**Home Care Services for Seniors**

**Problem Statement:**

As the global population continues to age, there is an increasing need for healthcare solutions that cater specifically to the unique needs of senior citizens. Many seniors prefer to receive care in the familiar environment of their homes, presenting a challenge in coordinating and delivering a broad range of home care services. The existing healthcare infrastructure often lacks a dedicated platform to seamlessly connect seniors, their caregivers, and qualified healthcare providers.

This project aims to address these challenges by creating a user-friendly and secure web-based application. The lack of a centralized system often results in inefficiencies, miscommunications, and difficulties in matching seniors with suitable healthcare providers. Additionally, the absence of a dedicated platform for appointment scheduling, real-time communication, and service coordination further compounds the challenges faced by seniors and their caregivers.

**Users of the System:**

**Role:**

1. Admin

2. User

Admin should have permission for CRUD operation.

**Role based Menu Options**

Admin:

* Register
* Login
* Post Service details
* Edit Service Details
* Delete Service Details
* View All Services
* View All Bookings
* Change Status of Booking
* View All Payment
* Logout

User:

* Register
* Login
* View All Services
* Book for a service
* Check the Status of Booking
* Make Payment for the Service
* Delete Booking
* Logout

**Functional Requirements:**

1.     **Service Offerings**: The application offers a variety of home care services, including but not limited to skilled nursing, personal care, medication management, physical therapy, and companionship with payment details

2.     **User Authentication**: Users, including seniors and their families or caregivers, are required to create accounts or log in to access the platform. This ensures data security and user privacy.

3.     **Appointment booking Scheduling**: Senior individuals or their caregivers can conveniently schedule home care visits or services through the platform. They can specify the type of service needed, preferred date and time, and any specific requirements.

4.     For each service they need to do the payment. User can pay for the service and also view the payment history

5. **Real-time Communication:** The platform facilitates real-time communication between seniors, their caregivers, and healthcare providers for updates, queries, and emergencies. The status of booking will be reflected in the portal

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

* Have appropriate filters for search.
* Email integration for intimate users about the service booking.
* Multi-factor authentication for the sign-in process

**Non-Functional Requirements:**

1. Security: The system must implement robust security measures to protect user data, including user authentication, secure data storage, and encrypted data transmission
2. Scalability: The system should be designed to handle an increasing number of bookings, services, payments, and users.
3. Usability: The user interface should be intuitive and user-friendly, with responsive design for mobile and desktop users.
4. Availability: The system should be available 24/7 with minimal downtime for maintenance.
5. Logging and Auditing: Support logging and auditing of system activities for monitoring and troubleshooting.

**Application Flow:**

 Admin Side:

The admin can change the status of booking such as "pending", "completed ", "rejected".

The admin can view all the users account and view all the payment history.

**Create a data flow diagram if student has logged in.**

User Side:

* The application flow for the portal begins with user registration, where prospective users create accounts by providing personal information. Upon logging in, users access the list of services offered.
* The user can book for the services listed.
* The user will do the payment for the service and can view the payment history.

**Create a data flow diagram if user has logged in.**

**Abstract:**

The Home Care Services for Senior’s project is a web-based application developed on the .Net framework, specifically designed to cater to the distinctive healthcare requirements of senior citizens. Focused on delivering comprehensive and personalized care, this platform empowers seniors to receive a variety of home-based healthcare services, ensuring comfort, convenience, and an enhanced quality of life and do the payment online.

**Modules of the Application:**

**ADMIN:**

Ø Register

Ø Login

Ø Dashboard

* Service details
* Booking details

§ Change the booking status

§ Payment details

**USER**:

Ø Register

Ø Login

Ø Dashboard

* Service Details List
* Payment History

Ø Booking status page.

Ø Make Payment

**Technology Stack**

**>Front End**

React , HTML, CSS

**> Back End**

Java, Spring boot, MySQL for database

**> Authentication**

JWT for User Authentication

**Application assumptions**:

1.  The login page should be the first page rendered when the application loads.

2.   Unless logged into the system, the user cannot navigate to any other pages.

3.   Logging out must again redirect to the login page.

4.   Design forgot password and forgot email buttons in login page.

**Validation**

**Client-Side Validation:**

Implement client-side validation using HTML5 attributes and JavaScript to validate user input before making API requests.

Provide immediate feedback to users for invalid input, such as displaying error messages near the input fields.

1. Basic email validation should be performed.

      2. Basic mobile number validation should be performed.

      3. Basic password should be performed

**Server-Side Validation:**

Implement server-side validation in the controllers to ensure data integrity.

Validate user input and API responses to prevent unexpected or malicious data from affecting the application.

Return appropriate validation error messages to the user interface for any validation failures.

**Exception Handling**

Implement exception handling mechanisms in the controllers to gracefully handle errors and exceptions.

Define custom exception classes for different error scenarios, such as API communication errors or database errors.

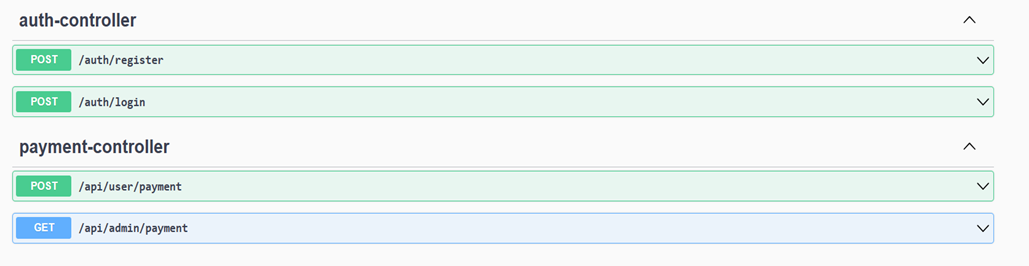
Log exceptions for debugging purposes while presenting user-friendly error messages to users. Record all the exceptions and errors handled store in separate table “**ErrorLogs**”.

