

## **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:

<b>Security</b>	<ul style="list-style-type: none"><li>• App Platform –UserName/Password-Based Credentials</li><li>• Sensitive data has to be categorized and stored in a secure manner</li><li>• Secure connection for transmission of any data</li></ul>
<b>Performance</b>	<ul style="list-style-type: none"><li>• Peak Load Performance</li><li>• Beautician System &lt; 3 Sec</li><li>• Admin application &lt; 2 Sec</li><li>• Non Peak Load Performance</li><li>• Admin Application &lt; 2 Sec</li></ul>
<b>Availability</b>	<ul style="list-style-type: none"><li>• 99.99 % Availability</li></ul>
<b>Standard Features</b>	<ul style="list-style-type: none"><li>• Scalability</li><li>• Maintainability</li><li>• Usability</li><li>• Availability</li><li>• Failover</li></ul>
<b>Logging &amp; Auditing</b>	<ul style="list-style-type: none"><li>• The system should support logging(app/web/DB) &amp; auditing at all levels</li></ul>

<b>Monitoring</b>	<ul style="list-style-type: none"> <li>• Should be able to monitor via as-is enterprise monitoring tools</li> </ul>
<b>Cloud</b>	<ul style="list-style-type: none"> <li>• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure</li> </ul>
<b>Browser Compatible</b>	<ul style="list-style-type: none"> <li>• IE 7+</li> <li>• Mozilla Firefox Latest – 15</li> <li>• Google Chrome Latest – 20</li> <li>• Mobile Ready</li> </ul>

### Technology Stack

Front End	React 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 React app should run in port 8081. Do not run the React app in the port: 3000.
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, method 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 AuthGuard by implementing the canActivate interface. For example, if the user enters as

<http://localhost:3000/signup> or <http://localhost:3000/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 Approved
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:

Beautician System

Login

Enter email

Enter Password

Login

New User? [Sign Up](#)

**Signup:**

Output Screenshot:

Beautician System

Sign Up

Enter email

Enter Username

Enter Mobilenumber

Password

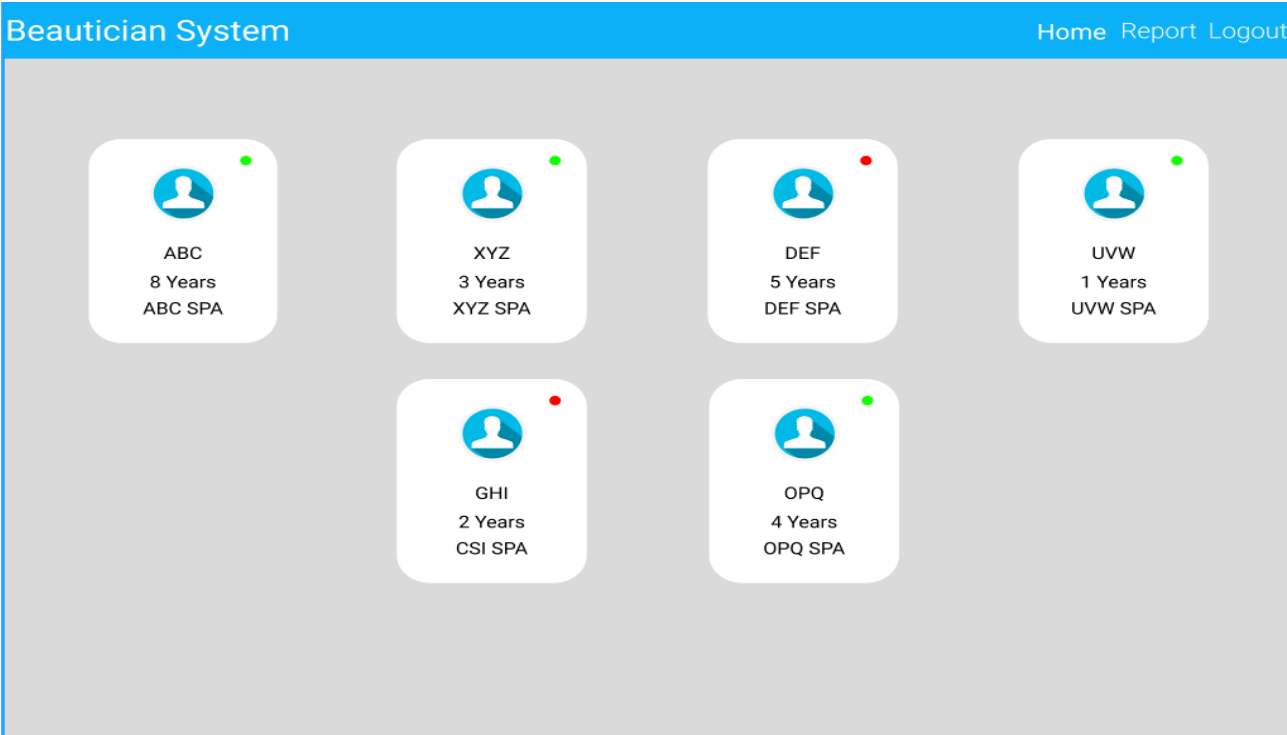
Confirm Password

Submit

Already a user? [Login](#)

