CareerMap

1. Project Charter:

1.1 Purpose:

The primary purpose of the CareerMap job board web application is to create an intuitive, user-friendly platform that connects job seekers and employers efficiently. This project aims to simplify the process of job hunting by allowing candidates to search for jobs based on location, industry, job type, and other relevant criteria. At the same time, it provides employers with a streamlined method to post job listings, manage applications, and identify suitable candidates. The platform is designed to address the challenges of modern recruitment by offering features such as dynamic search filters, real-time updates, and a seamless user experience, all within a single, integrated web application.

Additionally, the project aims to leverage modern web technologies to ensure a secure, scalable, and high-performing application that can handle a significant amount of data and user traffic. The backend is developed using Java Spring Boot to provide robust and reliable server-side processing, while the frontend is built using React to deliver a responsive and engaging user interface. The integration of a MySQL database ensures efficient data management and retrieval, enabling the application to store and handle large datasets, such as user profiles, job listings, and application records. By achieving these objectives, the CareerMap job board application seeks to create a valuable tool for both job seekers and recruiters, facilitating more effective and targeted job matching in the competitive employment market.

1.2 Objectives

The objectives of the CareerMap project are multifaceted, aiming to create a dynamic, interactive platform that effectively meets the needs of both job seekers and employers. A primary objective is to establish a secure and scalable backend system using Spring Boot, capable of handling all core functionalities such as user registration, authentication, job postings, and application management. Another key objective is to develop a highly responsive and intuitive frontend using React, providing a seamless user experience and efficient access to all functionalities. Furthermore, the project aims to integrate a MySQL database for efficient data management, ensuring reliable storage, quick retrieval, and effective manipulation of data related to job postings, applications, and user profiles. Overall, the project seeks to deliver a robust job board solution that simplifies the job search and recruitment processes, reduces manual overhead, and enhances the experience for all users.

1.3 Timeline:

The project was completed over an intensive period of 10 hours, strategically divided into three distinct phases to optimize time and resources. The first phase, spanning 3 hours, focused on backend development. During this time, the core backend functionalities were implemented,

including the creation of RESTful API endpoints for job postings, user management, and application handling. The second phase, taking 5 hours, concentrated on frontend development using React. This phase involved designing and developing a responsive user interface, creating various components such as the job search, application submission, and user profile management, and integrating these components with the backend APIs. The final phase, lasting 2 hours, was dedicated to rigorous testing and debugging to identify and resolve any issues, ensuring that all functionalities worked as expected and the application was ready for deployment. This structured approach enabled the project to progress efficiently and meet its objectives within the planned timeframe.

2.1 Scope of Work:

The scope of the CareerMap job board web application encompasses all the tasks and activities required to develop a fully functional and efficient platform that serves both job seekers and employers. This includes the design and implementation of a comprehensive backend system using Java Spring Boot, which provides RESTful APIs to handle core functionalities such as user registration, authentication, job posting, job applications, and management of user profiles. The backend development also involves integrating a MySQL database to store and manage all necessary data securely, ensuring data integrity and high performance. Key tasks under this scope include designing the database schema, setting up the necessary tables, creating relationships between entities such as jobs, applications, and users, and implementing robust data handling mechanisms.

On the frontend, the scope covers the development of a responsive and user-friendly interface using React. This includes building various components that enable job seekers to search for jobs, apply for positions, and manage their profiles. For employers, the frontend will provide a streamlined interface to post job listings, review applications, and manage job-related data. The scope also involves ensuring seamless integration between the frontend and backend, enabling real-time data exchange and updates. Additional frontend tasks include implementing dynamic search filters, handling user input validations, and optimizing the application for performance across different devices and browsers.

Beyond development, the scope extends to thorough testing of the application to ensure all functionalities are working as expected. This includes conducting unit tests, integration tests, and user acceptance tests to identify and resolve any bugs or issues. The project also encompasses preparing the application for deployment by configuring the necessary environment, optimizing for security and performance, and creating comprehensive documentation for future maintenance and upgrades. The deployment process includes steps for both staging and production environments to ensure a smooth transition from development to live use. The scope of work concludes with a post-deployment review and analysis to ensure the application meets all specified requirements and objectives.

2.2 Deliverables:

The project delivers a fully functional job board web application that meets all specified requirements. The primary deliverable is a robust backend system developed using Spring Boot, which includes all necessary API endpoints to handle job postings, user management, and application processes. Another deliverable is a responsive and intuitive frontend built with React, which offers a seamless user experience for job seekers and employers. The project also includes a MySQL database setup, designed to manage and store all essential data related to users, jobs, and applications effectively. Additionally, the deliverables include comprehensive project documentation detailing the system architecture, user guides to assist users in navigating the platform, and thorough testing reports that verify the functionality and reliability of the application under various conditions. Finally, the project deliverables also cover deployment readiness, with all necessary configurations and environment setups documented to facilitate a smooth transition to a live environment.

2.3 Acceptance Criteria:

The acceptance criteria for the CareerMap project are defined based on the successful implementation of all key functionalities and requirements. The backend must be fully operational, with all API endpoints correctly handling requests and responses in a secure manner. The frontend must be user-friendly, responsive, and capable of providing all intended functionalities, such as job search, application submission, and user profile management. The application should be free from critical bugs and demonstrate reliability and stability under expected usage conditions. Furthermore, the MySQL database must efficiently manage all data transactions, ensuring quick data retrieval and secure storage. The project will be deemed complete and acceptable when all these criteria are met, and the application is ready for deployment, offering a seamless and intuitive experience for both job seekers and employers.

3. Requirements Documentation:

3.1 Target Audience:

The CareerMap job board web application is designed to cater to two primary audiences: job seekers and recruiters. Job seekers include recent graduates, professionals looking for new opportunities, and individuals seeking to switch careers or re-enter the workforce. They use the platform to search for job openings, apply to positions, and manage their applications efficiently. Recruiters, on the other hand, represent small businesses, large corporations, and staffing agencies looking to fill vacant positions. They use the platform to post job openings, manage applications, and communicate with potential candidates. The platform aims to provide

a simplified, effective, and comprehensive tool that meets the diverse needs of both groups, enabling them to connect and interact more effectively in the job market.

3.2 Key Messages:

The key messages of the CareerMap platform are designed to highlight its core value proposition to both job seekers and recruiters. For job seekers, the message is clear: "Find your next opportunity effortlessly. Explore a wide range of job openings, apply easily, and manage your career journey all in one place." This emphasizes the platform's ease of use, extensive job listings, and powerful application management tools. For recruiters, the platform promotes itself as a "streamlined solution for talent acquisition," allowing them to "post job vacancies, manage applications efficiently, and find the right candidates faster." These messages communicate the platform's commitment to providing a reliable, efficient, and user-friendly experience for both audiences, thereby enhancing the overall hiring process.

4. Project Plan:

4.1 Milestones:

The project plan for the CareerMap job board web application was structured around several critical milestones that ensured a systematic and organized development process. Each milestone represented a significant phase of the project, providing a roadmap for achieving the project's objectives efficiently and effectively. The milestones were carefully planned to allow for iterative development, regular testing, and continuous improvement throughout theproject.

The first major milestone was Backend Development. This phase focused on setting up the core backend infrastructure using Spring Boot, a popular Java framework known for its robustness and flexibility. During this stage, the backend architecture was designed to handle a variety of tasks, including user registration, login, job posting, and application management. The team implemented several RESTful API endpoints to ensure smooth communication between the server and client sides. Special attention was given to data validation, error handling, and security measures, such as encrypting passwords and protecting sensitive user data. Additionally, the backend development included integrating the application with a MySQL database, creating the necessary tables, and establishing relationships between different entities, such as users, jobs, and applications. This phase also involved testing the API endpoints to ensure they were working correctly and handling different types of requests efficiently.

The second milestone was the Frontend Development. This stage was crucial for creating a responsive and intuitive user interface that would provide a seamless user experience for both job seekers and recruiters. The frontend was built using React, a powerful JavaScript library for building user interfaces. The development process began with setting up the basic structure of the application, including creating the initial components for job search, job application, and user

profile management. The focus was on creating a user-friendly design that was both aesthetically pleasing and easy to navigate. As the development progressed, more complex components were added, such as dynamic search filters, interactive job listings, and application forms. The team also implemented state management using Redux Toolkit to ensure efficient data flow and synchronization across different components. Integration with the backend APIs was a significant part of this milestone, allowing real-time data exchange between the frontend and backend. This integration required careful planning and testing to ensure that all API calls were correctly implemented and that the data was being displayed accurately on the frontend.