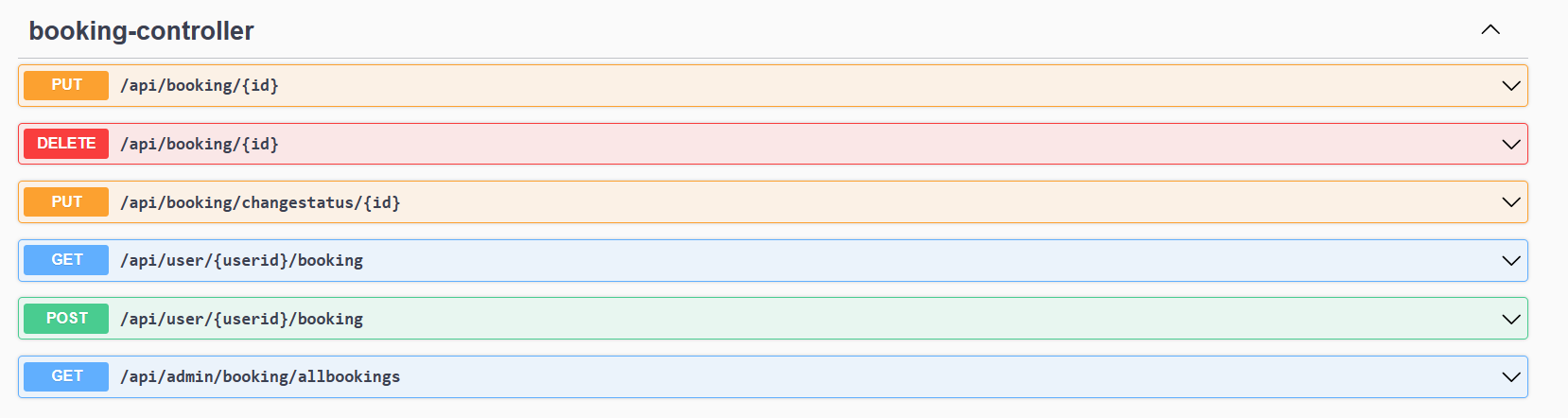
**Error Pages:**

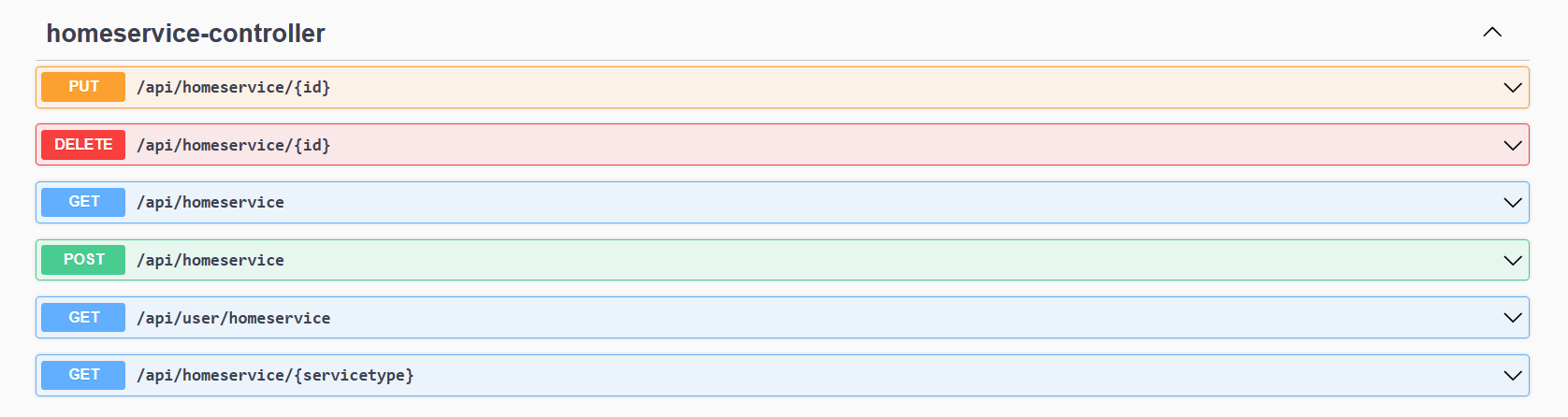
* Create custom error pages for different HTTP status codes (e.g., **404** Not Found, **500** Internal Server Error) to provide a consistent and user-friendly error experience.
* Ensure that error pages contain helpful information and guidance for users.
* Thus, create a reliable and user-friendly web application that not only meets user expectations but also provides a robust and secure experience, even when faced with unexpected situations.

**Project Tasks:**

API Endpoints:







**Backend Model:** Home Care Services for Seniors

class User

{

Long userId

String email

String password

String mobileNumber

String username

String userRole

}

class Customer

{

Long customerId;

String customerName;

String customerMobileNumber;

@OneToOne

User user;

@OneToMany

List<Booking> bookings;

@ManyToMany

List<HomeCareService> homeCareServices;

}

class HomeCareService

{

Long serviceID

String serviceType

String description

Double charges

String timing

@OneToMany

List<Booking> bookings

@ManyToMany

List<Customer> customers;

}

class Booking

{

Long bookingID

DateTime bookingDate

String description

Integer bookingStatus

@ManyToOne

HomeCareService homeCareServices;

@ManyToOne

Customer customer;

