**Online Job Application Portal**

**Problem Statement:**

In the current employment landscape, the process of job application and management often involves cumbersome procedures for both employers and job seekers. Employers face challenges in efficiently posting job listings and managing applications, while job seekers encounter difficulties in finding relevant opportunities and tracking the status of their applications. The lack of a centralized and user-friendly platform contributes to inefficiencies, data security concerns, and an overall suboptimal experience in the job application process.

**Users of the System:**

**1.     Job Applicant**

a.     Description: Individuals who are looking for job.

b.     Role:

* Create and manage their user accounts
* Check the list of jobs available
* Apply for the job and submit application
* Check the job application status
* Can Pay and get the access for premium jobs

**2.     Admin**

**a.     Description:** The job admin is responsible for managing jobs and overseeing the job application process

**b.    Role :**

* Access and navigate the admin dashboard for an overview of application process
* Manage the jobs and category into paid jobs and free jobs
* Review and evaluate the job application and change the status
* Communicate with job applicants through the portal
* View the payment done by applicants

**Functional Requirements:**

·       Build a portal that enables applicants can get the job details online and apply

·       The applicants can add/edit/view/delete the applications

·       The applicant can edit their user account

·       The applicants can view all free jobs

·       The applicants can pay to view the paid premium jobs

·       The applicant can search for the job

·       If the applicant paid and upgrade to premium they can apply for many jobs

·       The admin can add/edit/delete/view all free jobs

·       The admin can add/edit/delete/view all paid jobs

·       The admin can add/edit/delete/view all job applications

·       The admin will review the job application and change the status of job application such as “Submitted”, “In Review”,”Rejected”, “Accepted’ etc

·       The admin have access to a dashboard for managing job listings, applications.

·       The admin can view all job applicant user account profiles.

**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 job listings, applications, and users and their payments.
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:**

**Job Applicant Side:**

The application flow for the online job application portal begins with user registration, where prospective job applicants create accounts by providing personal information. Upon logging in, users access the user dashboard and view all the free jobs and job details .By default, job applicants, as normal users, can view all free jobs. Whenever they apply for job their dashboard should show the number of jobs applied by that day.

Job applicant can see the premium jobs tab but cannot see the premium job postings. If the job applicant pay the payment for premium user then all the premium job listings will be viewed and also they can apply for n number of jobs. Once the job applicant applied for any job the status will be shown as “Applied” in their Applied Jobs menu.

Throughout the process, job applicants can track the real time status of each job application in their dashboard. They can edit their user account profile. They can do their payment and become as premium customer. They can see their payment history.

**Admin Side:**

The administrative flow within the online job application portal begins with administrators accessing the admin dashboard, providing a comprehensive overview of jobs, application, payments and user accounts. From the dashboard, administrators proceed to add the free jobs or paid jobs. He can edit or delete the job details.

Administrators proceed to the job application review module, where they evaluate and change the status of job applications. He have the ability to mark applications as accepted, rejected, verified, pending based on their evaluations. Administrator will manage the job listings and applications. He can also edit/delete the payment history.

Admin can view all the user account details.

**Abstract**

The Job Application Management System is a web-based application designed to streamline the process of job postings and applications. It offers a convenient platform for companies to list job vacancies and for job seekers to apply for those positions. The system aims to simplify the management of job-related data and enhance the application experience for applicants. This innovative platform caters to two main user roles; Job Applicants and Administrators.

**ROLES:**

1.     ADMIN

2.     APPLICANT

**Mandatory Modules:**

**Modules of the Application:**

**ADMIN:**

1. Register
2. Login
3. Post Free Job details
4. Post Paid Premium Job Details
5. Edit Job Details
6. Delete Job Details
7. View All Jobs
8. View All Applications
9. Change Status of Application
10. View Payment History
11. View all user accounts
12. Logout

**APPLICANT:**

1. Register
2. Login
3. View All Free Jobs
4. Apply for Job
5. Can apply for only 5 jobs per day as normal user
6. Can pay and upgrade to premium user
7. Can do the payment
8. As premium user can view all premium jobs
9. As premium user can apply for n number of jobs
10. Check the Status of Application
11. Delete Application
12. View or Edit his user account
13. Logout

**Technology Stack:**

**Front End**

React, HTML, CSS, JavaScript

**Back End**

Java, Spring Boot, MySQL Database

**Authentication**

JWT for User Authentication

**Application assumptions**:

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

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

3.   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.

