. Enter valid username or email. 2. Provide an incorrect password. 3. Click on Login button.	User should receive an error message indicating that the password is incorrect.
4	TC-LOGIN-004 Successful Log	Login Form	1. Enter valid username or email. 2. Provide correct password. 3. Click on Login button.	User should be successfully logged in and redirected to the home page or user dashboard.
2	TC-LOGIN-002 Invalid Email	Login Form	1. Enter an invalid email format. 2. Provide valid password. 3. Click on Login button.	User should receive an error message indicating that the email format is invalid.
6	TC-LOGIN-006 Forgot Password	Login Form	1. Click on the "Forgot Password" link. 2. Follow the password recovery process.	User should be able to recover their password through a secure and user-friendly process.

7. INPUT DESIGN AND OUTPUT DESIGN

7.1 Input Design for Medicare Project:

1. **Registration Form:**
    - Description: This form allows users to create a new account on Medicare by providing their details.
    - Fields: User Name, Email, Password, Photo , Gender , UserType
    - Validation: Ensure unique username and valid email format. Password should meet complexity requirements.
    - Messages: Provide clear instructions and error messages for data entry.
2. **Login Form:**
    - Description: Form used for user login, accessible by both students and admins.
    - Fields: Username, Password
    - Validation: Verify credentials against database records.
    - Messages: Display appropriate messages for successful or failed login attempts.
3. **Search and Filtering Options:**
    - Description: Enable users to easily find doctors based on preferences.
    - Input Elements: Search bar
    - Interaction: Allow users to input search queries and select filters.
    - Validation: Ensure filter selections are valid options.
4. **User Profile Form:**
    - Description: Form for users to manage their profile details.
    - Fields: Name, Email, Password, Profile Picture
    - Interaction: Allow users to update their profile information.
    - Validation: Validate email format and password strength.

7.2 Output Design for Medicare Project:

1. **Doctors Section:**
    - Description: This section displays doctors based on their reviews and average ratings, allowing users to make informed decisions about their healthcare provider.
    - Format: Users can view doctors in either thumbnail or list view, offering flexibility in how they browse through available healthcare professionals.
    - Content: Detailed information is provided about each doctor, including their educational background, professional experiences, and available rating slots, ensuring transparency and aiding users in selecting the most suitable healthcare provider.
2. **ChatView Section:**
    - Description: The chat view facilitates real-time communication between patients and healthcare professionals, ensuring efficient interaction and support.
    - Format: Conversational messaging interface with options for attachments, emojis, and message status indicators.
3. **User Dashboard:**
    - Description: Overview of user account information and activity.
    - Content: Display user profile details, booking history, account settings, and subscription status.
4. **Doctor Dashboard:**
    - Description: Overview of doctor account information and activity.
    - Content: Display user profile details, booking history, account settings, and new bookings.

5. **Admin Dashboard:**
    - Description: Interface for administrators to manage system operations and user accounts.
    - Content: Overview of user statistics, content management tools, and system settings.

8. FUTURE ENHANCEMENTS

1. Smart Prescription Management System: Develop an intelligent prescription management system that leverages machine learning algorithms to suggest personalized medication regimens based on patient history, allergies, and drug interactions. This system could also automatically generate electronic prescriptions and facilitate seamless integration with pharmacies for prescription fulfillment, enhancing medication adherence and patient safety.

2. Advanced Teleconsultation Features: Enhance the teleconsultation capabilities by introducing features such as virtual waiting rooms, screen sharing for medical images or test results, and collaborative whiteboards for explaining treatment plans visually. Additionally, integrating remote monitoring devices for vital signs during teleconsultations would enable healthcare professionals to conduct more comprehensive assessments and provide higher quality care remotely.

3. Patient Health Record Portability: Implement a standardized patient health record format compatible with international standards such as HL7 FHIR (Fast Healthcare Interoperability Resources) to enable seamless sharing of patient data across healthcare systems and institutions. This enhancement would empower patients to have greater control over their health information, improve care coordination among healthcare providers, and facilitate continuity of care across different healthcare settings.

4. Predictive Analytics for Disease Prevention: Integrate predictive analytics capabilities into the platform to identify at-risk patient populations and proactively recommend preventive interventions or screenings based on demographic, lifestyle, and genetic factors. By leveraging predictive modeling techniques, healthcare providers can anticipate and address potential health issues before they escalate, leading to early detection, prevention, and better management of chronic conditions.