@OneToOne

Payment payment

}

class Payment

{

paymentId: long

status: String

totalAmount: Double

paymentDate: Date

modeOfPayment: String

@OneToOne

Booking booking

}

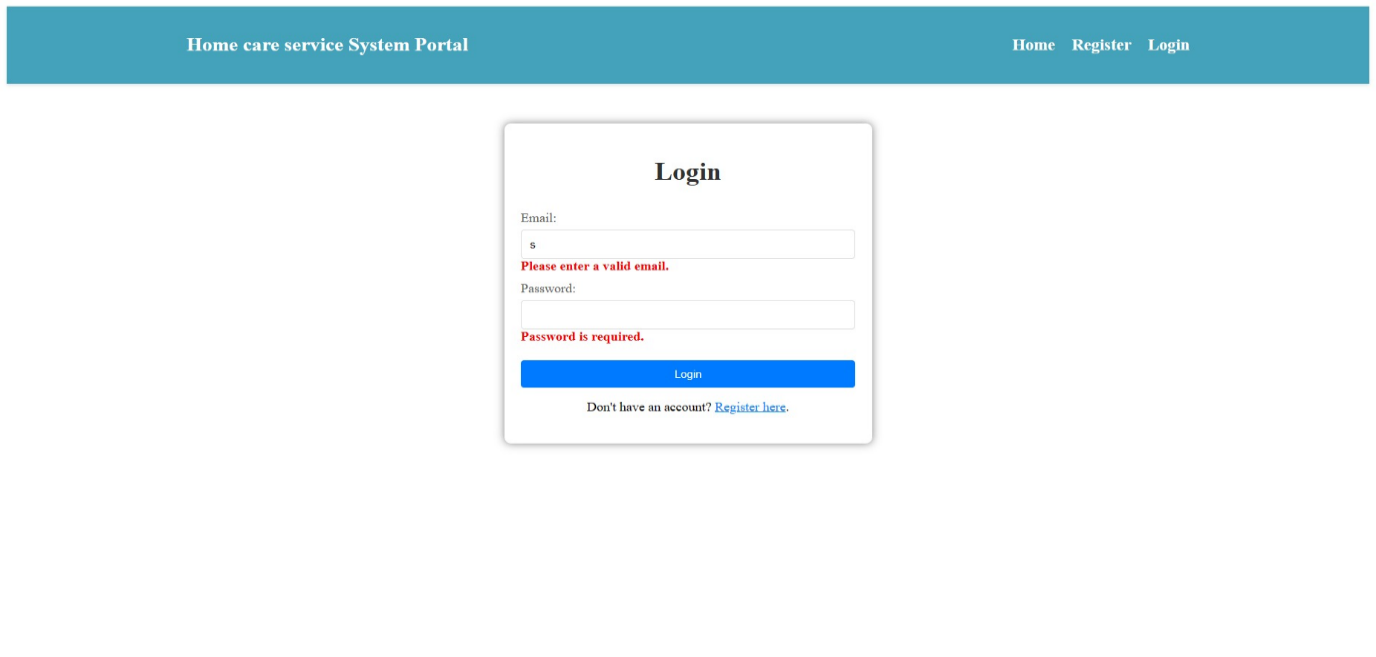
Note:

* Import model files, services and components as required.
* Create a required folder component inside source file.

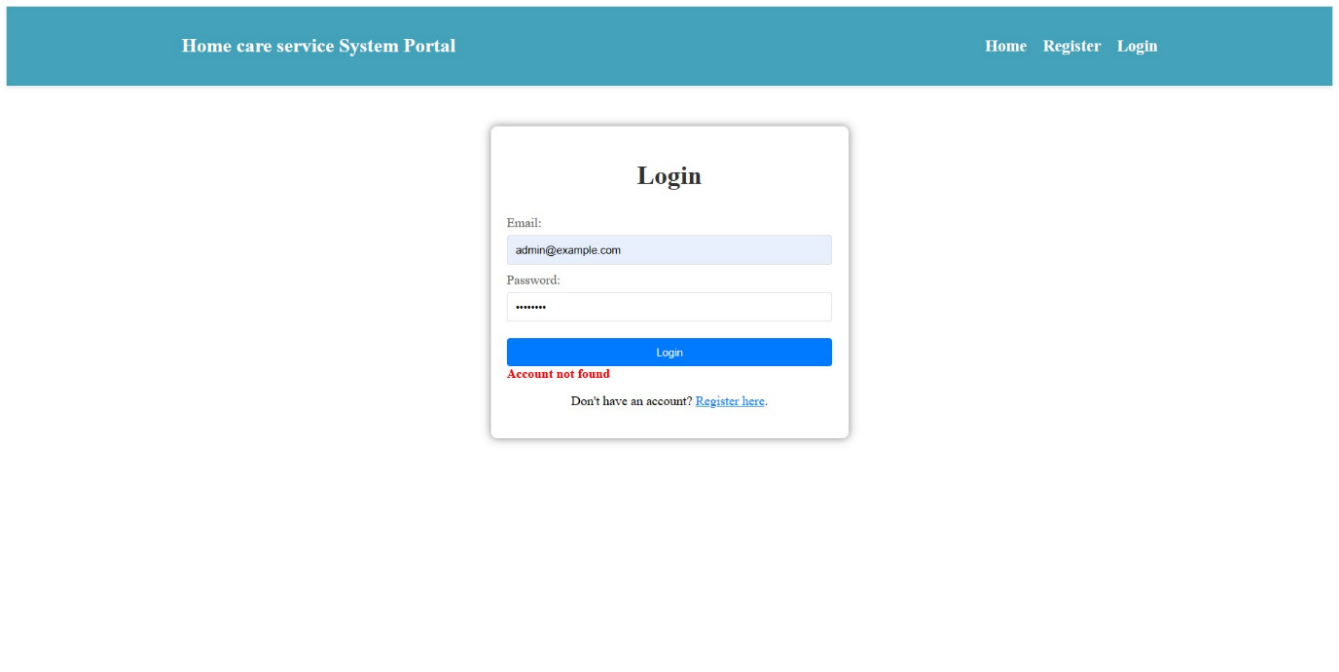
**Frontend Sample Screenshots:**

ADMIN SIDE:

Login Page (Error message should Display)



Error message "Account not found" need to display if wrong email/password given:



Error message should display on registration page when field is touched

**User Registration Form Instruction Note:**

Class Names and IDs Used:

Container:

> Class Name: container

Form Elements:

>Username Field:

>ID: username

>Class Name (Validation): text-danger

Email Field:

>ID: email

>Class Name (Validation - Required): text-danger

>Class Name (Validation - Pattern): text-danger

Password Field:

I>D: password

>Class Name (Validation - Required): text-danger

>Class Name (Validation - Minlength): text-danger

Confirm Password Field:

>ID: confirmPassword

>Class Name (Validation): text-danger

Mobile Number Field:

>ID: mobileNumber

>Class Name (Validation - Required): text-danger

>Class Name (Validation - Min/Max length): text-danger

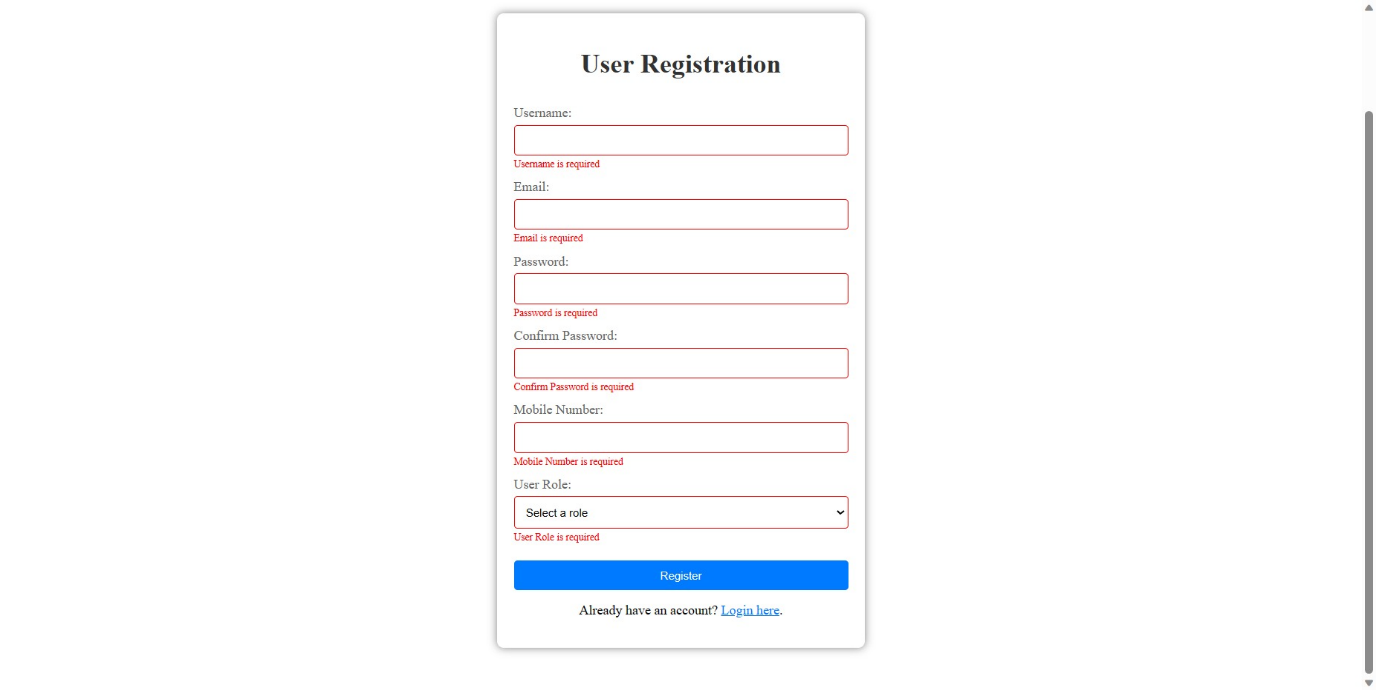
User Role Field:

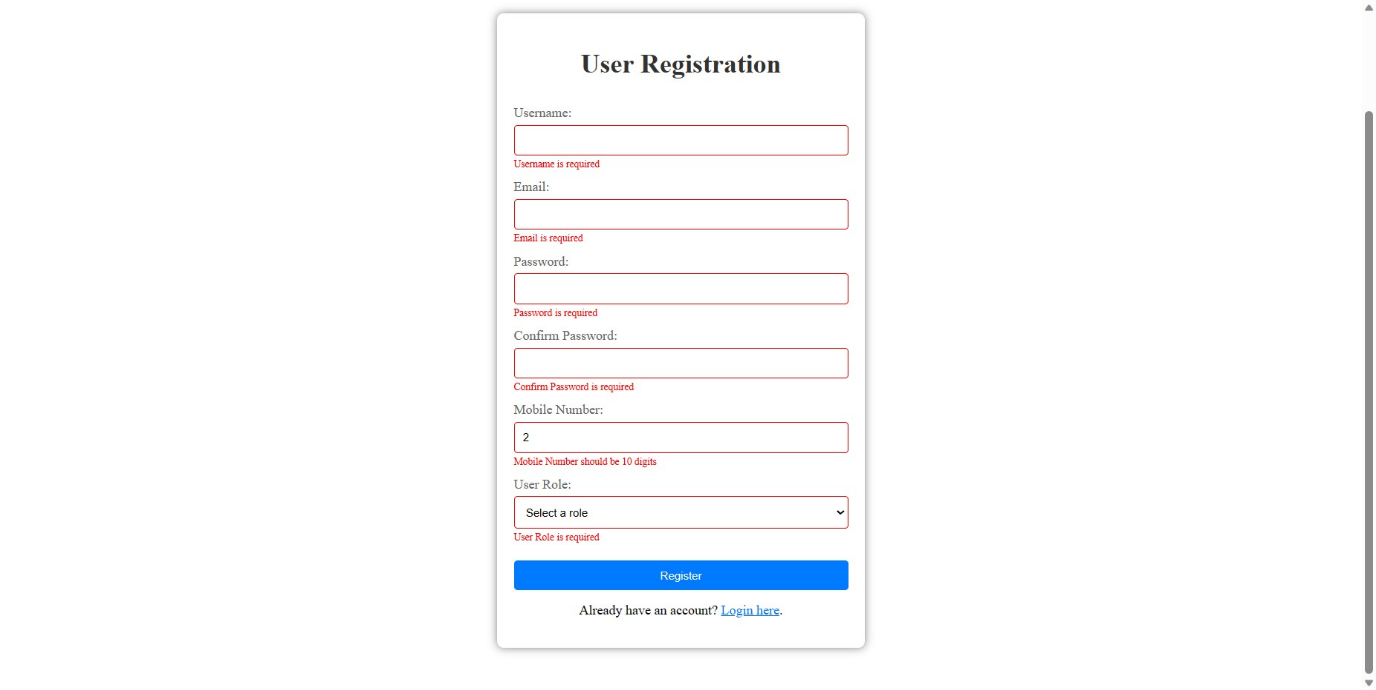
>ID: userRole

>Class Name (Validation): text-danger

Validation Indicators:

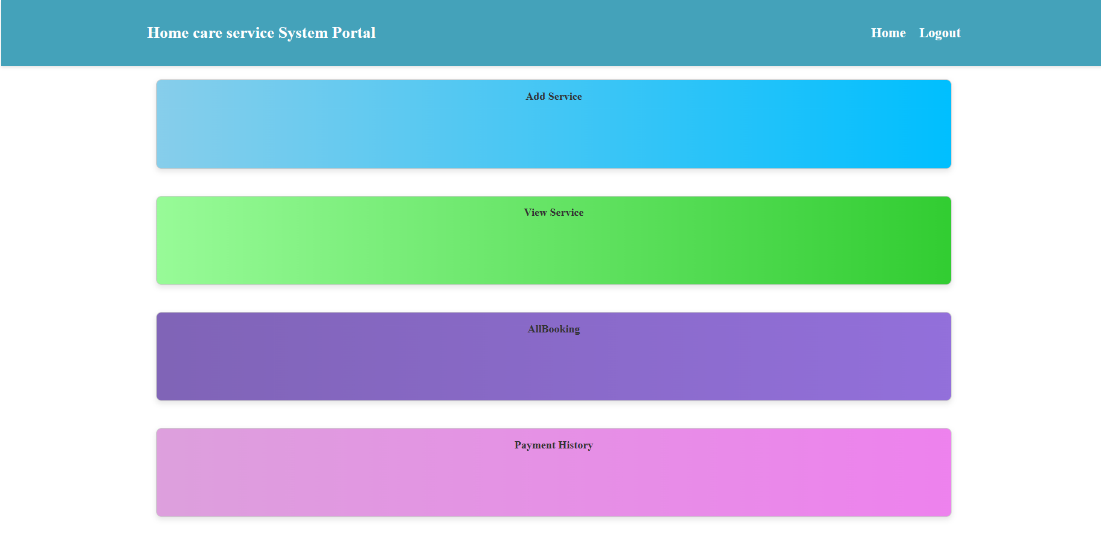
Error messages for field validations are shown using small elements with class text-danger.

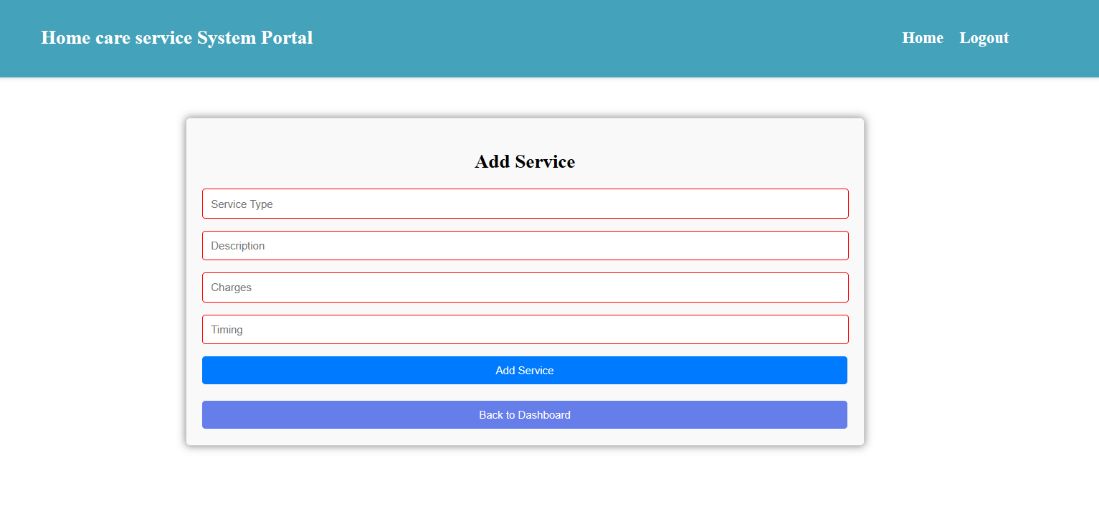


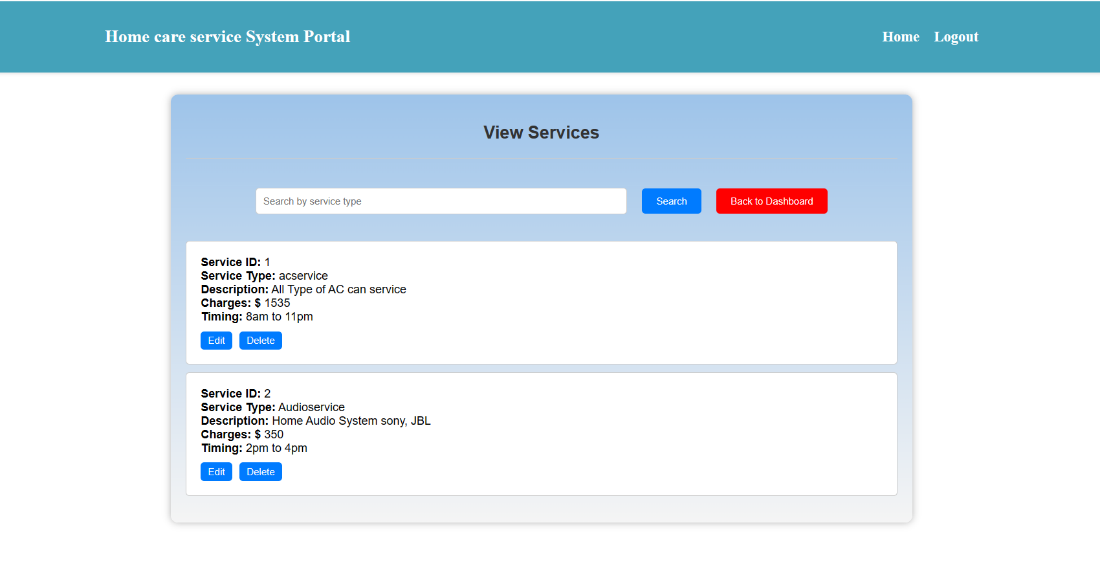


Admin Dashboard :