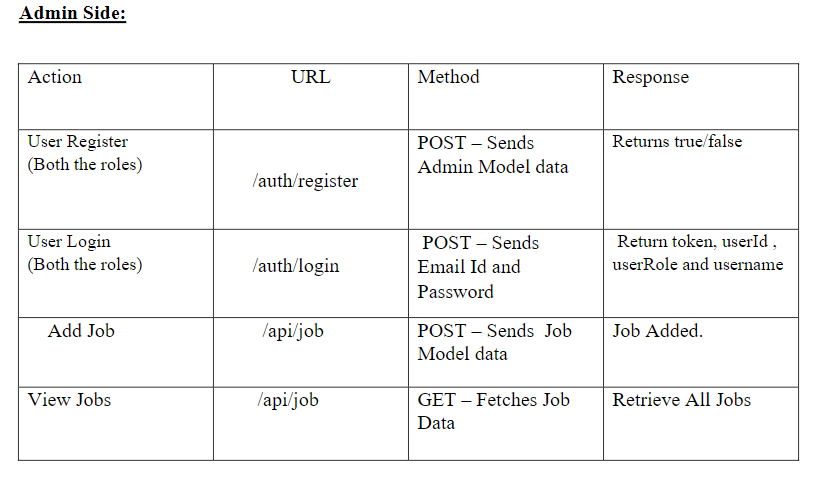
Thus, create a reliable and user-friendly web application that not only meets user expectations but also provides a robust and secure experience, even when faced with unexpected situations.

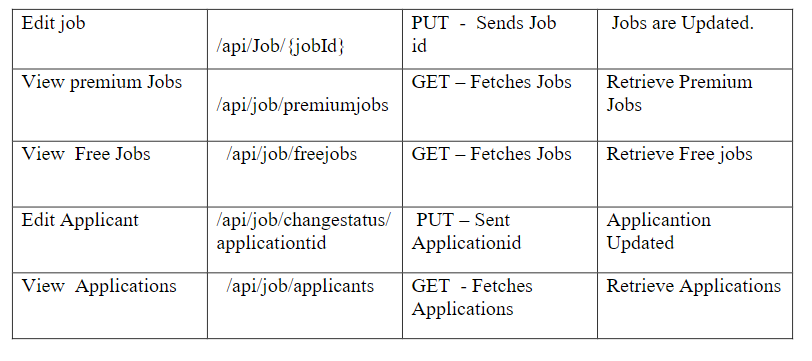
Controller:

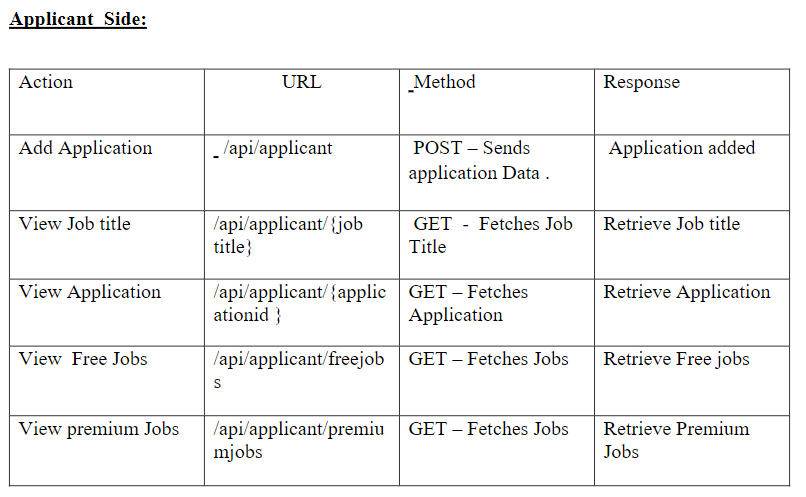
* **AuthController** should take care of authentication