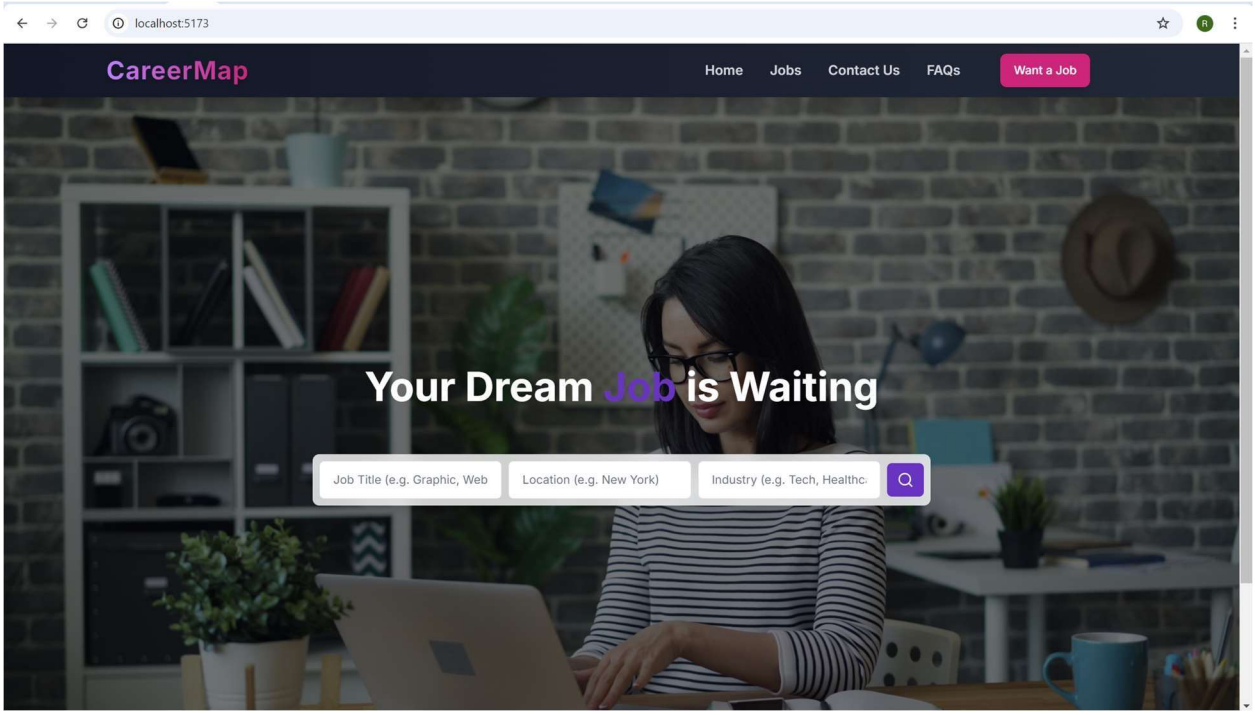
The third milestone was the Testing and Debugging Phase. This phase was essential to ensure the stability and reliability of the entire application. A comprehensive testing strategy was developed to cover all aspects of the application, including backend functionalities, frontend components, and overall user experience. Various testing methodologies were employed, such as integration testing, and user acceptance testing. tests were conducted to verify the functionality of individual components and functions, while integration tests were performed to ensure that different parts of the application worked together seamlessly. User acceptance testing was carried out to validate that the application met all user requirements and provided a satisfactory experience. During this phase, several bugs and issues were identified and resolved, such as API communication errors, data validation problems, and UI/UX inconsistencies. The testing and debugging phase also included performance testing to ensure the application could handle a large number of users and data transactions without any significant slowdowns or crashes. The feedback received during testing was used to make further improvements and optimizations, resulting in a more polished and reliable application.

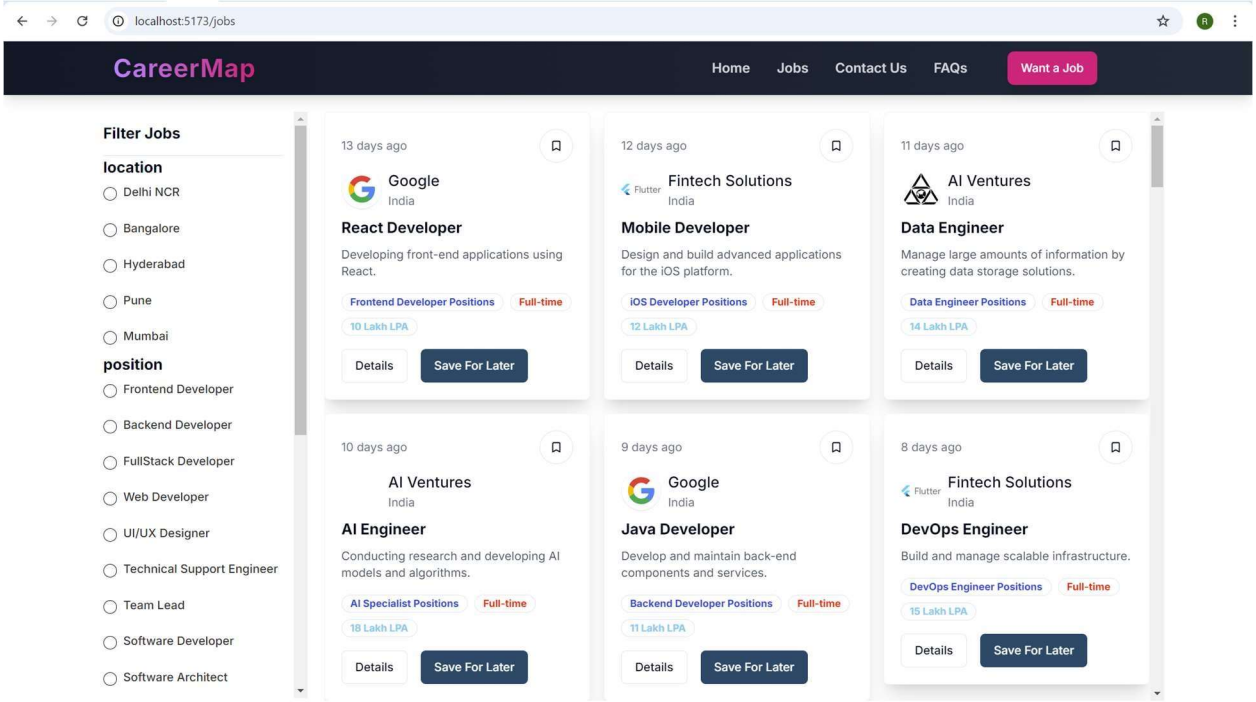
The fourth milestone involved User Authentication and Authorization. This phase focused on implementing a secure and efficient user authentication system to manage access control and protect sensitive information integrated role-based access control (RBAC) to differentiate between different types of users, such as job seekers and recruiters, ensuring that each user only had access to the features and data relevant to their role. This milestone required extensive testing to ensure that the authentication system was robust, secure, and free from vulnerabilities. It also involved creating a user-friendly registration and login process that provided a seamless experience for users while maintaining a high level of security.

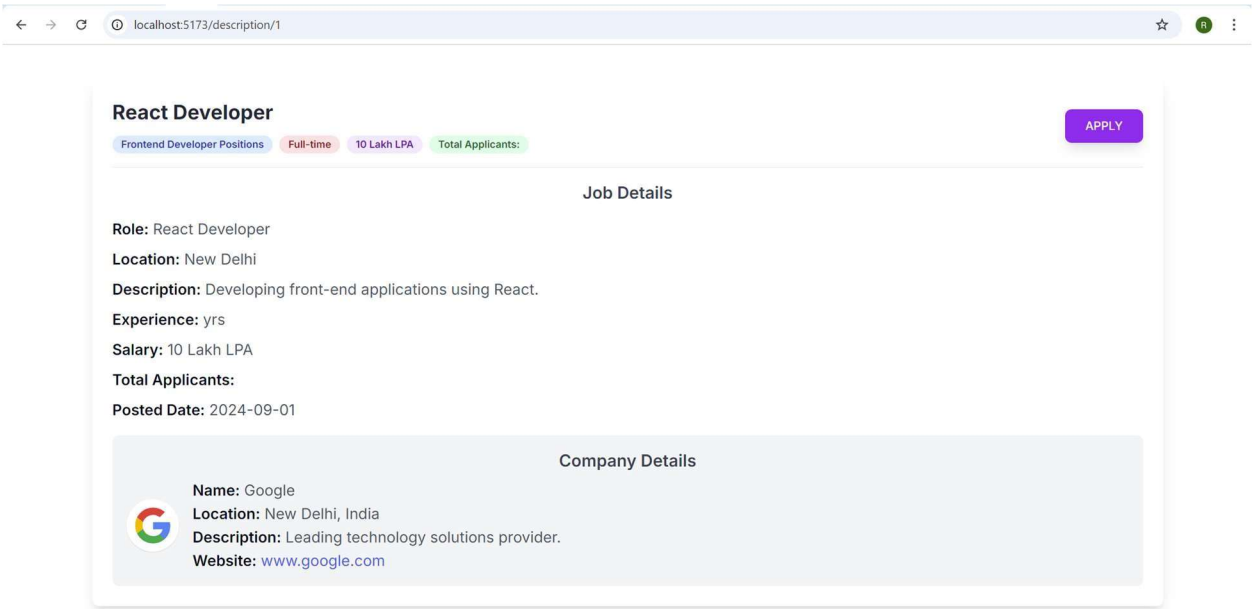
4.2 Resource Allocation:

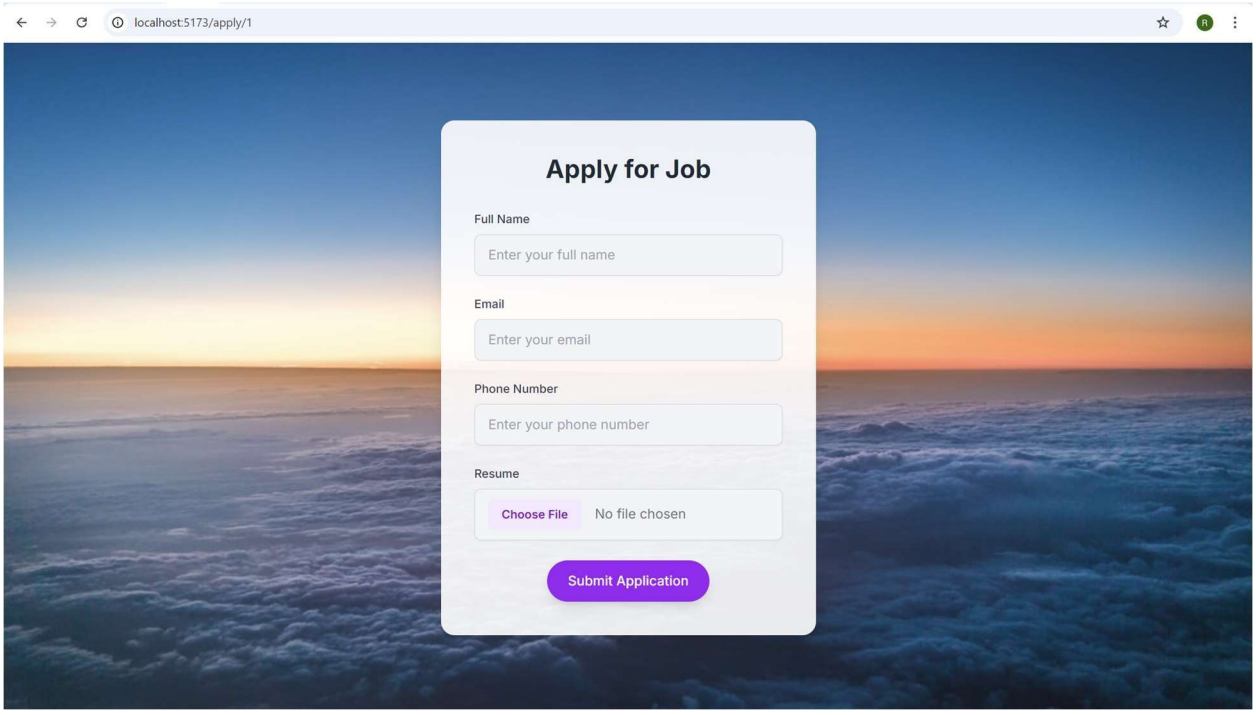
This project was entirely undertaken by me, and responsible for all aspects, including backend development, frontend design and implementation, database setup, testing, and deployment preparation. This focused approach allowed for a streamlined development process, minimizing communication overhead and enabling rapid decision-making and problem resolution and utilized various resources, including documentation, libraries, frameworks, and tools, to ensure that the application met all technical requirements and functioned as intended.

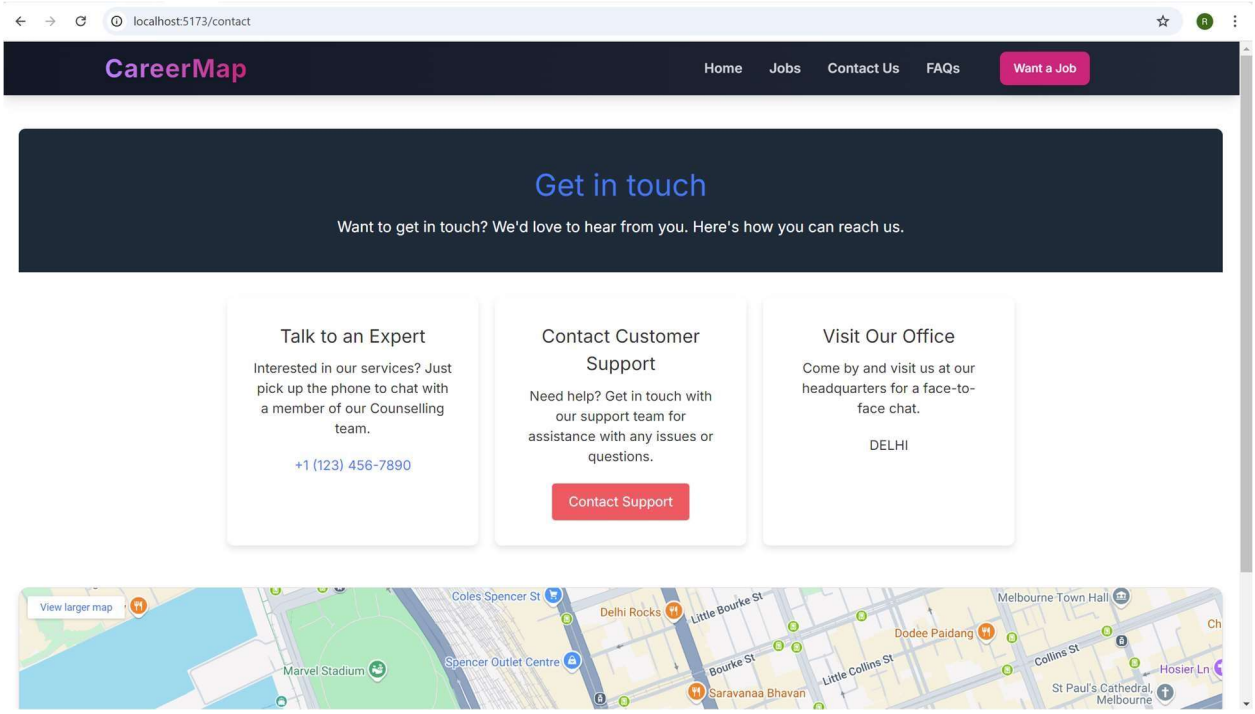
4.3 Results:

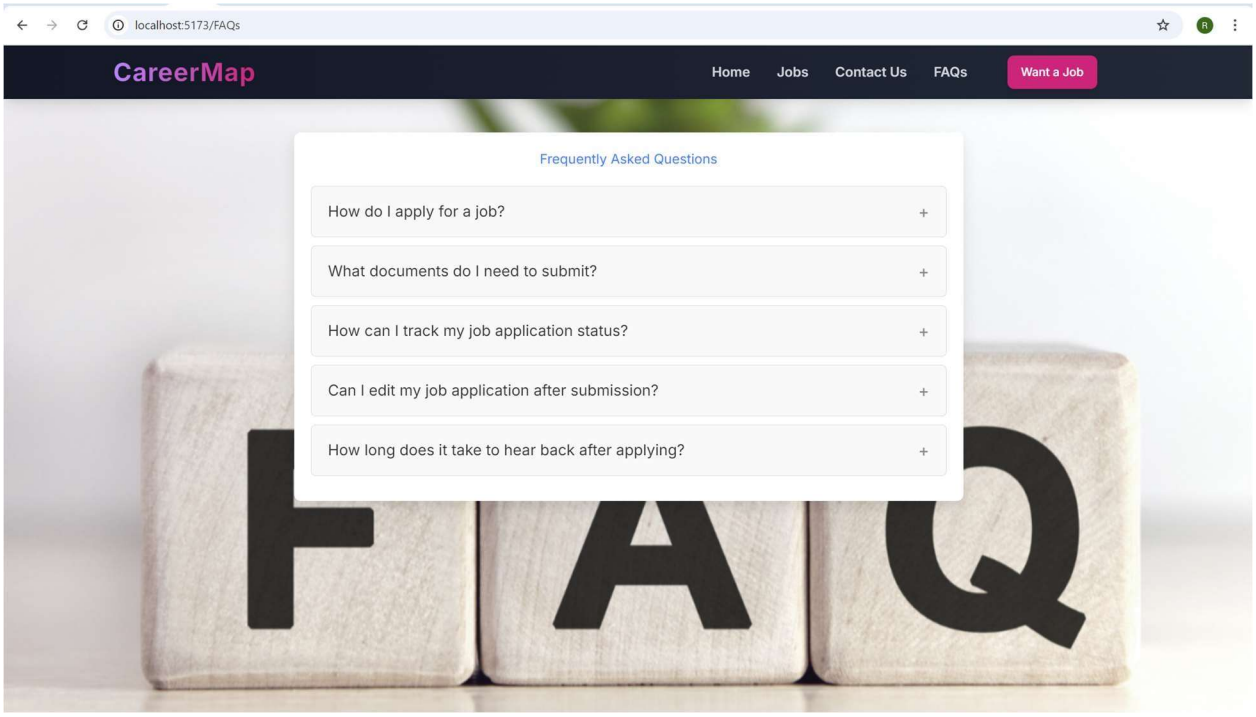

localhost:5173
CareerMap
Home
Jobs
Contact Us
FAQs
Want a Job
Your Dream Job is Waiting
Job Title (e.g. Graphic, Web
Location (e.g. New York)
Industry (e.g. Tech, Healthc

