Online Beautician Booking System

Objective:

Beautician System is an online application to book online appointment with a Beauty Parlour. User should be able to book appointments with the Beautician based on beautician availability. The Beautician should be able to accept or reject a request.

Users of the System:

1. Admin
2. Beauty Parlours
3. User

Functional Requirements:

- Clients should be able to check Beautician availability and book an appointment.
- Beautician should be able to accept or reject appointment.
- Beautician should be able to view all bookings in the system.
- Billing user to bill based on all the transactions done and keep a record of the same.
- **There should be one user per slot.**

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

- Multi-factor authentication for the sign-in process
- Payment Gateway

Output/ Post Condition:

- Weekly based Beauty Parlourwise case report file
- Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Beautician System -< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance • Admin Application < 2 Sec
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover
Logging &	• The system should support logging(app/web/DB) & auditing at

Auditing	all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	Angular 7+ Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do's and Don'ts):

1. The angular app should run in port 8081. Do not run the angular app in the port: 4200.
2. Spring boot app should run in port 8080.

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. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, mhLethod definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

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

2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:4200/signup or http://localhost:4200/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

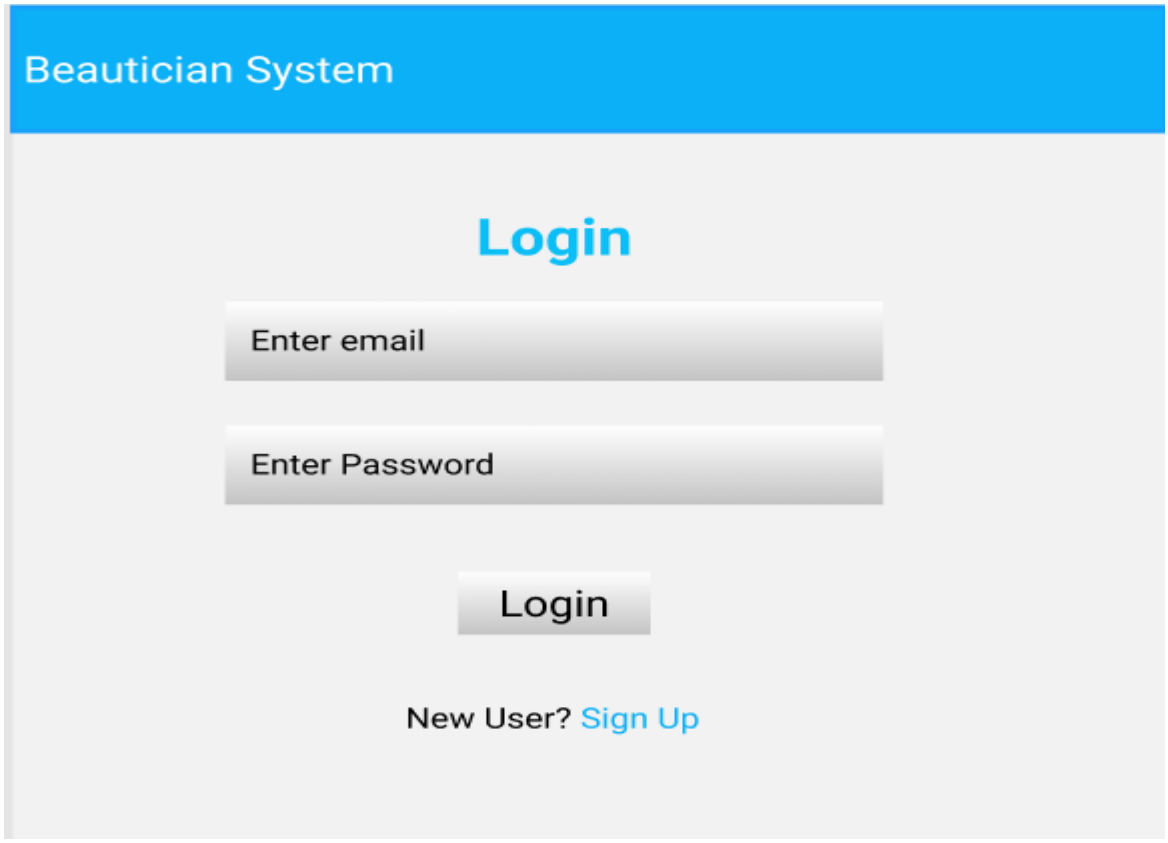
CLIENTS			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get All Beautician	/Beautician	GET	Array of Beauticians
Add Booking	/booking	POST	Booking Created
Remove Booking	/booking/{id}	DELETE	Booking Removed
Get Appointment	/Appointment/{id}	GET	Return the Appointment based on id
Get Appointment Report	/checkupReport/{id}	GET	Return the resport
BEAUTICIAN			
Action	URL	Method	Response
Get All Booking	/Beautician/booking	GET	Array of Booking
Approve Booking	/ Beautician/booking	POST	Booking Appproved
Reject Booking	/ Beautician/booking/{id}	DELETE	Booking Deleted
Add Appointment	/Beautician/Appointment	POST	Appointment Created
Update Appointment	/Beautician/Appointment/{id}	PUT	Appointment Updated
Delete Appointment	/Beautician/Appointment/{id}	DELETE	Appointment Deleted

