**Hotel Management Web Application**

**EDAC august 2020 batch**

**Submitted by**

Lavina khichi

(PRN no.: 200950181049)

Preeti barua

(PRN no.: 200950181075)

Konark Tripathi

(PRN no.: 200950181046)

Vikas

(PRN no.: 200950181112)

**What project does?**

* This is a website for hotel management
* Maintain a database of details of Customers, Rooms and Staff of hotel.
* Accessible for authorities from anywhere to check current updates.
* Others can see Facilities given by hotel on the site Home page and contact details of respective authorities of hotel.
* It has two logins Admin/Receptionist.
* Admin will update and change Staff’s details (Appointment, salary, contact details etc.) and Update (Bedroom type) and Add Rooms.
* Receptionist will maintain Customers details (Id, address and contacts etc), Book rooms (Check-In, Check-out details).
* Also Receptionist will update room’s cleaning status.
* Like this it will help to manage hotel.

**Adding an Employee by Admin**

* By clicking on *Admin login,* admin log in page will appear there admin have to fill login credentials.
* Than after successfully login a post login page will appear.
* In post login page Admin will have four buttons-

1. View all employees, 2) Add employees, 3) Add Rooms 4) Room info.

* When admin will click on *Add Employees* event will get detect and

1. **Presentation Layer**
2. App.js will route to *AddEmployee.jsx* component
3. There local storage *Token* will get checked, if token will be empty the page will not be accessible.
4. If token value will be present it will made add employee accessible.
5. After that admin will enter all the details in the respective text boxes. And click the save button.
6. As soon as button will get clicked *SaveEmployee* function will get called.
7. First it will check if all the text boxes are filled or not, if any of the boxes are empty, if will give an error message “check all the columns are filled , before submitting”
8. If all columns are filled, the values will get *into Employee* object and object will be passed in *CreateEmployee* function of *EmployeeService* Component.
9. There in *CreateEmployee* function axios will connect to the rest-controller of server and call the *addEmployee* function of that and passed the object through Post mapping.
10. The Controller function will return an object with ResponseEntity.*ok* (Httpstatus-200) contains (status and reason).
11. Then in AddEmployee component, Httpstatus will get checked if it is not 200 then it will direct to an error page, which will show “server did not respond” message.
12. If it is 200, it will check *status* of object.
13. If *status* is 0 it will show *reason* as error message.
14. If status is 1 it will route to *ViewEmployee page.*
15. And there the new employee will get added and all details show in a table.
16. **Service layer**
17. The function *addEmployee* in rest-controller will call *addEmployee* function of Service interface
18. And pass the object into it.
19. In service implementation *addEmployee* function firstly will check that the id entered by admin already exist or not.
20. If employee id already exist it will return object with status 0.
21. If not present it will call DAO interface inbuilt save function provided by JPA repository and pass the object to it, and return the object of *CResult* model class with status 1.
22. **DAO Layer**
23. DAO interface will inherit JPA repository.
24. Which will provide all inbuilt function to access database table’s details.

**Scenarios where we got stuck-**

* When we were sending data via axios to database for add customer it was giving error in console and was not able to connect to back-end.

**Reason**- a variable name of the object going to backend was not matching with entity class variable name as first letter was capital in react.

**Approach**- we used browser console and then carefully checked the backend with postman then observed the mistake and resolved it.

* When we were adding customer and changing room status from unbooked to booked then in this we observed in some scenarios the customer got added while room didn’t get updated.

**Reason-** one table got updated and other one gave exception then the customer got added while room didn’t get updated.

**Approach-** we have to use @transactional annotation for applying transaction, we did poc for the same, then applied it. So that it can roll back to previous state if any error or exception occurs in the second table updation.

* When the backend gave exception or didn’t respond then on front end the user usually was not getting any response and was unaware of the situation.

**Reason**- we were not handling the response Httpstatus.

**Approach**- we did poc for responseEntity implementation and then used responseEntity for this at backend and handled it in react with error page using if else condtion.

* When we were trying to access our pages which were supposed to be accessed only after login, the pages were accessible by url and even after logout by clicking back button.

**Reason**- we were not using sessions for user and that is why the pages were accessible to all.

**Approach**- we used localstorage for storing key value pair for every user.

**Our learnings-**

* We learned the most important thing that is how to keep patience and move step by step forward while working on a project.
* We learned the importance of poc in a project.
* We learned about the need of user sessions by applying loalstorage.
* We learned about the need and use of database transactions.
* We learned about Httpresponses that should be handled accordingly in react.
* We learned about react, dataflow from frontend to backend then backend to frontend and springboot.