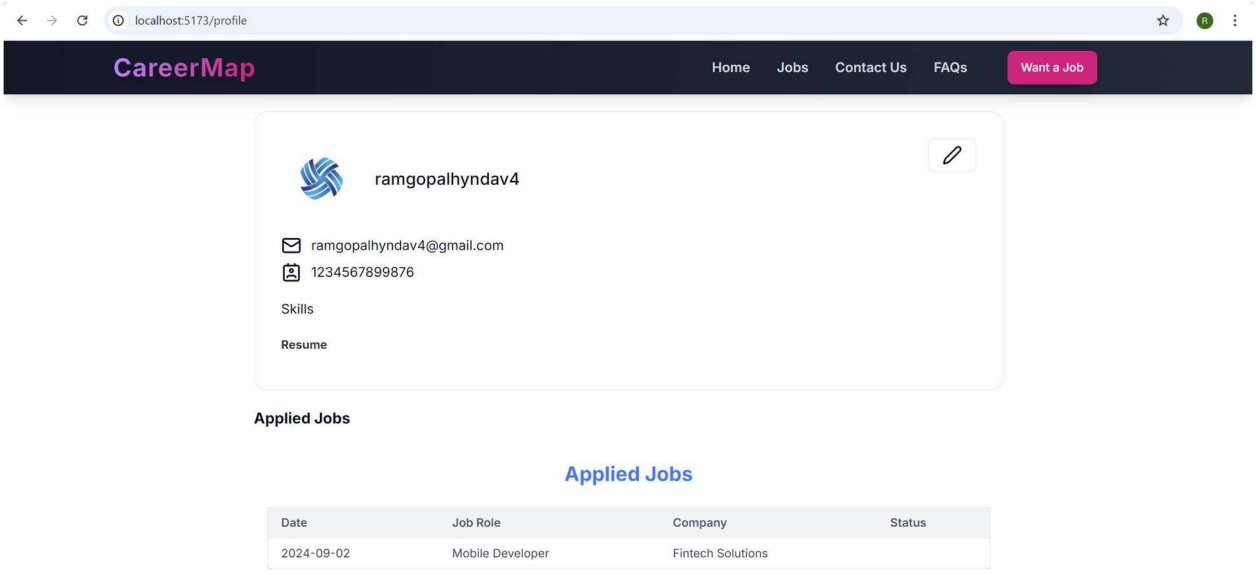

localhost:5173/jobs
CareerMap
Home
Jobs
Contact Us
FAQs
Want a Job
Filter Jobs
location
Delhi NCR
Bangalore
Hyderabad
Pune
Mumbai
position
Frontend Developer
Backend Developer
FullStack Developer
Web Developer
UI/UX Designer
Technical Support Engineer
Team Lead
Software Developer
Software Architect
13 days ago
Google
India
React Developer
Developing front-end applications using React.
Frontend Developer Positions
Full-time
10 Lakh LPA
Details
Save For Later
12 days ago
Fintech Solutions
India
Mobile Developer
Design and build advanced applications for the iOS platform.
iOS Developer Positions
Full-time
12 Lakh LPA
Details
Save For Later
11 days ago
AI Ventures
India
Data Engineer
Manage large amounts of information by creating data storage solutions.
Data Engineer Positions
Full-time
14 Lakh LPA
Details
Save For Later
10 days ago
AI Ventures
India
AI Engineer
Conducting research and developing AI models and algorithms.
AI Specialist Positions
Full-time
18 Lakh LPA
Details
Save For Later
9 days ago
Google
India
Java Developer
Develop and maintain back-end components and services.
Backend Developer Positions
Full-time
11 Lakh LPA
Details
Save For Later
8 days ago
Fintech Solutions
India
DevOps Engineer
Build and manage scalable infrastructure.
DevOps Engineer Positions
Full-time
15 Lakh LPA
Details
Save For Later

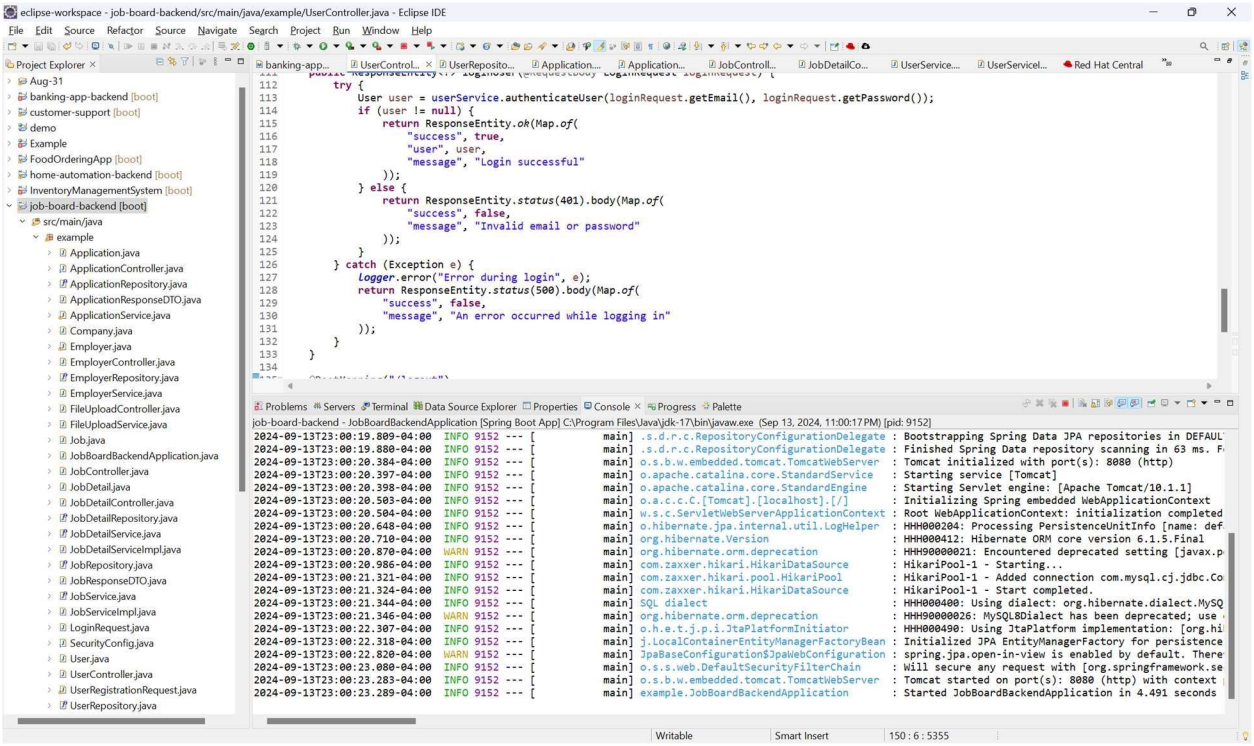

localhost:5173/description/1
React Developer
Frontend Developer Positions
Full-time
10 Lakh LPA
Total Applicants:
APPLY
Job Details
Role: React Developer
Location: New Delhi
Description: Developing front-end applications using React.
Experience: yrs
Salary: 10 Lakh LPA
Total Applicants:
Posted Date: 2024-09-01
Company Details
Name: Google
Location: New Delhi, India
Description: Leading technology solutions provider.
Website: www.google.com


localhost:5173/apply/1
Apply for Job
Full Name
Enter your full name
Email
Enter your email
Phone Number
Enter your phone number
Resume
Choose File
No file chosen
Submit Application