5. Virtual Health Assistant with Natural Language Processing (NLP): Develop a virtual health assistant powered by natural language processing (NLP) technology to provide personalized health recommendations, answer patient queries, and assist with appointment scheduling or medication reminders through voice or text interactions. This AI-driven assistant could also offer tailored health education resources and behavioral coaching to empower patients in managing their health and wellness effectively.

9. CONCLUSION

In conclusion, the Doctor Appointment system embodies a transformative approach to healthcare management, ushering in an era of efficiency, accessibility, and personalized care. By harnessing the power of technology, this platform not only simplifies the process of scheduling appointments but also fosters meaningful connections between patients and healthcare providers. The system's intuitive interface empowers users to navigate effortlessly through available doctors, review comprehensive profiles detailing practitioners' education and experiences, and make informed decisions based on peer reviews and ratings. Such transparency not only instills confidence in patients but also cultivates trust in the healthcare ecosystem, laying the foundation for a collaborative and patient-centric approach to healthcare delivery.

Furthermore, the integration of telemedicine capabilities within the platform marks a significant leap forward in expanding access to healthcare services beyond traditional brick-and-mortar settings. Through virtual consultations, patients can seek medical advice from the comfort of their homes, eliminating barriers such as geographical constraints and long waiting times. This not only enhances convenience for patients but also optimizes resource utilization for healthcare providers, ultimately leading to improved patient outcomes and enhanced overall efficiency within the healthcare system. Moreover, the potential for future enhancements, such as AI-driven predictive analytics and virtual health assistants, holds promise for revolutionizing healthcare delivery by enabling proactive disease prevention, personalized health recommendations, and enhanced patient engagement.

In essence, the Doctor Appointment system represents more than just a scheduling tool; it embodies a paradigm shift towards a more interconnected, proactive, and patient-centric approach to healthcare. By leveraging technology to facilitate seamless communication, empower patients, and drive innovation, the platform has the potential to redefine the healthcare experience, ensuring that every individual has access to quality care when and where they need it most.

10. REFERENCES

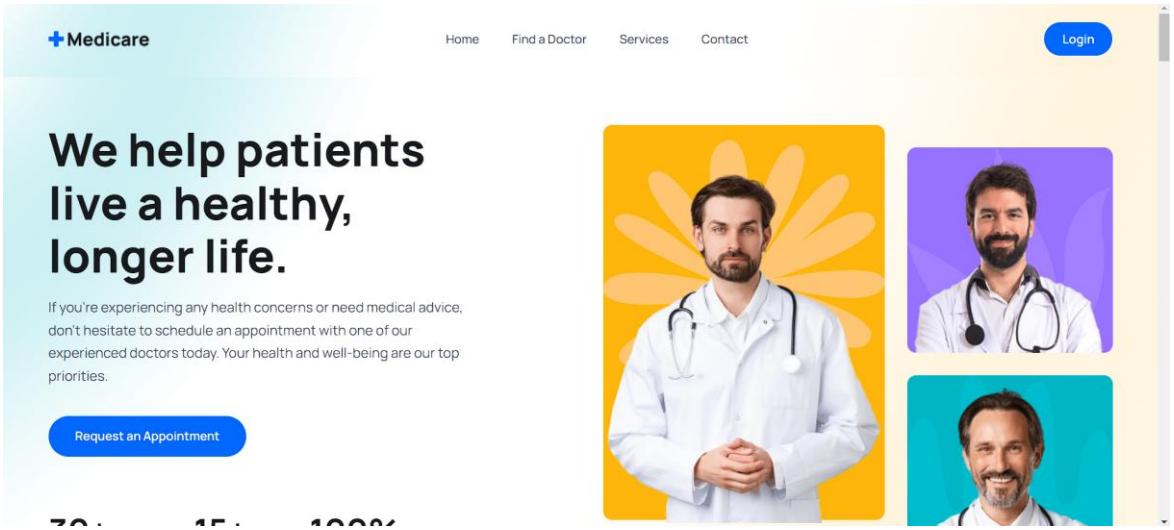
Bibliography

1. https://github.com/animeshthomas/Doccure-MERN
2. https://tailwindcss.com/docs/installation
3. https://nodejs.org/docs/latest/api/
4. https://www.youtube.com/@CodingWithMuhib
5. https://devfolio.co/projects/medicare-0d1c
6. https://cloud.mongodb.com/v2#/org/65a81550aba554066476d2d8/projects
7. https://react.dev/learn/installation