Create Corresponding Controllers for the application

**Service Implementation is mandatory to handle the HttpClient service call to connect web API endpoints**





  
  
public class User

{

  Long id

  String userName

  String password

  String email

  String role

}

class Applicant{

   Long id

   String applicantName

   String jobTitle

   String status

   @OneToOne

   User user;

   @ManyToMany

   List<Job> jobs;

   @OneToOne

   Payment payment;

}

class Job

{

  Long jobId;

  String title;

  String dept;

  String location;

  String responsiblity;

  String qualification;

  Date deadline;

  String category;

  @ManyToMany

  List<Applicant> applicants;

}

class Payment

{

  Long id;

  String status;

  Double totalAmount;

  Date paymentDate;

  String modeOfPayment;

  @OneToOne

  Applicant applicant;

}

class JobsApplied

{

Integer id;

Date appliedDate

@ManytoOne

private Applicant applicant

@ManyToOne

private Job job

}

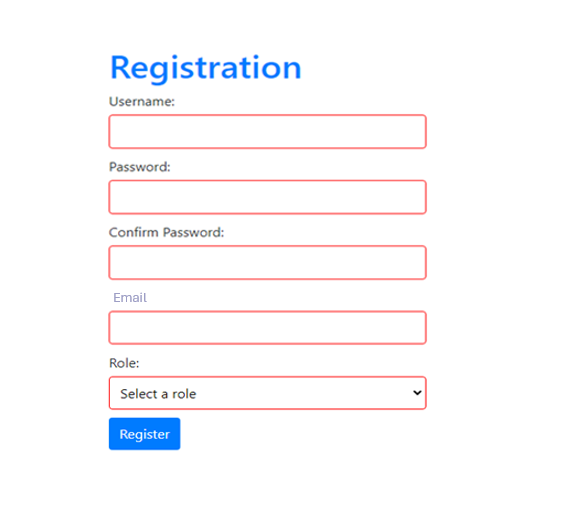
**Frontend Sample screenshots**

Job Applicant Side:

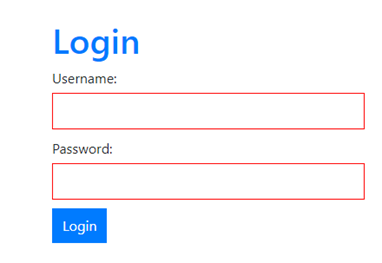
1.     **Auth**: Design an auth component where the student can authenticate login and register credentials.

2.     **Register:** Design a register page component (Name the component as register for react app) inside the auth where the new customer has options to sign up.

Whenever the user are signing up they should select the role.