localhost:5173/contact
CareerMap
Home
Jobs
Contact Us
FAQs
Want a Job
Get in touch
Want to get in touch? We'd love to hear from you. Here's how you can reach us.
Talk to an Expert
Interested in our services? Just pick up the phone to chat with a member of our Counselling team.
+1 (123) 456-7890
Contact Customer Support
Need help? Get in touch with our support team for assistance with any issues or questions.
Contact Support
Visit Our Office
Come by and visit us at our headquarters for a face-to-face chat.
DELHI
View larger map
Coles Spencer St
Delhi Rocks
Little Bourke St
Melbourne Town Hall
Dodee Paidang
Marvel Stadium
Spencer Outlet Centre
Bourke St
Little Collins St
Collins St
Hosier Ln
Saravanaa Bhavan
St Paul's Cathedral, Melbourne


localhost:5173/FAQs
CareerMap
Home
Jobs
Contact Us
FAQs
Want a Job
Frequently Asked Questions
How do I apply for a job?
What documents do I need to submit?
How can I track my job application status?
Can I edit my job application after submission?
How long does it take to hear back after applying?
FAQ


localhost:5173/profile
CareerMap
Home
Jobs
Contact Us
FAQs
Want a Job
ramgopalhyndav4
ramgopalhyndav4@gmail.com
1234567899876
Skills
Resume
Applied Jobs
Applied Jobs
Date
Job Role
Company
Status
2024-09-02
Mobile Developer
Fintech Solutions


eclipse-workspace - job-board-backend/src/main/java/example/UserController.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
Aug-31
banking-app-backend [boot]
customer-support [boot]
demo
Example
FoodOrderingApp [boot]
home-automation-backend [boot]
InventoryManagementSystem [boot]
job-board-backend [boot]
src/main/java
example
Application.java
ApplicationController.java
ApplicationRepository.java
ApplicationResponseDTO.java
ApplicationService.java
Company.java
Employer.java
EmployerController.java
EmployerRepository.java
EmployerService.java
FileUploadController.java
FileUploadService.java
Job.java
JobBoardBackendApplication.java
JobController.java
JobDetail.java
JobDetailController.java
JobDetailRepository.java
JobDetailService.java
JobDetailServiceImpl.java
JobRepository.java
JobResponseDTO.java
JobService.java
JobServiceImpl.java
LoginRequest.java
SecurityConfig.java
User.java
UserController.java
UserRegistrationRequest.java
UserRepository.java
try {
User user = userService.authenticateUser(loginRequest.getEmail(), loginRequest.getPassword());
if (user != null) {
return ResponseEntity.ok(Map.of(
"success", true,
"user", user,
"message", "Login successful"
));
} else {
return ResponseEntity.status(401).body(Map.of(
"success", false,
"message", "Invalid email or password"
));
}
} catch (Exception e) {
logger.error("Error during login", e);
return ResponseEntity.status(500).body(Map.of(
"success", false,
"message", "An error occurred while logging in"
));
}
}
Problems Servers Terminal Data Source Explorer Properties Console Progress Palette
job-board-backend - JobBoardBackendApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe (Sep 13, 2024, 11:00:17 PM) [pid: 9152]
2024-09-13T23:00:19.809-04:00 INFO 9152 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAUL
2024-09-13T23:00:19.880-04:00 INFO 9152 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 63 ms. F
2024-09-13T23:00:20.384-04:00 INFO 9152 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2024-09-13T23:00:20.397-04:00 INFO 9152 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-09-13T23:00:20.398-04:00 INFO 9152 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.1]
2024-09-13T23:00:20.503-04:00 INFO 9152 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-09-13T23:00:20.504-04:00 INFO 9152 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2024-09-13T23:00:20.648-04:00 INFO 9152 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: def
2024-09-13T23:00:20.710-04:00 INFO 9152 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.1.5.Final
2024-09-13T23:00:20.870-04:00 WARN 9152 --- [main] org.hibernate.orm.deprecation : HHH90000021: Encountered deprecated setting [javax.p
2024-09-13T23:00:20.986-04:00 INFO 9152 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-09-13T23:00:21.321-04:00 INFO 9152 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.Co
2024-09-13T23:00:21.324-04:00 INFO 9152 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-09-13T23:00:21.344-04:00 INFO 9152 --- [main] SQL dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQ
2024-09-13T23:00:21.346-04:00 WARN 9152 --- [main] org.hibernate.orm.deprecation : HHH90000026: MySQL8Dialect has been deprecated; use
2024-09-13T23:00:22.307-04:00 INFO 9152 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hi
2024-09-13T23:00:22.318-04:00 INFO 9152 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
2024-09-13T23:00:22.820-04:00 WARN 9152 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2024-09-13T23:00:23.080-04:00 INFO 9152 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.se
2024-09-13T23:00:23.283-04:00 INFO 9152 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context
2024-09-13T23:00:23.289-04:00 INFO 9152 --- [main] example.JobBoardBackendApplication : Started JobBoardBackendApplication in 4.491 seconds
Writable
Smart Insert
150 : 6 : 5355

MySQL Workbench

Local instance MySQL81 - W...

File Edit View Query Database Server Tools Scripting Help

```
19        ADD COLUMN description TEXT,
20        ADD COLUMN website VARCHAR(255),
21        ADD COLUMN logo VARCHAR(255),
22        ADD COLUMN userId BIGINT,
23        ADD COLUMN createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
24        ADD COLUMN updatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;
25    ALTER TABLE application MODIFY resume BLOB;
26
27    UPDATE company
28    SET website = 'www.google.com'
29    WHERE id = 1; -- Replace '1' with the actual company ID you want to update
30
31
32    INSERT INTO company (location, name, industry, description, website, logo, userId, createdAt, updatedAt) VALUES
33    ('New Delhi, India', 'Tech Innovators', 'Technology', 'Leading technology solutions provider.', 'https://techinnovators.com', 'https://
```

Result Grid

id	applicant_name	resume	job_id	user_id
13	ramgopalhyndav	BLOB	2	1
14	ramgopalhyndav	BLOB	3	1
15	ramgopalhyndavtest-15	BLOB	50	1
16	ramgopalhyndav	BLOB	2	1
17	ramgopalhyndav-test20	BLOB	2	1
18	ramgopalhyndavtesting21	BLOB	1	1
19	ramgopalhyndav	BLOB	2	4
20	ramgopalhyndav-test8	BLOB	1	1

Output

#	Time	Action	Message	Duration / Fetch
17	16:00:34	select * from application	19 row(s) returned	0.000 sec / 0.000 sec
18	16:39:25	select * from application	20 row(s) returned	0.000 sec / 0.000 sec
19	16:39:44	select * from application	21 row(s) returned	0.000 sec / 0.000 sec
20	16:47:35	SELECT * FROM job.company	8 row(s) returned	0.000 sec / 0.000 sec
21	16:53:41	SELECT * FROM job.company	8 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

Local instance MySQL81 - W...

File Edit View Query Database Server Tools Scripting Help

```
1    SELECT * FROM job.company;
```

