HOTEL MANAGEMENT SYSTEM

eDAC Sept 2020 Batch

By :

Abdul Azeem Khan(200950181001),

Syeda Tasneem Fathima(200950181105),

Mohd. Danish(200950181053),

Filza Ali(200950181033)

WHAT THE PROJECT DOES

- A hotel management app to Enable users to log in and book hotels among all the available hotels.
- A user can log in with his/her credentials.
- New User can register by going to register.
- On successful register users can log in with their credentials.
- After successful login user will be directed to their homepage
- The homepage will have the list of hotels and users can enter the number
  of rooms they want to book and click book next to the name of the hotel they
  want to book
- The user will then be officially checked in and will be redirected to checked In page.
- From there User can download their payment invoice or check out at any time.
- After check out users will be back to their home page.
- On homepage users can edit their account details.
- The Admin login consists of Admin who will login with His/Her credentials
  and come to Admin module
- Amin Id is “Admin1” and password is 123456.
- Admin homepage will have the list of all hotels.
- Amin can add a new Hotel, update an Existing Hotel or delete an existing Hotel.

TYPICAL SCENARIOS

HOTEL

- Add new Hotel
- Update Hotel
- Get All Hotels
- Get Hotel Details By Hotel Name
- Delete Hotel
- All use Hotel repository Object in Service Layer
- Hotel Repository created via JPA and spring-boot

USER

- Add new User
- Update User
- Get All Users
- Get Hotel User Details By email
- Validate User with email and password (Login)
- All use User repository Object in Service Layer
- User Repository created via JPA and spring-boot

BOOKING

- Add new Hotel
- Update Hotel
- Get All Bookings
- Get Booking Details By Hotel Name
- Delete Booking
- All use Booking repository Object in Service Layer
- Booking Repository created via JPA and spring-boot

SCENARIO

LOGIN

- validateLogin method in Usercontroller gets the email ID and password entered by the user in react front-end through path variables. It returns a CustomResponseEntity object with Status as the transaction status and content as the username of the user if present otherwise an empty string.
- It'll the called the UserService method validate() through the autowired UserService object which is autowired using the @autowired annotation provided by Spring
- In UserService there's an autowired UserRepository object.
- UserRepository is the JPA repository which enables us to store changes in the "customer" table mapped to the User "Entity" class.
- Now the validate function in UserService does a findById() on the email to find the user by that email and Stores that in an Optional user object.
- findById() is a default method provided by JPA to access all details of the table by ID i.e., the Primary key.
- If the user by that email is actually present Optional object will have that user which we can get easily by .get().
- Now we check that user's password from the password the customer entered in react-frontend and which we got as input parameter in validate function.
- validate() then returns a CustomResponseEntity object to validateLogin() in UserController which in turn returns it to react.
- In react we check if the email is not an empty string then login successful and cookies are updated accordingly and user is redirected to User Homepage

WHERE WE GOT STUCK?

Security:

- Had to implement Login with Security
- For that we either use session, store token in localStorage or use cookies
- We decided to go with cookies

Booking History

- Had to implement a module where use can see his/her booking history.
- We used One to many relation from User Table to Booking table
- The Booking table contains booking details of all each user.

WHAT WE LEARNT?

- React JS
- Spring Boot
- Connecting Spring Boot with react
- Axios with React
- JPA with Spring
- Using Browser Cookies
- Using APIs in react installed through NPM