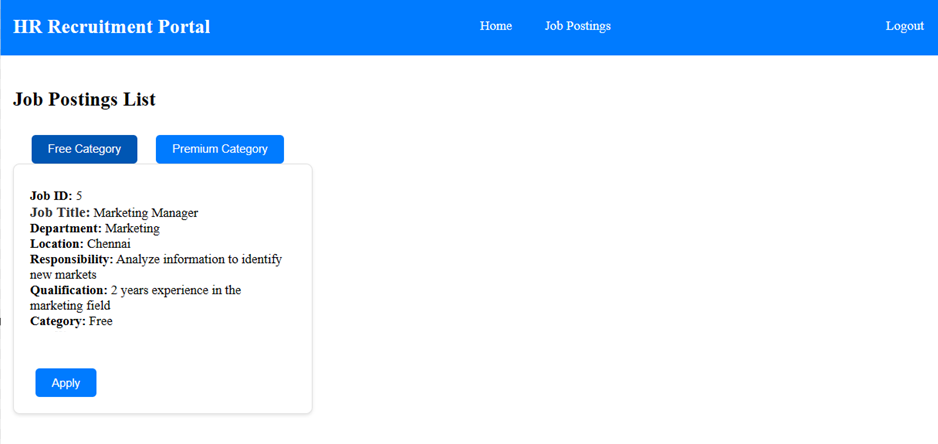


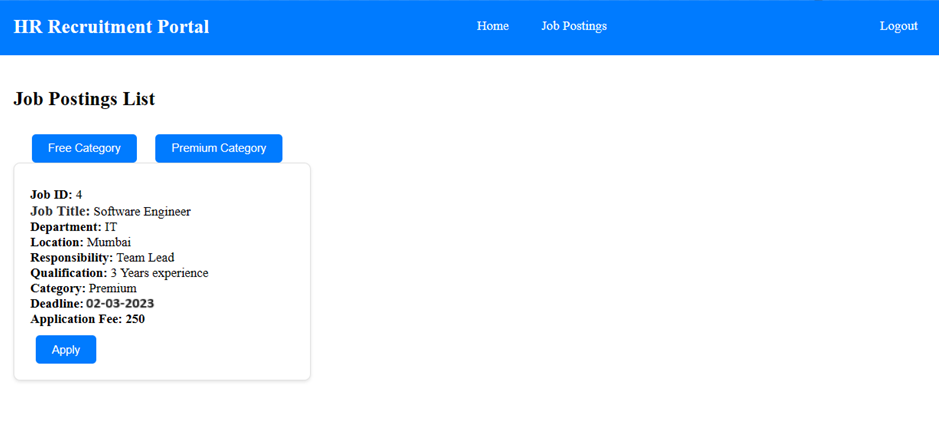
Login page:



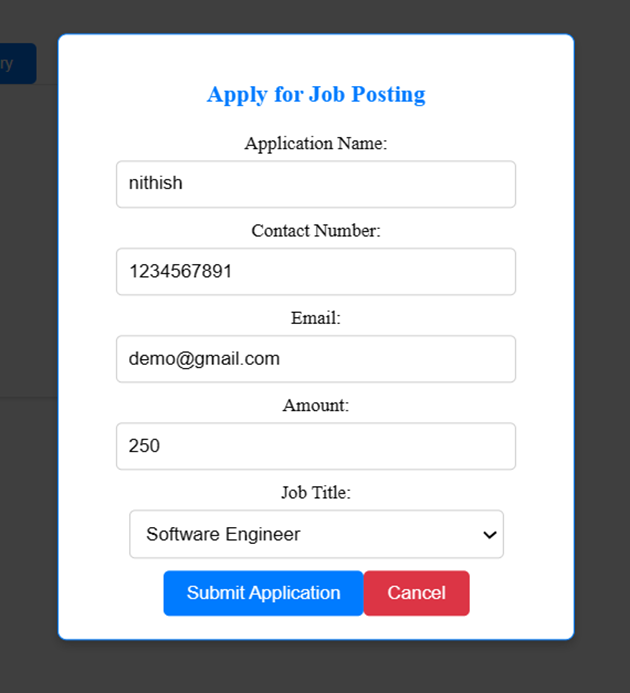
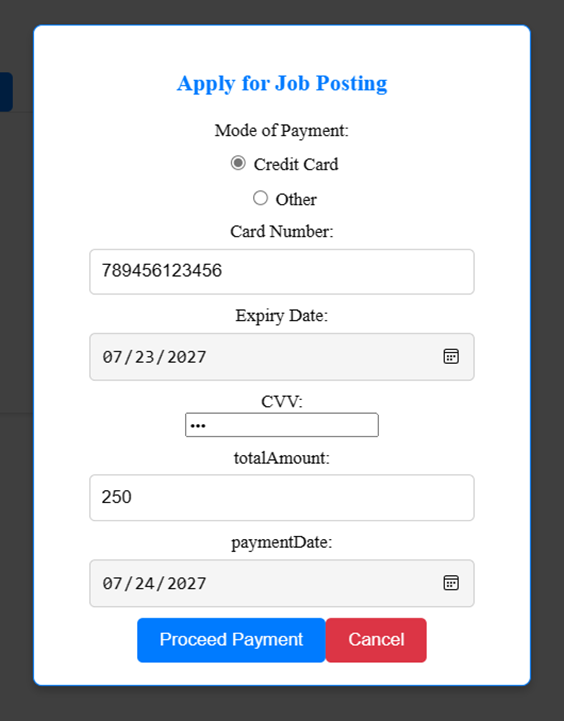
**Applicant Side:**