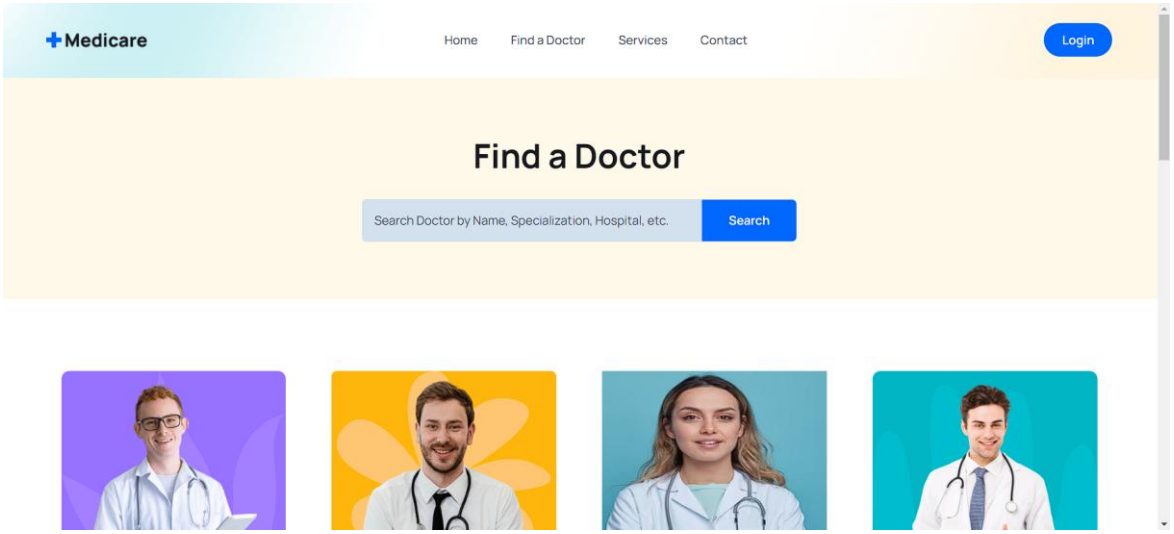
11. Annexure

A. Screenshots

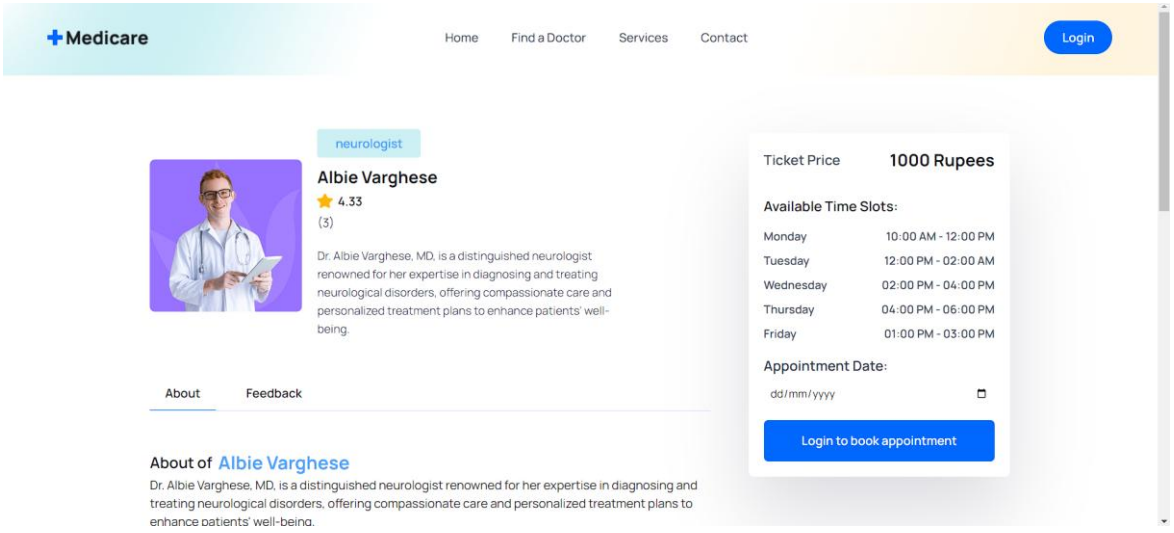
Home Page