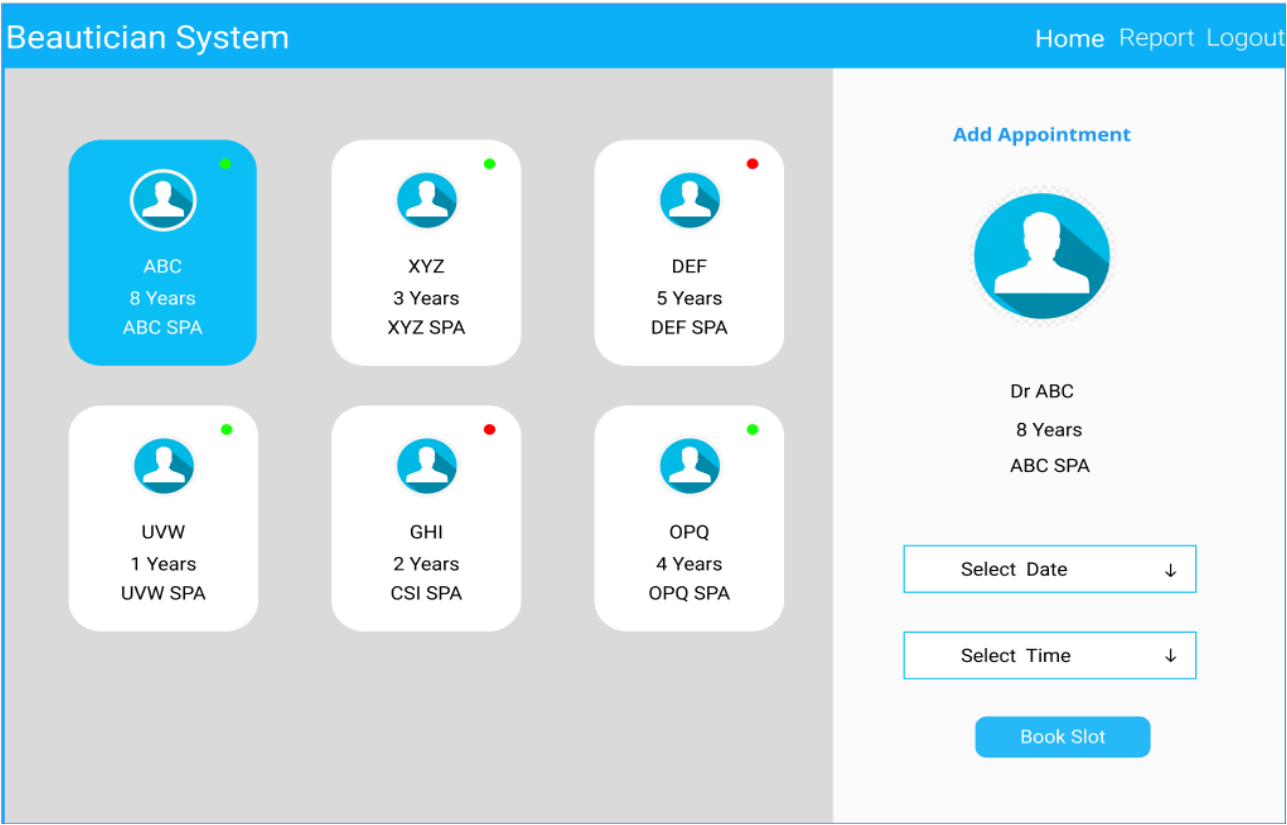
**Home:**

Output Screenshot:



Appointment:

Output Screenshot:



Report:

Output Screenshot:

Beautician Syatem

Home Report Logout

Booking ID	Beautician	Date
F34E-RST1-OPQS	ABC	10-02-2021
ASDF-45DF-FSIL	GHI	18-01-2021
WSIL-21R2-FVEE	UVW	01-01-2021

ABC SPA Report

Mr. XYZ    28    10-02-2021

Bill Details goes here.....

Total: 1200

Signature  
Digitally verified.

**Beautician:**

**Home:**

Output Screenshot:

Beautician System

Home Report Logout

User 1  
18-04-2021  
10:00 AM

User 2  
18-04-2021  
12:00 PM

User 3  
18-04-2021  
3:00 PM

User 4  
21-04-2021  
11:00 AM

User 5  
21-04-2021  
02:00 PM

User 6  
21-04-2021  
04:00 PM

Appointment List

User 7    18-04-2021    04:00 PM    ✓ ✗

User 8    22-04-2021    11:30 AM    ✓ ✗

User 9    21-04-2021    03:00 PM    ✓ ✗

**Report:**

## Output Screenshot:

The screenshot displays the 'Beautician System' interface. The top navigation bar includes 'Home', 'Report', and 'Logout'. The main content area is divided into two sections. The left section features a grid of six user cards, each with a profile icon, a user ID (User E1 to User E6), a date, and a time. The right section is titled 'Add/Update Report' and contains a form for entering report details. The form includes a user selection dropdown (currently showing 'User 1'), a date picker (showing '18-03-2021'), a time picker (showing '10:00 AM'), a text input field for 'Enter the amount', a larger text area for 'Enter the Bill details here...', and a blue 'SUBMIT' button.

User ID	Date	Time
User E1	18-03-2021	10:00 AM
User E2	18-02-2021	12:00 PM
User E3	18-02-2021	3:00 PM
User E4	21-01-2021	11:00 AM
User E5	21-01-2021	02:00 PM
User E6	21-01-2021	04:00 PM

## 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 Approve 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 user 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.