Frontend:

Client:

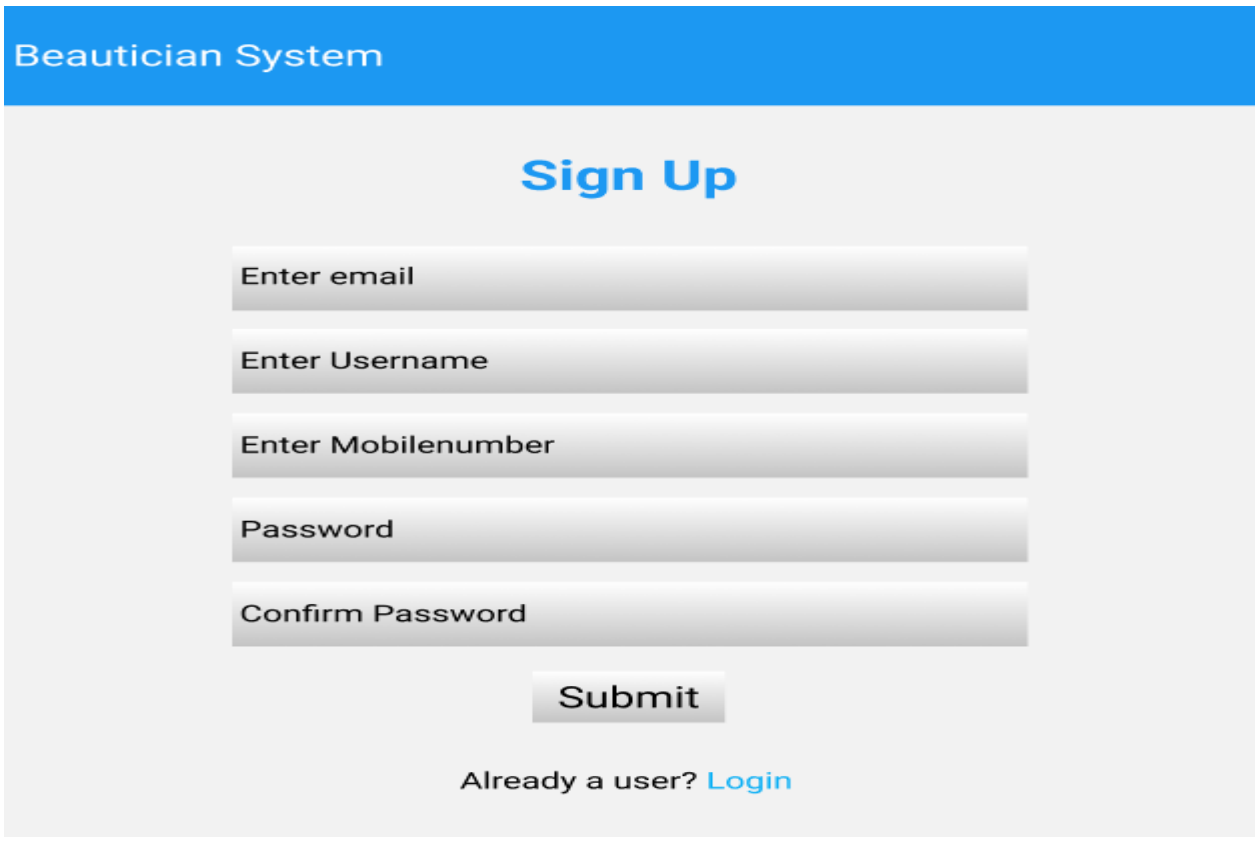
Login:

Output Screenshot:



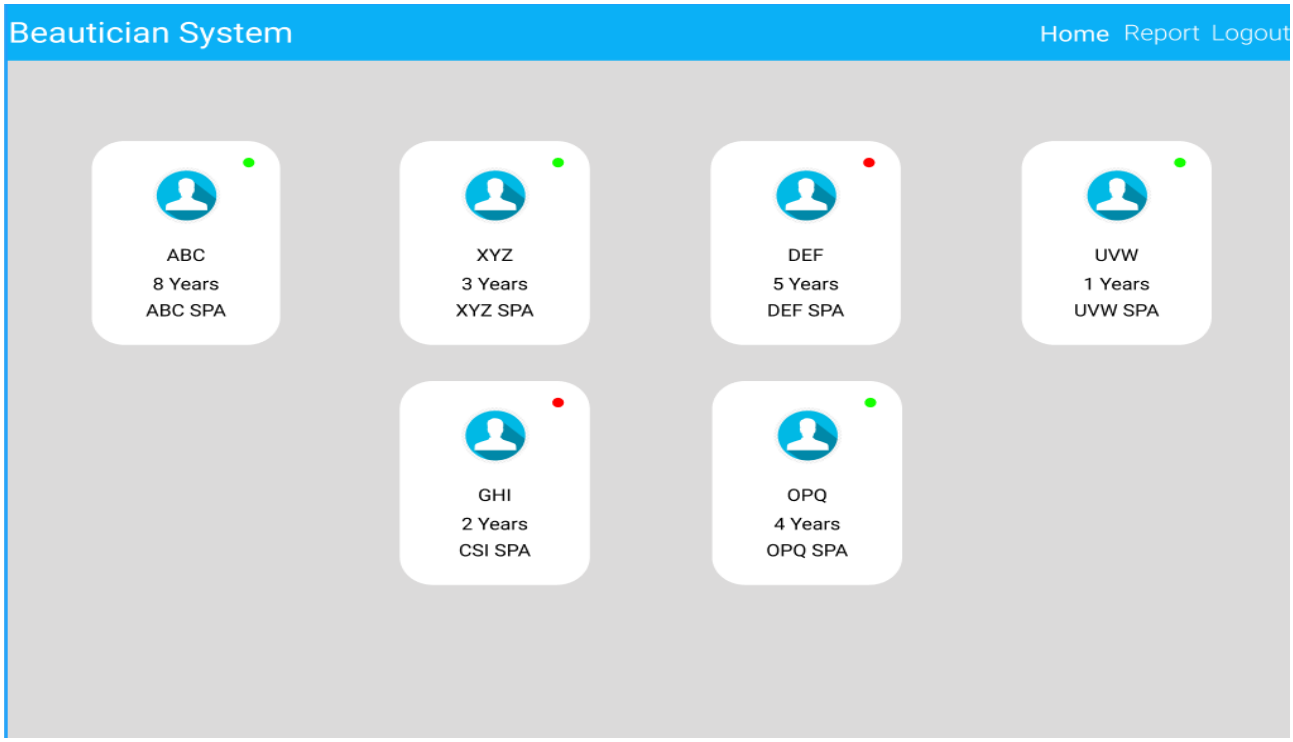
Signup:

Output Screenshot:



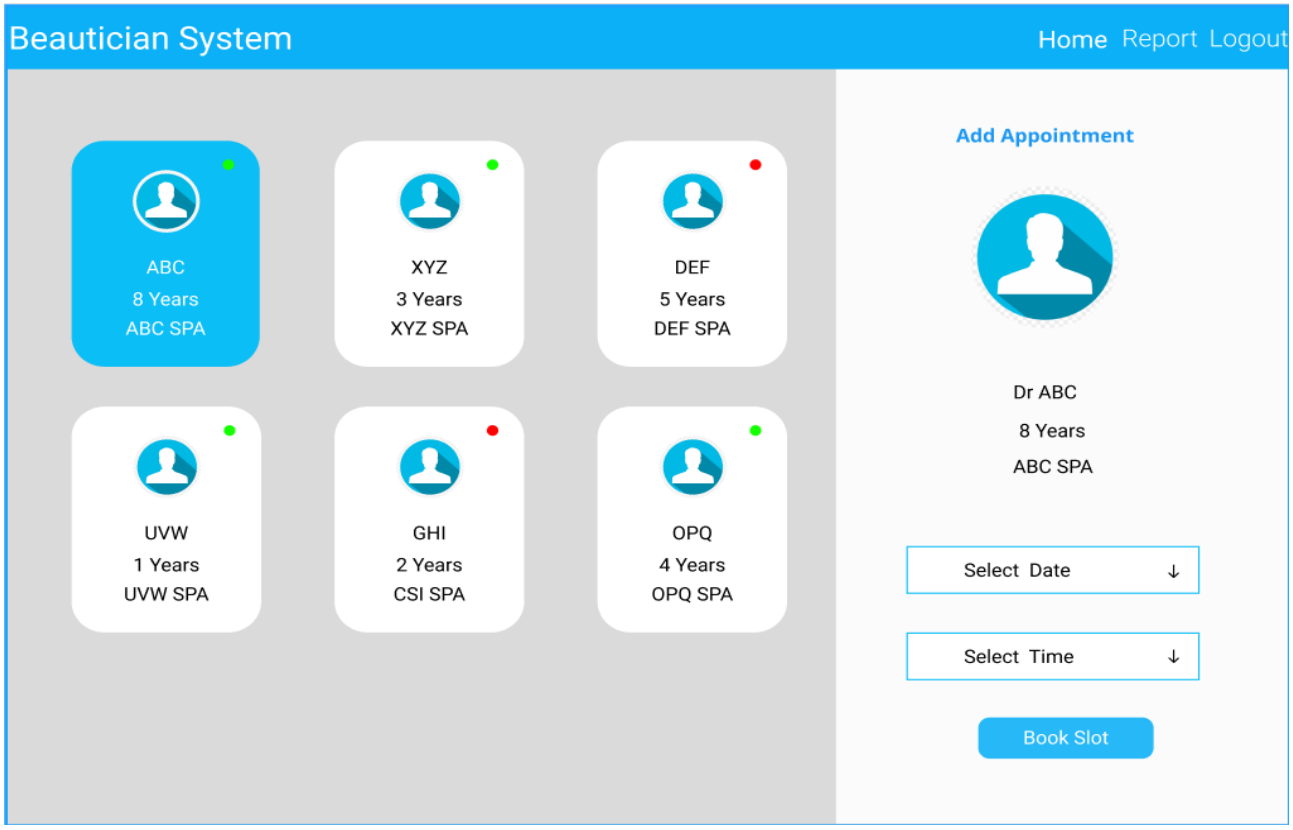
Home:

Output Screenshot:



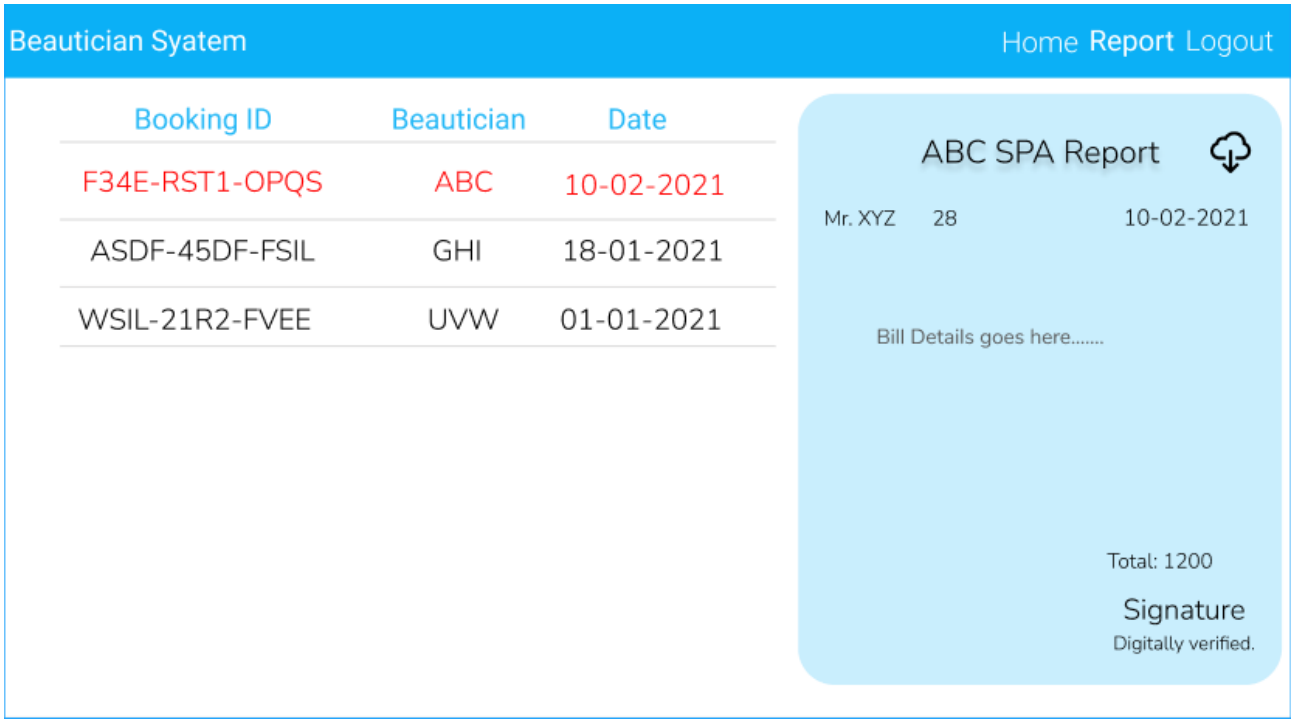
Appointment:

Output Screenshot:



Report:

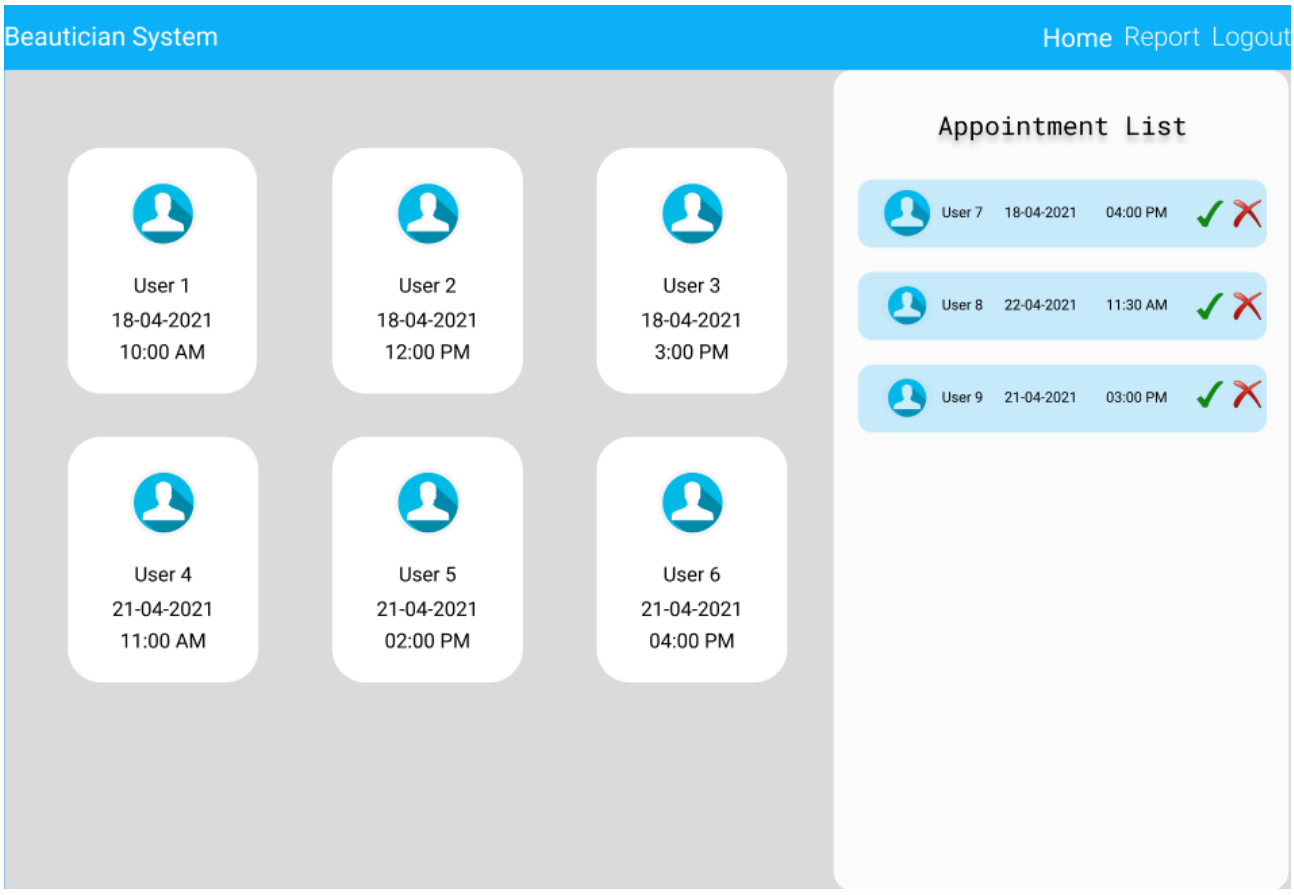
Output Screenshot:



Beautician:

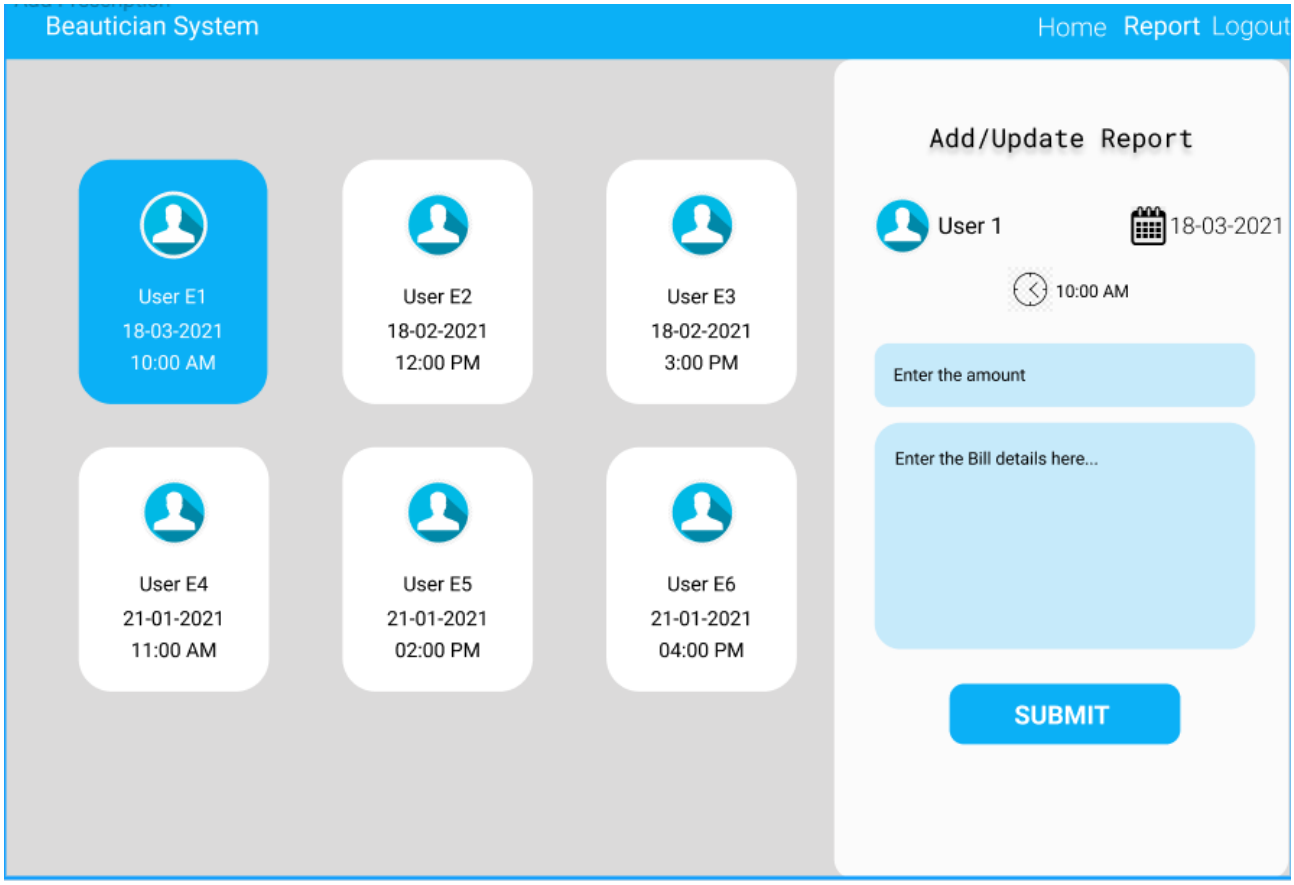
Home:

Output Screenshot:



Report:

Output Screenshot:



Backend:

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the customer) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. username: String
        iv. mobileNumber: String
        v. active: Boolean
        vi. role: String
    b. Methods: -

2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. BookingModel: This class stores the appointment details.
    a. Attributes:
        i. bookingId: String
        ii. clientDetail: UserModel
        iii. BeauticianDetail: BeauticianModel
        iv. lawfirmName: String
        v. date: Date
        vi. time: Date
        vii. bookingStatus: Boolean
    b. Methods: -
4. AppointmentModel: This class stores the Appointment details for the users.
    a. Attributes:
        i. AppointmentID: String
        ii. userId: UserModel
        iii. date: Date
        iv. description: List <String>
        v. issuedBy: UserModel
    b. Methods: -
5. ReportModel: This class stores the.
    a. Attributes:
        i. AppointmentDetail: UserModel
        ii. date: Date
        iii. amount: String
        iv. report: String
        v. issuedBy: UserModel
    b. Methods: -

Controller Layer:

6. SignupController: This class control the user signup

a. Attributes: -

b. Methods:

i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.

7. LoginController: This class controls the user login.

a. Attributes: -

b. Methods:

i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false

8. BookingController: This class controls the adding, upding, removing the booking details.

a. Attributes: -

b. Methods:

i. List<BookingModel> getBooking(): This method helps the admin to fetch all Booking from the database.

ii. List< BookingModel > getBookingByBeautician(): This method helps the Beauty Parlourto retrieve their all the booking from the database.

iii. BookingModel bookingById(String id): This method helps to retrieve a booking from the database based on the bookingId.

iv. statusModifier(BookingModel data): This method helps the Beauty Parlourto edit a booking and save the status as Aprrove or Reject.

v. addBooking(BookingModel data): This method helps the client to add a new booking to the database.

vi. removeBooking(String id): This method helps the Beauty Parlourto delete a booking from the database.

9. AppointmentController: This class helps in adding the Appointment, deleting the Appointment from the cart, updating the Appointment.

a. Attributes: -

b. Methods:

i. addAppointment(AppointmentModel data): This method helps the Beauty Parlourto add the Appointment to the user.

ii. updateAppointment(AppointmentModel data): This method helps to update the Appointment.

iii. deleteAppointment(String id): This method helps the Beautician to delete a Appointment from the user.

10. ReportController: This class helps with the Beauty Parlourto create/read/update the report details about the Clients.

a. Attributes: -

b. Methods:

   i. List<ReportModel> getCheckupDetails(String id): This method helps to list the details based on the userI id.

   ii. addReport(ReportModel data): This method helps to save the report details in the database.

   iii. updateCheckup(ReportModel data): This method helps to update the report details and store it in the database.