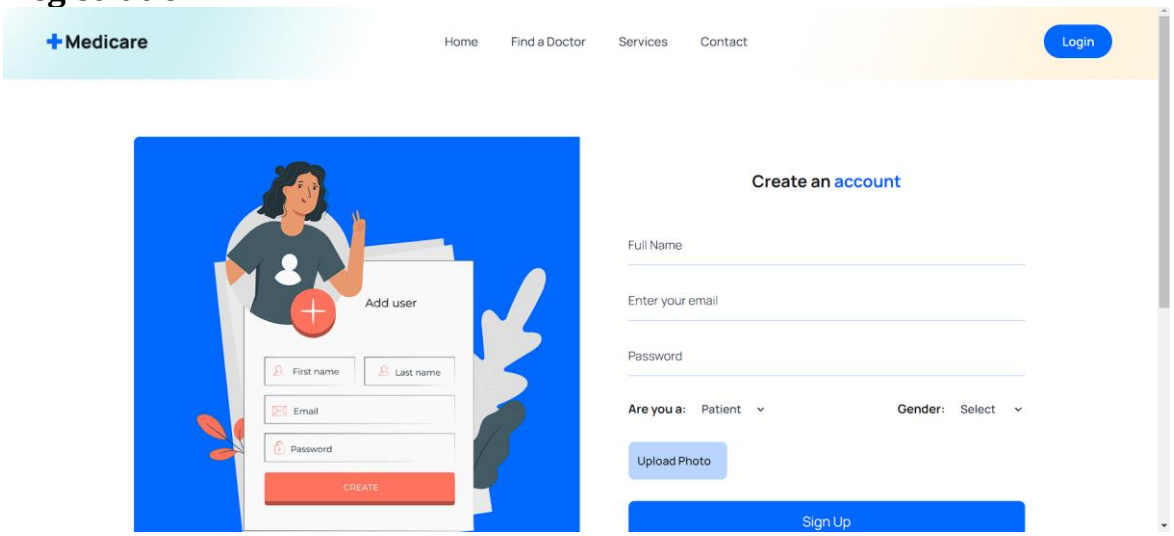
Find a Doctor



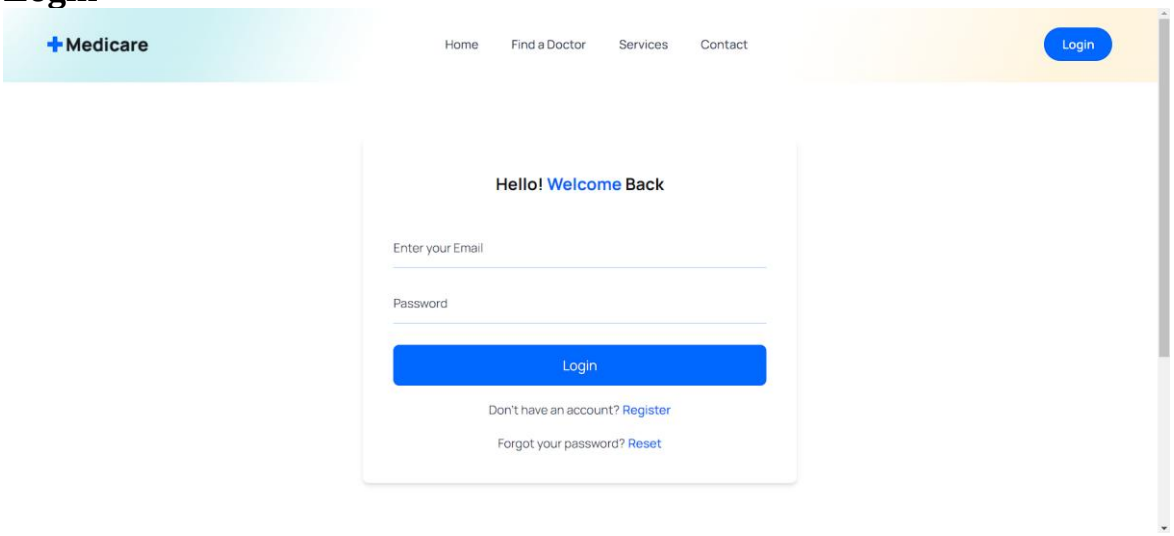
Doctor Details