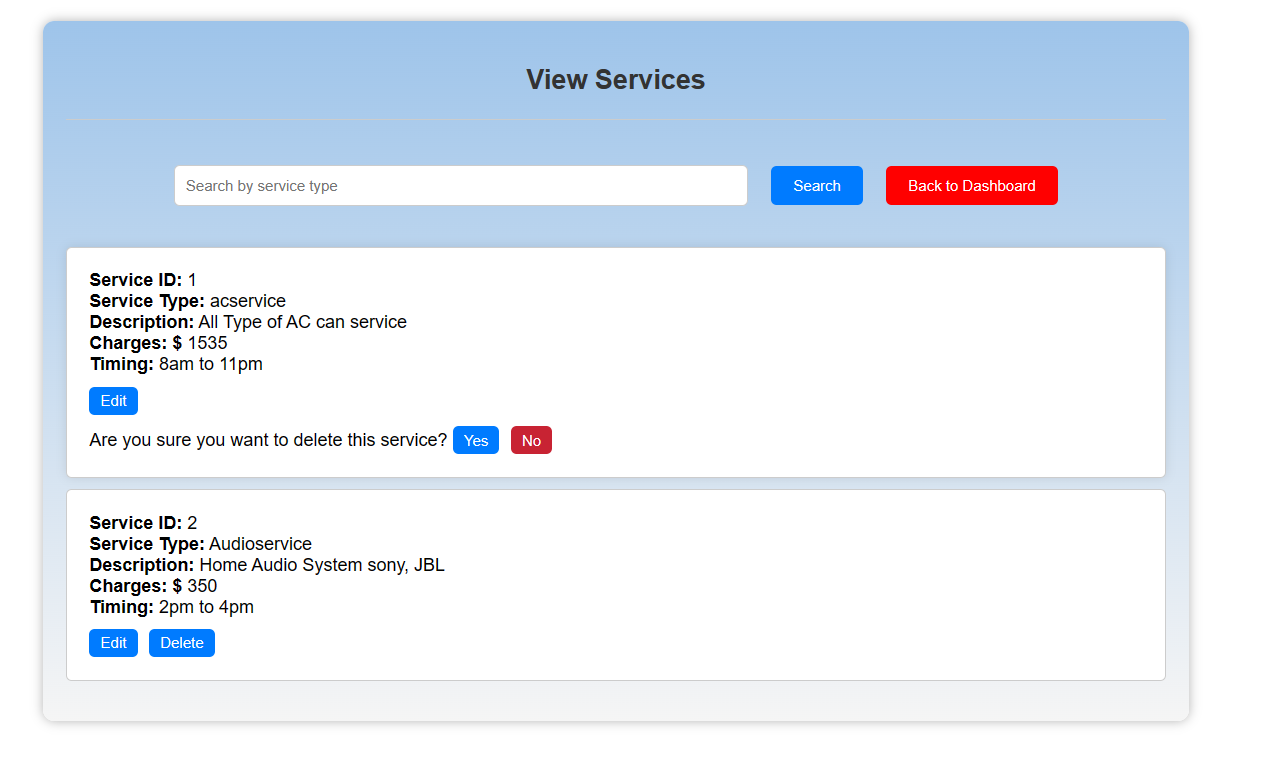
Implement route to navigate to page

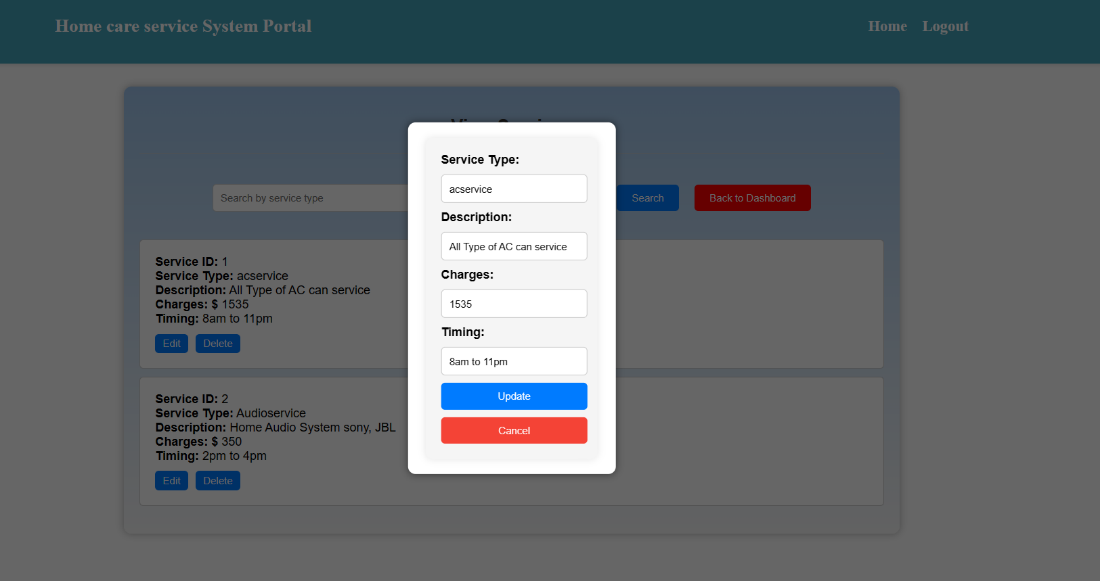


Admin will add the service details by clicking “Add Service” in dashboard  
  
  
  
Admin can View All Services and can edit & delete the service details:

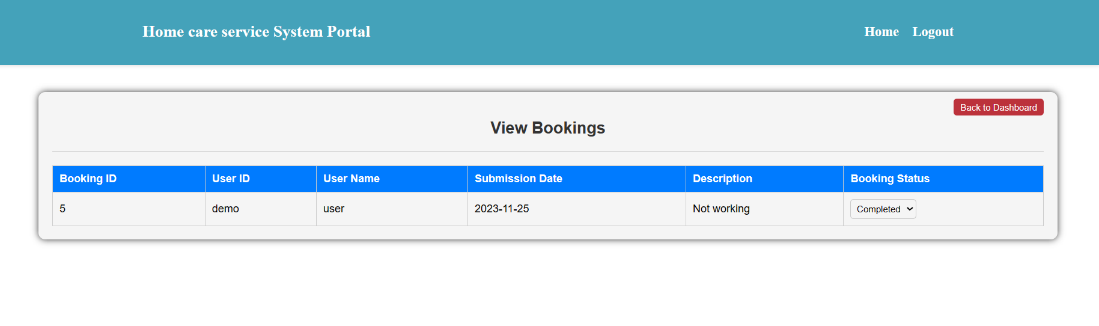


confirmation model is displayed on clicking “delete” button

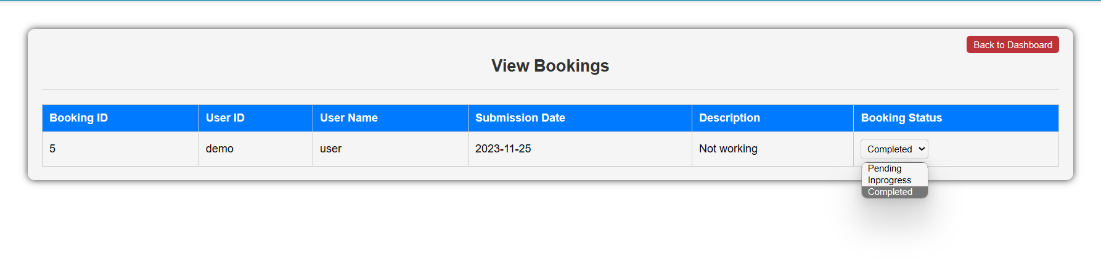


Admin an edit the service details  
  


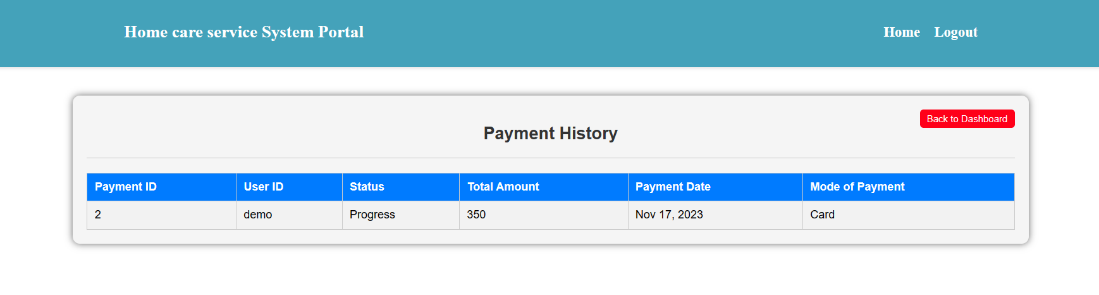
Admin can view all Booking’s done:



Admin can edit Status of the User Booking:



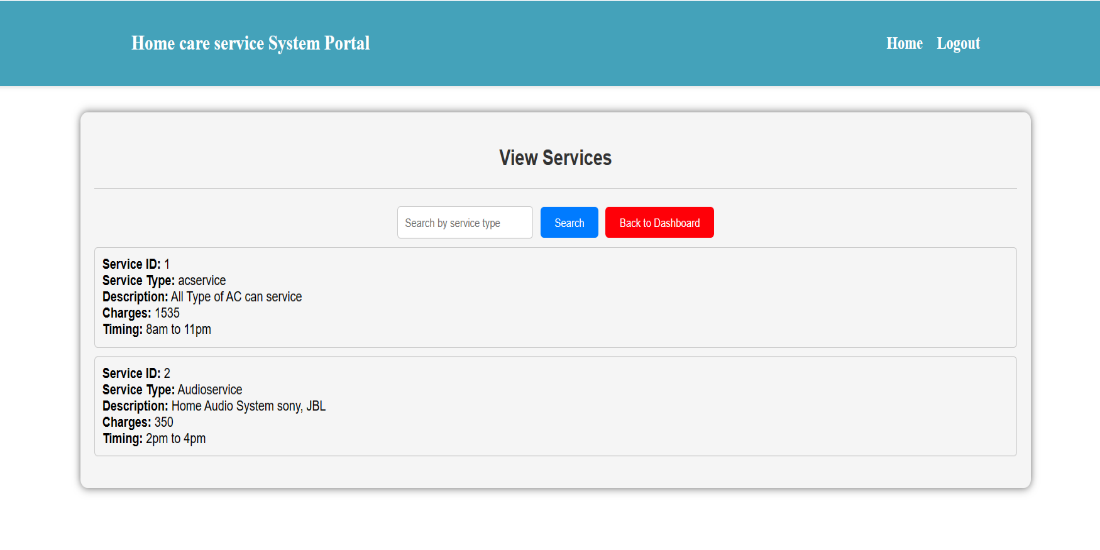
Admin can view all Payment :



User Side:  
  
User Dashboard:

A screenshot of a web page

Description automatically generated

User Can View the Service:  
  
onclicking “View Service” in dashboard, all the available service details will be displayed.  
  
  
The service Type should fetch form the endpoint url and the current services available should load in the dropdown.

User Can Book the Service:  
  
Customer Id should be stored once the user is logged in successfully and it should be prepopulate in the input field “Customer Id” as in the below screenshot.  
  