Once the job applicant logged in, the home page should list the number of free jobs.

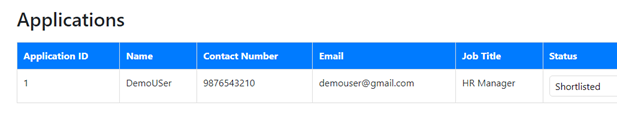


Show the list of premium jobs, this is admin side, it should has actions like edit & delete option, whereas for applicant side, there should be only apply bottom.  
  


Should return the payment status to the job applicant

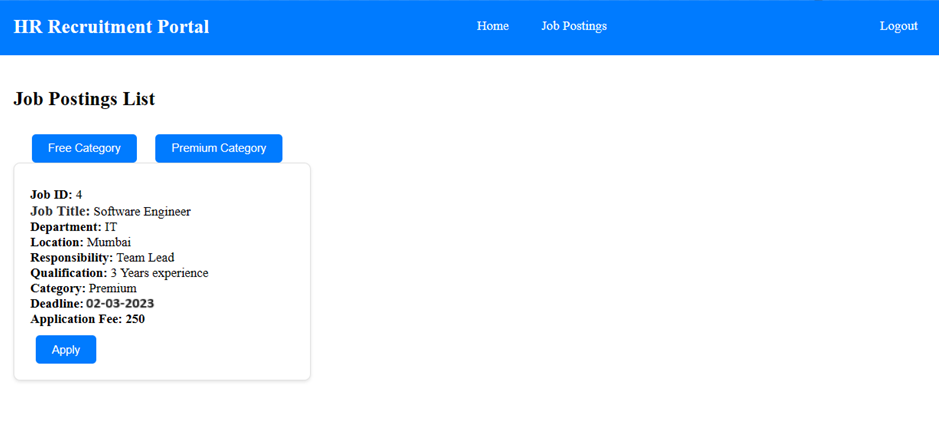
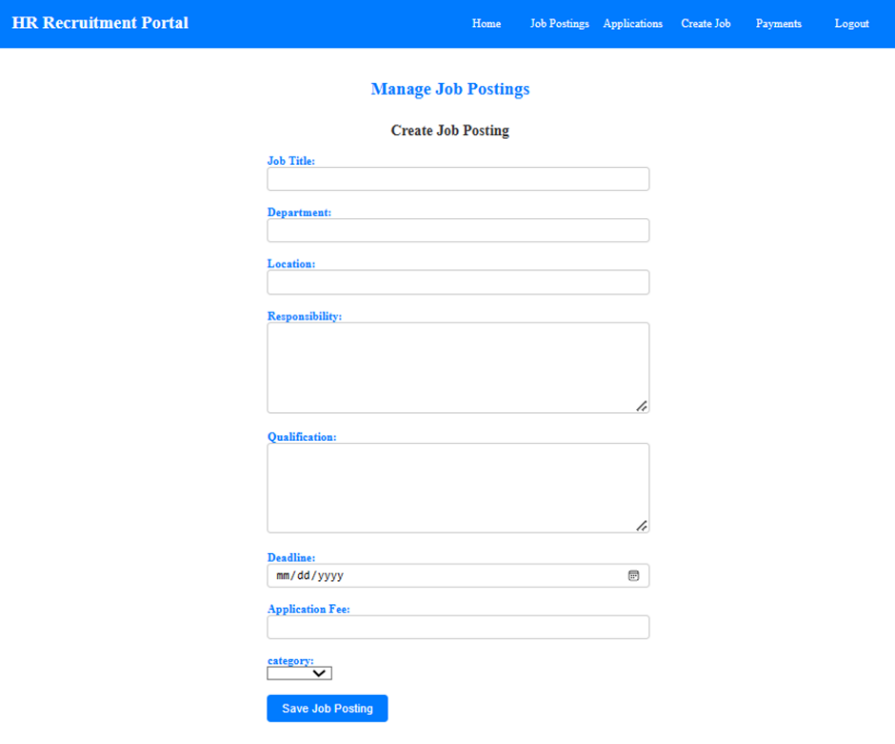
  