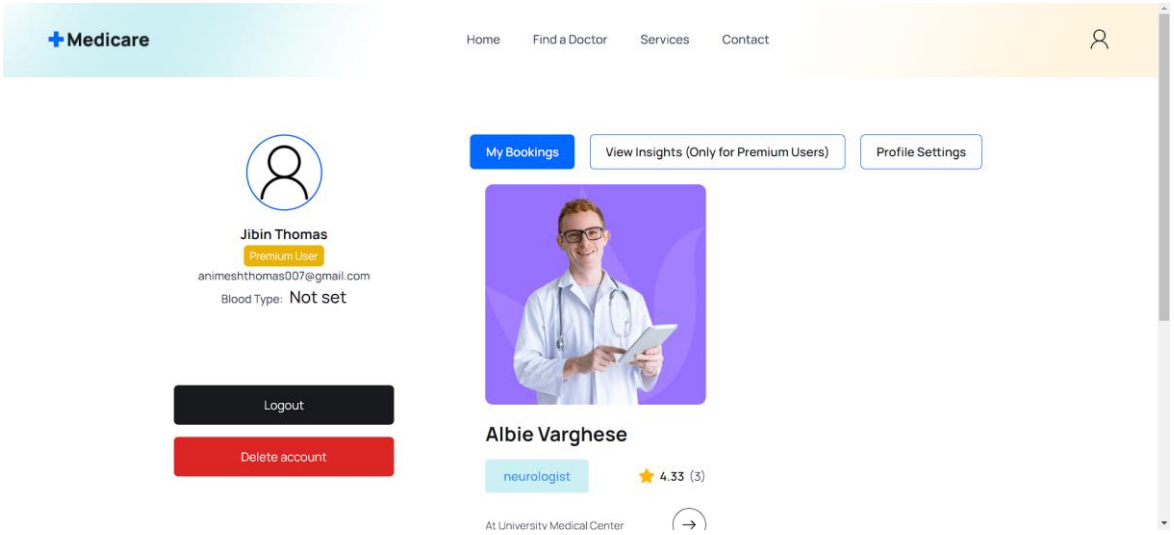
Registration



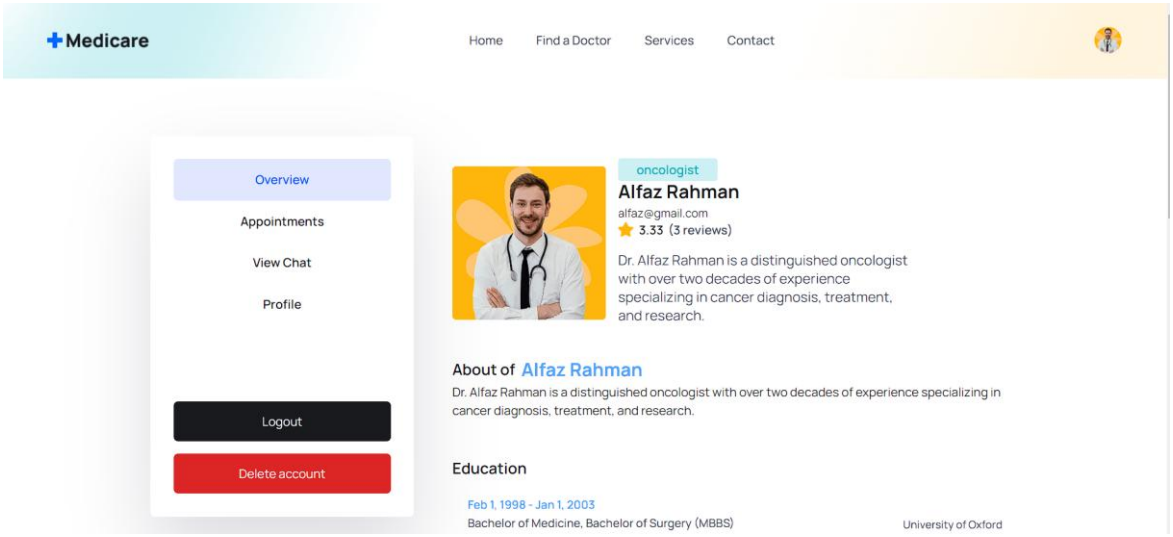
Login



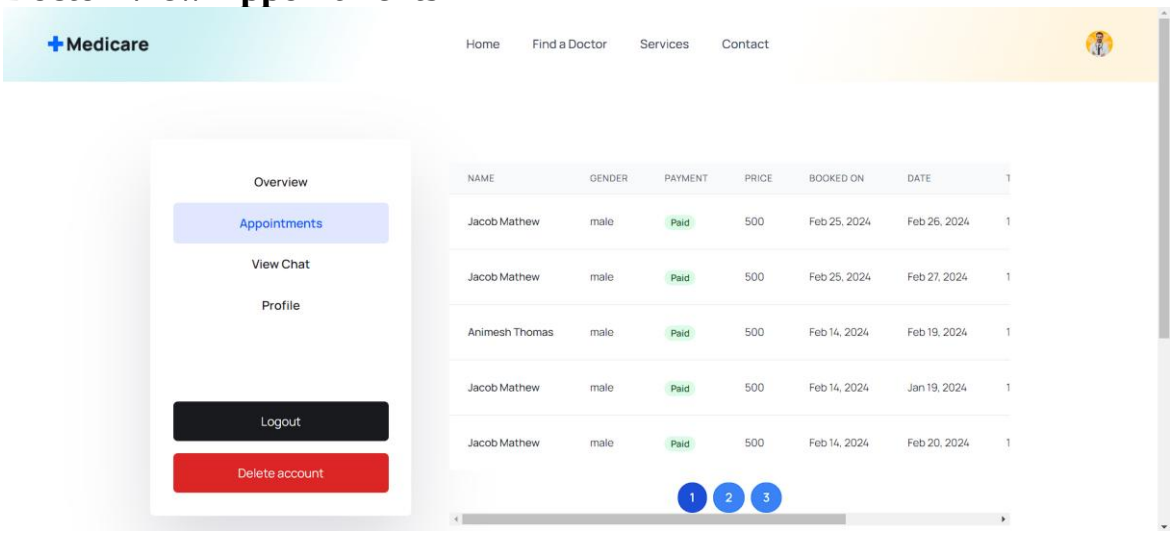