A screenshot of a computer

Description automatically generated

User Can view mybooking can Delete and Make payment for each booking

User Can Also check Booking Status :

A screenshot of a computer

Description automatically generated

A screenshot of a website

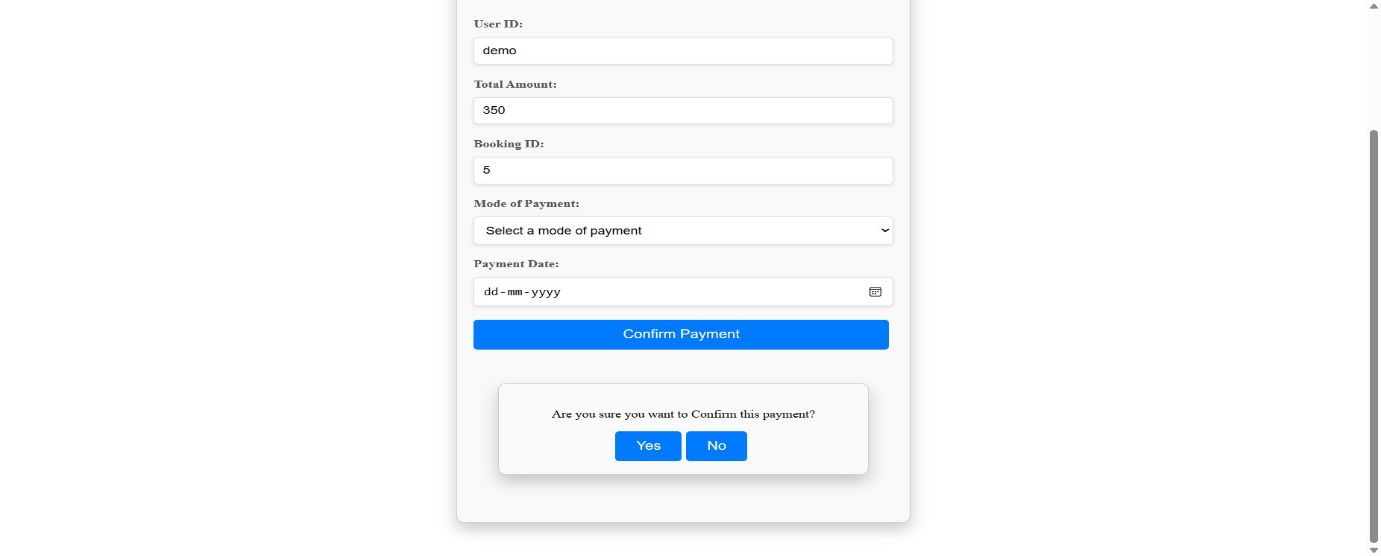
Description automatically generated

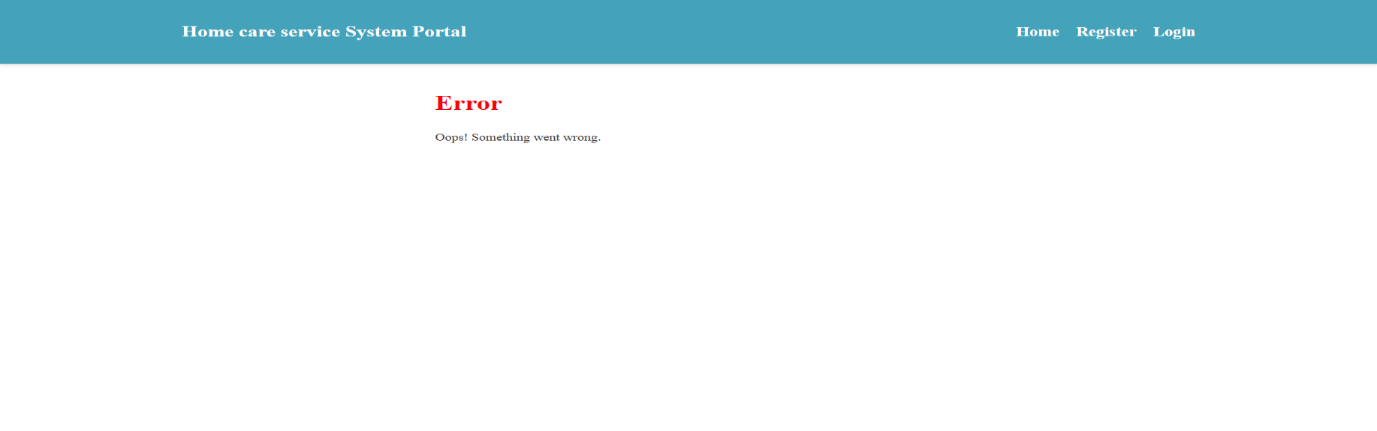
On Clicking Make Payment :

 -> Model Displayed with Data

A screenshot of a computer

Description automatically generated



Error message page for both Admin and User.

**Platform Prerequisites (Do’s and Don’ts):**

**Ø** The react app should run in port 8081.

Ø The spring boot app should run in port 8080.

**Other Important Key factors in the application:**

**•** Should use Custom Exceptions mandatory.

• Tables should have proper relationship and keys

• Frontend Application should be menu driven.

• Proper Menu / Navigation for corresponding role

• Client side validations and server side validations are mandatory

• Error should be handled.

• Follow best programmer practice while developing

• Provide proper Naming Conventions

**Key points to remember:**

**1.**The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.

2.    Remember to check the screenshots provided with the SRS.

3.    Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.

4.    Adhere strictly to the endpoints mentioned in API endpoints section.

5.    Don't delete any files in a project environment.

**HOW TO RUN THE PROJECT :**

**FRONTEND:**

**Step 1:**

Use “nvm use 14” cmd to change node version to 14

**Step 1:**

Use "cd reactapp" command to go inside the reactapp folder

Install Node Modules **- "**npm install**"**

**Step 2:**

Write the code inside src folder

Create the necessary components

**Step 3:**

Click the run test case button to run the test cases

**Note :**

* Click PORT 8081 to view the result / output
* If any error persists while running the app , delete the node modules and reinstall them

**BACKEND**

**API endpoint:**

8080

**Platform Guidelines:**

To run the command use **Terminal**in the platform.

**Spring Boot:**

Navigate to the springapp directory => **cd springapp**

To start/run the application '**mvn spring-boot:run**'

**To Connect Database open terminal**

Cmd: Mysql -u root –protocol=tcp -p

Password: examly