Result Grid

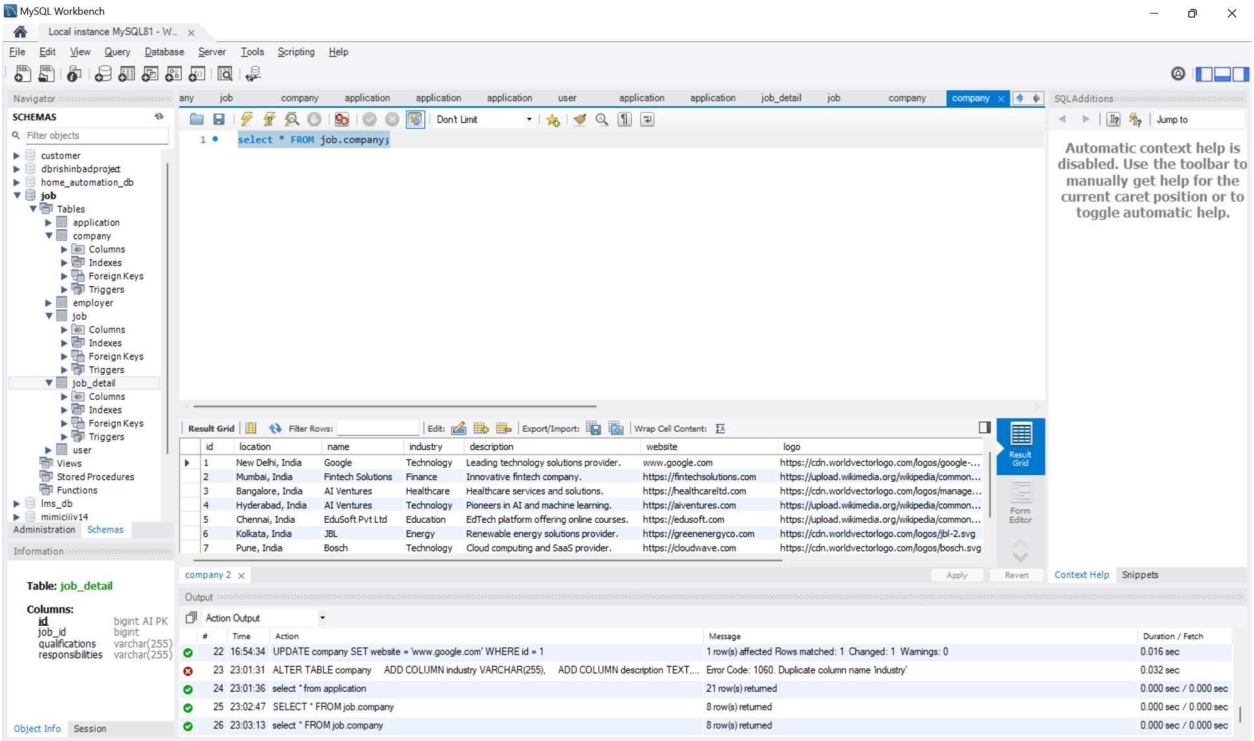
id	location	name	industry	description	website	logo
1	New Delhi, India	Google	Technology	Leading technology solutions provider.	www.google.com	https://cdn.worldvectorlogo.com/logos/google-...
2	Mumbai, India	Fintech Solutions	Finance	Innovative fintech company.	https://fintechsolutions.com	https://upload.wikimedia.org/wikipedia/common...
3	Bangalore, India	AI Ventures	Healthcare	Healthcare services and solutions.	https://healthcareltd.com	https://cdn.worldvectorlogo.com/logos/manage...
4	Hyderabad, India	AI Ventures	Technology	Pioneers in AI and machine learning.	https://aiventures.com	https://upload.wikimedia.org/wikipedia/common...
5	Chennai, India	EduSoft Pvt Ltd	Education	EdTech platform offering online courses.	https://edusoft.com	https://upload.wikimedia.org/wikipedia/common...
6	Kolkata, India	JBL	Energy	Renewable energy solutions provider.	https://greenenergyco.com	https://cdn.worldvectorlogo.com/logos/jbl-2.svg
7	Pune, India	Bosch	Technology	Cloud computing and SaaS provider.	https://cloudwave.com	https://cdn.worldvectorlogo.com/logos/bosch.svg

Table: company

Columns:
id bigint AI PK
location varchar(255)
name varchar(255)
industry varchar(255)
description text
website varchar(255)
logo varchar(255)
userId bigint
createdAt timestamp

Output

#	Time	Action	Message	Duration / Fetch
21	16:53:41	SELECT * FROM job.company	8 row(s) returned	0.000 sec / 0.000 sec
22	16:54:34	UPDATE company SET website = 'www.google.com' WHERE id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
23	23:01:31	ALTER TABLE company ADD COLUMN industry VARCHAR(255), ADD COLUMN description TEXT,...	Error Code: 1060. Duplicate column name 'industry'	0.032 sec
24	23:01:36	select * from application	21 row(s) returned	0.000 sec / 0.000 sec
25	23:02:47	SELECT * FROM job.company	8 row(s) returned	0.000 sec / 0.000 sec



5. Project Status Reports:

5.1 Format:

The project status reports for the CareerMap job board web application are designed to provide detailed and structured updates on the progress of the project. Each report begins with an overview of the current status, highlighting key activities and overall progress. This is followed by a section on completed tasks, which outlines what has been achieved since the last report, such as backend development, database integration, and initial frontend setup. The ongoing tasks section details the work currently in progress, such as frontend-backend integration, user interface enhancements, and continuous testing and debugging. The pending tasks section outlines future work, including finalizing user authentication, enhancing the UI/UX, and preparing the application for deployment. The report also includes a section on issues and challenges, describing any obstacles faced and the strategies employed to overcome them. Finally, the report details any changes to the project timeline and outlines the next steps to be taken.

6. Final Deliverables and Closure Report:

6.1 Summary of Outcomes:

The CareerMap job board web application project successfully achieved its primary objectives within the allocated 10-hour timeframe. The backend API was fully developed using Spring Boot, providing all necessary functionalities for user management, job postings, and application processing. The MySQL database was effectively integrated to handle data storage and retrieval, ensuring quick and secure transactions. The frontend, built with React, offers a responsive and user-friendly interface that allows job seekers and recruiters to interact with the platform seamlessly. While the initial version does not yet support job posting from the frontend, this functionality is planned for future updates. Rigorous testing confirmed that all implemented features operate as expected, providing a stable foundation for the next phase of development.

6.2 Final Code:

ApplicationController:

```
package example;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.util.List;
import java.util.stream.Collectors;


@CrossOrigin(origins = "http://localhost:5173")

@RestController

@RequestMapping("/api/applications") public class ApplicationController {


  @Autowired private ApplicationService applicationService;


  @Autowired private JobService jobService;
```

```
@Autowired private UserService userService; // Add this line to inject the UserService

@GetMapping("/user/{userId}/applications")
public ResponseEntity<List<ApplicationResponseDTO>> getApplicationsByUserId(@PathVariable Long userId) {
    try {
        List<Application> applications = applicationService.getApplicationsByUserId(userId);

        if (applications == null || applications.isEmpty()) { return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }

        // Convert Application entities to DTOs
        List<ApplicationResponseDTO> applicationDTOs = applications.stream()
            .map(application -> new ApplicationResponseDTO(
                application.getId(), application.getApplicantName(), application.getResume(), application.getJob() != null ? application.getJob().getTitle() : "N/A",
                application.getJob() != null ? application.getJob().getCompany().getName() : "N/A", application.getJob() != null ? application.getJob().getCreatedAt() : null))
            .collect(Collectors.toList());
```

```
        return new ResponseEntity<>(applicationDTOs, HttpStatus.OK);

    } catch (Exception e) {

        System.err.println("Error fetching applications: " + e.getMessage());            return new
ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }

  }



  @GetMapping("/{id}")

  public        ResponseEntity<ApplicationResponseDTO>   getApplicationById(@PathVariable
Long id) {

    Application  application  =  applicationService.getApplicationById(id);                if
(application != null) {
        Job job = application.getJob();

        ApplicationResponseDTO                        applicationDTO                =                new
ApplicationResponseDTO(

            application.getId(),              application.getApplicantName(),
            null,              job != null ? job.getTitle() :
"N/A",                                  job   !=   null   ?
job.getCompany().getName() : "N/A",              job
!= null ? job.getCreatedAt() : null
        );

        return new ResponseEntity<>(applicationDTO, HttpStatus.OK);

    } else {

        return new ResponseEntity<>(HttpStatus.NOT_FOUND);

    }

  }
```

```
    @GetMapping("/resume/{id}") public ResponseEntity<byte[]> downloadResume(@PathVariable Long id) { Application application = applicationService.getApplicationById(id); if (application != null && application.getResume() != null) { HttpHeaders headers = new HttpHeaders(); headers.setContentType(MediaType.APPLICATION_PDF); headers.setContentDispositionFormData("attachment", "resume_" + application.getId() + ".pdf");

        return new ResponseEntity<>(application.getResume(), headers, HttpStatus.OK);

    } else {

        return new ResponseEntity<>(HttpStatus.NOT_FOUND);

    }

  }


    @PostMapping("/apply/{jobId}") public ResponseEntity<?> createApplication(
        @PathVariable("jobId") Long jobId,

        @RequestParam("applicantName") String applicantName,

        @RequestParam("resume") MultipartFile resumeFile,

        @RequestParam("userEmail") String userEmail,

        @RequestParam("userId") Long userId) {


    if (jobId == null) {

        return new ResponseEntity<>("Job ID is required.",
```

```
HttpStatus.BAD_REQUEST);

        }


        if (resumeFile == null || resumeFile.isEmpty()) {

            return new ResponseEntity<>("Resume file is required.",
HttpStatus.BAD_REQUEST);

        }

         try {
            byte[] resumeData = resumeFile.getBytes();



            Job job = jobService.getJobById(jobId);            if
(job == null) {
```

return new ResponseEntity<>("Job not found.", HttpStatus.NOT_FOUND);

}

Application application = new Application(); application.setApplicantName(applicantName); application.setJob(job); application.setResume(resumeData);

// Set the user for this application using userId User user = userService.findById(userId); if (user == null) {

return new ResponseEntity<>("User not found.", HttpStatus.NOT_FOUND);

}

application.setUser(user);

applicationService.saveApplication(application, userEmail);

int totalApplicants = applicationService.countApplicationsByJobId(jobId); job.setApplicationsCount(totalApplicants); jobService.saveJob(job);

```
        Job updatedJob = jobService.getJobById(jobId);

        return ResponseEntity.status(HttpStatus.CREATED).body(updatedJob);

    } catch (IOException e) {
        return new ResponseEntity<>("Failed to upload file.",
HttpStatus.INTERNAL_SERVER_ERROR);
    } catch (IllegalArgumentException e) {
        return new ResponseEntity<>(e.getMessage(),
HttpStatus.BAD_REQUEST);
    } catch (Exception e) {
        return new ResponseEntity<>("An unexpected error occurred.",
HttpStatus.INTERNAL_SERVER_ERROR);
    }
  }
}
```