User Dashboard



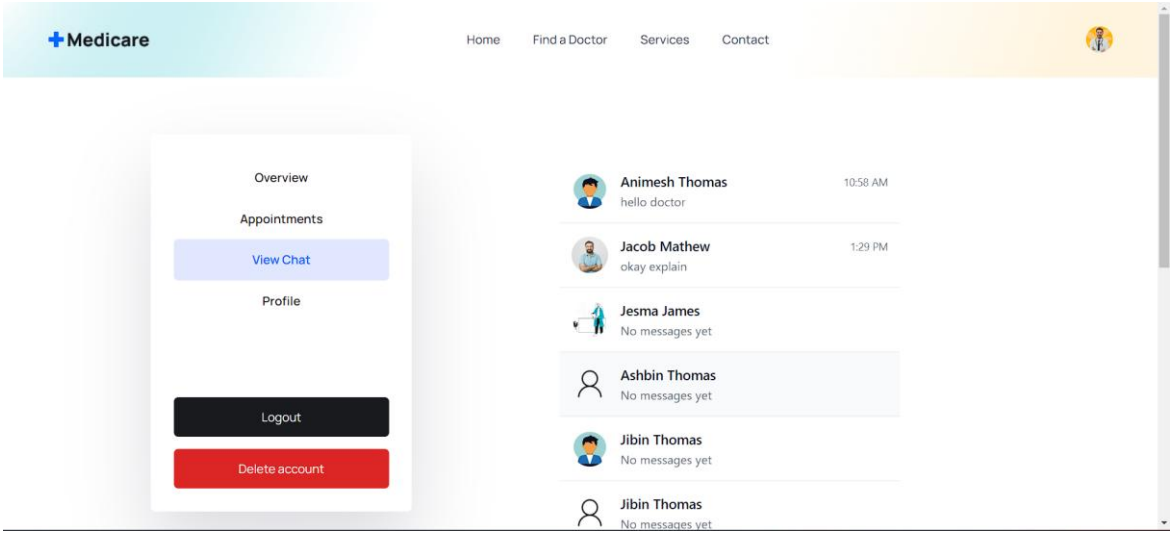
Doctor Dashboard



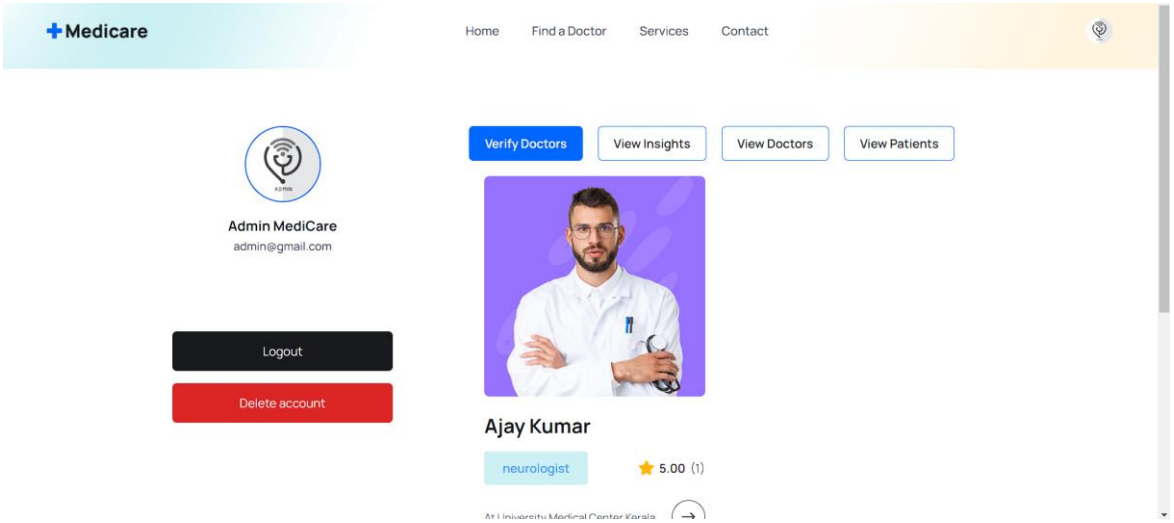