  
  


Job Applicant can check the status of all application:

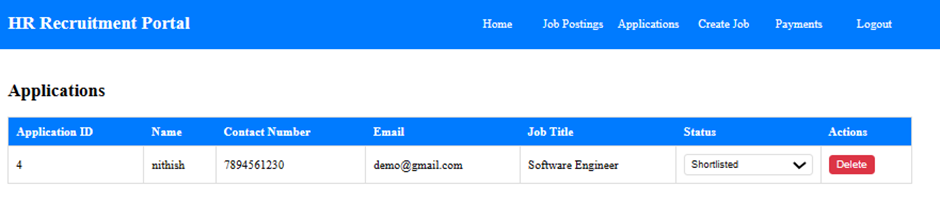


**ADMIN Side:**

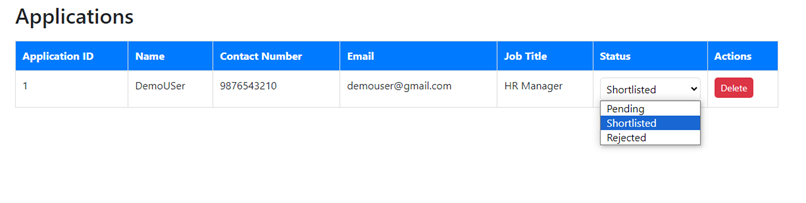
Admin should have the access to view, edit & delete the Job Postings(Premium or free):

  
Add a new job posting and should select the category as free jobs or premium jobs.

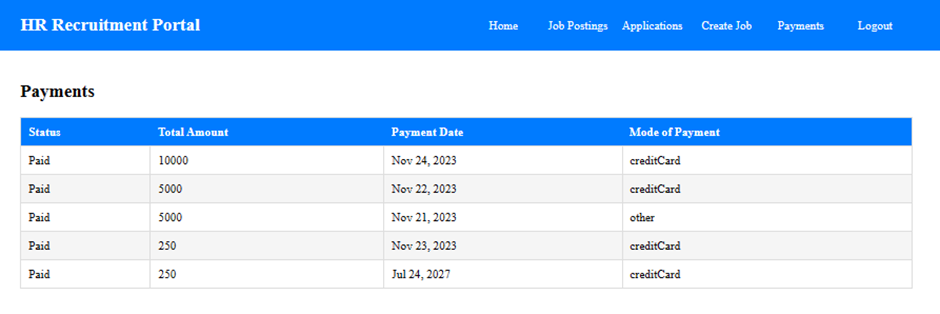
Admin can view the list the job listings



Admin can change the status of the application



Admin can view all the payment History

  
  
**Note:**

•            You should use NotFound(), NoContent(), BadRequest(), CreatedAtAction() to handle the HTTP status code as return values for the Controller methods as mentioned.

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

**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

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

1. The react app should run in port 8081.

2. The springboot app should run in port 8080.

**﻿**

**HOW TO RUN THE PROJECT :**

**FRONTEND:**

**Step 1:**

Open the terminal

Use “nvm use 14” command 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**'

Click on the Run Test Case button to pass all the test cases

**To Connect Database open terminal**

Cmd: mysql -u root –protocol=tcp -p

Password: examly