Doctor View Appointments



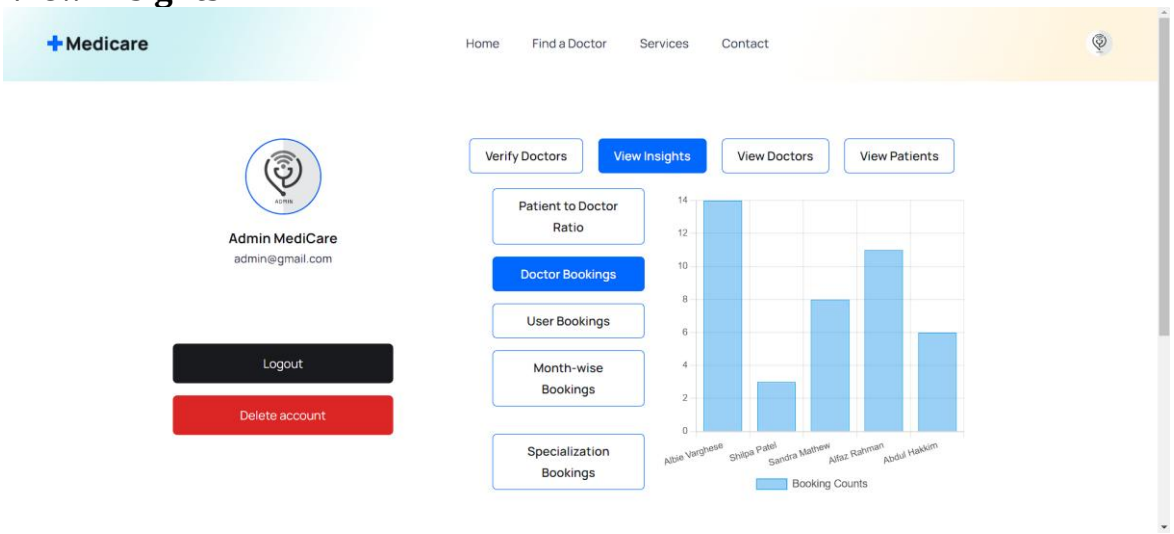
View Chat



Admin Dashboard



View Insights



View Doctors