JobController: package example;

```
import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.http.HttpStatus; import
org.springframework.http.ResponseEntity; import
org.springframework.web.bind.annotation.*;
```

```
import java.util.List;


@CrossOrigin(origins = "http://localhost:5173")

@RestController

@RequestMapping("/api/jobs")  public  class
JobController {


   @Autowired

   private JobService jobService;



   @GetMapping              public   ResponseEntity<List<JobResponseDTO>>
getAllJobs() {          List<JobResponseDTO> jobs = jobService.getAllJobs();
return new ResponseEntity<>(jobs, HttpStatus.OK);
   }



   @GetMapping("/{id}")                              public       ResponseEntity<JobResponseDTO>
getJobById(@PathVariable Long id) {
       Job job = jobService.getJobById(id);           if
(job != null) {
           return new ResponseEntity<>(new JobResponseDTO(job), HttpStatus.OK);

       } else {

           return new ResponseEntity<>(HttpStatus.NOT_FOUND); // Return 404 if not found

       }

   }
```

```
    @PostMapping

    public ResponseEntity<Job> createJob(@RequestBody Job job) {            Job
createdJob    =    jobService.saveJob(job);                                return    new
ResponseEntity<>(createdJob, HttpStatus.CREATED);
    }



    @PutMapping("/{id}")        public ResponseEntity<Job> updateJob(@PathVariable Long id,
@RequestBody Job job) {        job.setId(id);
        Job updatedJob = jobService.saveJob(job);                    return new
ResponseEntity<>(updatedJob, HttpStatus.OK);
    }



    @DeleteMapping("/{id}")                        public      ResponseEntity<Void>
deleteJob(@PathVariable Long id) {          jobService.deleteJob(id);          return
new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }

}
```

JobDetailController:

```
package                                example;                                import
org.springframework.beans.factory.annotation.Autowired;                import
org.springframework.web.bind.annotation.*;


import java.util.List;



@RestController

@RequestMapping("/api/job-details")  public  class
JobDetailController {
```

```
    @Autowired private JobDetailService jobDetailService;

    @GetMapping public List<JobDetail> getAllJobDetails() { return jobDetailService.getAllJobDetails();
    }

    @GetMapping("/{id}") public JobDetail getJobDetailById(@PathVariable Long id) { return jobDetailService.getJobDetailById(id);

    }

    @PostMapping public JobDetail createJobDetail(@RequestBody JobDetail jobDetail) { return jobDetailService.saveJobDetail(jobDetail);
    }

    @PutMapping("/{id}") public JobDetail updateJobDetail(@PathVariable Long id, @RequestBody JobDetail jobDetail) { jobDetail.setId(id); return jobDetailService.saveJobDetail(jobDetail);
    }

    @DeleteMapping("/{id}") public void deleteJobDetail(@PathVariable Long id) { jobDetailService.deleteJobDetail(id);
    }

}
```

UserController: package example;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.*; import org.springframework.web.multipart.MultipartFile; import org.springframework.security.crypto.password.PasswordEncoder; import org.slf4j.Logger; import org.slf4j.LoggerFactory;

import java.util.Map;

@RestController

@RequestMapping("/api/users") public class UserController {

@Autowired

private UserService userService;

@Autowired private PasswordEncoder passwordEncoder;

@Autowired private FileUploadService fileUploadService;

private static final Logger logger = LoggerFactory.getLogger(UserController.class);

```
@PostMapping("/register")
public ResponseEntity<?> registerUser(
    @RequestParam String fullname,

    @RequestParam String email,

    @RequestParam String phoneNumber,

    @RequestParam String password,

    @RequestParam String role,

    @RequestParam(required = false) MultipartFile file) {


    try {

        // Check if the email is already in use
        if (userService.existsByEmail(email)) {
            return ResponseEntity.badRequest().body(Map.of(
                "success", false,

                "message", "Email is already in use"

            ));

        }



        // Handle file upload using the service
        String filePath = null;
        if (file != null && !file.isEmpty()) {
            filePath = fileUploadService.saveFile(file);
        }



        // Create and save the user
        User user = new User();
        user.setFullname(fullname);
        user.setEmail(email);
        user.setPhoneNumber(phoneNumber);
```

```
user.setPassword(passwordEncoder.encode(password
));                                    user.setRole(role);
user.setProfilePicture(filePath);


        User savedUser = userService.saveUser(user);



        // Ensure user is saved correctly            if (savedUser.getId()
== null) {              return ResponseEntity.status(500).body(Map.of(
                "success", false,

                "message", "User could not be saved"

            ));

        }



        return ResponseEntity.ok(Map.of(

            "success", true,

            "user", savedUser,

            "message", "User registered successfully"

        ));

    } catch (Exception e) {

        logger.error("Error registering user", e);                return
ResponseEntity.status(500).body(Map.of(
            "success", false,

            "message", "An error occurred while registering the user"

        ));
```

```
    }

  }



  @GetMapping("/email/{email}")                    public    ResponseEntity<?>
getUserByEmail(@PathVariable String email) {
    try {

        User user = userService.getUserByEmail(email);

        if (user != null) {                    return
ResponseEntity.ok(Map.of(
            "success", true,

            "user", user

        ));
} else {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(Map.of(

            "success", false,

            "message", "User not found"

        ));

        }

    } catch (Exception e) {          logger.error("Error fetching
user by email", e);
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(Map.of(

        "success", false,

        "message", "An error occurred while fetching the user"

        ));
```

```
    }

  }



  @PostMapping("/login") public ResponseEntity<?> loginUser(@RequestBody
LoginRequest loginRequest) {
    try {

        User user = userService.authenticateUser(loginRequest.getEmail(),
loginRequest.getPassword());
        if (user != null) { return
ResponseEntity.ok(Map.of( "success",
true,

            "user", user,

            "message", "Login successful"

          ));

        } else { return
ResponseEntity.status(401).body(Map.of(
            "success", false,

            "message", "Invalid email or password"

          ));

        }

    } catch (Exception e) { logger.error("Error during
login", e); return ResponseEntity.status(500).body(Map.of(
          "success", false,

          "message", "An error occurred while logging in"

        ));
```

```
        }

    }



    @PostMapping("/logout")

    public ResponseEntity<?> logoutUser() {

        try {

            // Implement logout logic, such as invalidating the session or clearing cookies            return
ResponseEntity.ok(Map.of(
                "success", true,

                "message", "Logout successful"

            ));

        } catch (Exception e) {                logger.error("Error during
logout",     e);                                                    return
ResponseEntity.status(500).body(Map.of(
                "success", false,

                "message", "An error occurred while logging out"

            ));

        }

    }

}
```

REACT:

JobDescription:

```
import React, { useEffect } from 'react'; import { Badge
} from './ui/badge'; import { Button } from './ui/button';
import { useParams, useNavigate } from 'react-router-
```

```
dom'; import axios from 'axios'; import {
JOB_API_END_POINT } from '@/utils/constant';
import { setSingleJob } from '@/redux/jobSlice';
import { useDispatch, useSelector } from 'react-redux';
import { toast } from 'sonner';


const JobDescription = () => {      const { singleJob } =
useSelector((store) => store.job);        const { user } =
useSelector((store) => store.auth);        const dispatch =
useDispatch();   const navigate = useNavigate();


  const { id: jobId } = useParams();



  const fetchJobDetails = async () => {    console.log('Fetching
job details for ID:', jobId);     if (!jobId) return;

    try {

      const res = await axios.get(`${JOB_API_END_POINT}/${jobId}`, { withCredentials: true });
if (res.data && res.status === 200) {        dispatch(setSingleJob(res.data));
      } else {            toast.error('Failed to fetch job
details');
      }

    } catch (error) {       console.error('Error fetching
job:', error);            toast.error('Error fetching job
details');
    }

  };



  const applyJobHandler = () => {

    if (!user) {       toast.error('Please log in to apply for this
job.');      return;
```

```
  }

  navigate(`/apply/${jobId}`);

 };



 useEffect(() => {     fetchJobDetails();
 }, [jobId, dispatch]);



 return (

  <div className="max-w-7xl mx-auto my-10 p-6 bg-white shadow-lg roundedlg">

   <div className="flex items-center justify-between border-b pb-4 mb-4">        <div>
     <h1 className="text-2xl font-bold text-gray-800">{singleJob?.title}</h1>

     <div className="flex items-center gap-2 mt-3">

       <Badge className="bg-blue-100 text-blue-800">{singleJob?.position}
Positions</Badge>

                                        <Badge          className="bg-red-100          text-red-
800">{singleJob?.jobType}</Badge>

       <Badge className="bg-purple-100 text-purple-800">{singleJob?.salary} LPA</Badge>

       <Badge className="bg-green-100 text-green-800">Total Applicants:
{singleJob?.applications?.length}</Badge>

     </div>

    </div>

    <Button          onClick={applyJobHandler}          className="rounded-lg bg-purple-600
hover:bg-purple-700 text-white shadow-lg px-6 py-2 transition duration-300"
    >
```

```
        APPLY

      </Button>

    </div>



    <div className="space-y-4">

      <h2 className="text-xl font-semibold text-gray-700">Job Details</h2>

      <div className="space-y-2">

        <p className="text-lg"><span className="font-semibold">Role:</span>
<span className="text-gray-600">{singleJob?.title}</span></p>

                                        <p       className="text-lg"><span
className="fontsemibold">Location:</span>         <span  className="text-gray-
600">{singleJob?.location}</span></p>

                                        <p       className="text-lg"><span
className="fontsemibold">Description:</span>     <span  className="text-gray-
600">{singleJob?.description}</span></p>

                                        <p       className="text-lg"><span
className="fontsemibold">Experience:</span>     <span  className="text-gray-
600">{singleJob?.experience} yrs</span></p>

        <p     className="text-lg"><span     className="font-semibold">Salary:</span>     <span
className="text-gray-600">{singleJob?.salary} LPA</span></p>

        <p className="text-lg"><span className="font-semibold">Total
Applicants:</span>                          <span                          className="text-gray-
600">{singleJob?.applications?.length}</span></p>

        <p className="text-lg"><span className="font-semibold">Posted
Date:</span>                          <span                          className="text-gray-
600">{singleJob?.createdAt?.split('T')[0]}</span></p>
```

```
</div>

{/* Display Company Details */}
{singleJob?.company && (
  <div className="mt-6 p-4 bg-gray-100 rounded-lg">
    <h2 className="text-xl font-semibold text-gray-700">Company Details</h2>
    <div className="flex items-center mt-2">
      {singleJob?.company?.logo && (
        <img src={singleJob.company.logo} alt="Company Logo" className="h-16 w-16 rounded-full mr-4" />
      )}
      <div>
                <p className="text-lg"><span className="fontsemibold">Name:</span> <span className="text-gray-600">{singleJob?.company?.name}</span></p>
                <p className="text-lg"><span className="fontsemibold">Location:</span> <span className="text-gray-600">{singleJob?.company?.location}</span></p>
                <p className="text-lg"><span className="fontsemibold">Description:</span> <span className="text-gray-600">{singleJob?.company?.description}</span></p>
        {singleJob?.company?.website && (
                <p className="text-lg"><span className="font-
```

```
semibold">Website:</span> <a href={singleJob.company.website} target="_blank" rel="noopener noreferrer" className="text-blue-600 hover:underline">{singleJob.company.website}</a></p>

            )}

          </div>

        </div>

      </div>

    )}

  </div>

 </div>

);

};



export default JobDescription;
```

Profile:

```
import React, { useState } from 'react' import Navbar from './shared/Navbar' import { Avatar, AvatarImage } from './ui/avatar' import { Button } from './ui/button' import { Contact, Mail, Pen } from 'lucide-react' import { Badge } from './ui/badge' import { Label } from './ui/label' import AppliedJobTable from './AppliedJobTable' import UpdateProfileDialog from './UpdateProfileDialog' import { useSelector } from 'react-redux' import useGetAppliedJobs from '@/hooks/useGetAppliedJobs'


// const skills = ["Html", "Css", "Javascript", "Reactjs"]
```

```
const isResume = true;



const Profile = () => {     useGetAppliedJobs();     const
[open, setOpen] = useState(false);        const {user} =
useSelector(store=>store.auth);


   return (

      <div>

         <Navbar />

         <div className='max-w-4xl mx-auto bg-white border border-gray-200 rounded-2xl my-5
p-8'>

            <div className='flex justify-between'>

               <div className='flex items-center gap-4'>

                  <Avatar className="h-24 w-24">

                     <AvatarImage        src="https://www.shutterstock.com/imagevector/circle-line-
simple-design-logo-600nw-2174926871.jpg" alt="profile" />

                  </Avatar>

                  <div>

                     <h1 className='font-medium text-xl'>{user?.fullname}</h1>

                     <p>{user?.profile?.bio}</p>

                  </div>

               </div>

               <Button        onClick={()        =>        setOpen(true)}        className="text-right"
variant="outline"><Pen /></Button>

            </div>
```

```
<div className='my-5'>
    <div className='flex items-center gap-3 my-2'>
        <Mail />
        <span>{user?.email}</span>
    </div>
    <div className='flex items-center gap-3 my-2'>
        <Contact />
        <span>{user?.phoneNumber}</span>
    </div>
</div>
<div className='my-5'>
    <h1>Skills</h1>
    <div className='flex items-center gap-1'>
        {
            user?.profile?.skills.length !== 0 ? user?.profile?.skills.map((item, index) =>
<Badge key={index}>{item}</Badge>) : <span>NA</span>
        }
    </div>
</div>
<div className='grid w-full max-w-sm items-center gap-1.5'>
    <Label className="text-md font-bold">Resume</Label>
```

```
                {
                    isResume ? <a target='blank' href={user?.profile?.resume}
className='text-blue-500 w-full hover:underline cursor-
pointer'>{user?.profile?.resumeOriginalName}</a> : <span>NA</span>
                }
            </div>
        </div>
        <div className='max-w-4xl mx-auto bg-white rounded-2xl'>
            <h1 className='font-bold text-lg my-5'>Applied Jobs</h1>
            {/* Applied Job Table   */}
            <AppliedJobTable />
        </div>
        <UpdateProfileDialog open={open} setOpen={setOpen}/>
    </div>
  )
}


export default Profile
```

HeroSection:

```
import React, { useState } from 'react'; import { Button } from
'./ui/button'; import { Search } from 'lucide-react'; import {
useDispatch } from 'react-redux'; import { setSearchedQuery }
```

```
from '@/redux/jobSlice'; import { useNavigate } from 'react-
router-dom';


const HeroSection = () => {     const [query, setQuery]
= useState("");         const [location, setLocation] =
useState("");           const [industry, setIndustry] =
useState("");    const dispatch = useDispatch();    const
navigate = useNavigate();


   const searchJobHandler = () => {

      // Combine the search parameters into a single string for dispatch
const searchQuery = `query:${query} location:${location}
industry:${industry}`.trim();



      // Dispatch the search query
dispatch(setSearchedQuery(searchQuery));        navigate("/jobs");
   };



   return (        <div        className='relative bg-center text-
white h-screen'          style={{
            backgroundImage:          `url('https://images.unsplash.com/photo1713947506663-
7f630ef496ba?q=80&w=2232&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3
D%3D')`,

            backgroundSize: 'cover',          backgroundPosition:
'center',          backgroundRepeat: 'no-repeat'
         }}

      >

         <div className='absolute inset-0 bg-black opacity-50'></div> {/* Overlay */}

         <div className='relative z-10 flex flex-col items-center justify-center h-full text-center'>
```

```
<h1 className='text-5xl font-bold leading-snug mb-4'>

  Your Dream <span className='text-[#6A38C2]'>Job</span> is Waiting

</h1>

{/* Enhanced Search Bar */}

<div className='relative w-full md:w-[70%] lg:w-[60%] xl:w-[50%] mxauto mt-8'>

  <div className='flex items-center gap-2 p-2 bg-white rounded-lg shadow-lg backdrop-blur-md bg-opacity-80'>

    <input type="text" placeholder='Job Title (e.g. Graphic, Web Developer)' onChange={(e) => setQuery(e.target.value)} className='w-1/3 py-3 px-4 text-sm text-gray-800 bg-white outline-none rounded-md placeholder-gray-500 focus:ring-2 focus:ring-[#6A38C2] transition-colors duration-300 ease-in-out'

    />

    <input

      type="text" placeholder='Location (e.g. New York)' onChange={(e) => setLocation(e.target.value)} className='w-1/3 py-3 px-4 text-sm text-gray-800 bg-white outline-none rounded-md placeholder-gray-500 focus:ring-2 focus:ring-[#6A38C2] transition-colors duration-300 ease-in-out'

    /> <input type="text" placeholder='Industry (e.g. Tech, Healthcare)' onChange={(e) => setIndustry(e.target.value)} className='w-1/3 py-3 px-4 text-sm text-gray-800 bg-white outline-none rounded-md placeholder-gray-500 focus:ring-2 focus:ring-[#6A38C2] transition-colors duration-300 ease-in-out'

    />

    <Button onClick={searchJobHandler} className="bg-[#6A38C2] hover:bg-purple-700 p-3 rounded-md text-white transition-all duration-300 flex items-center justify-center"

    >

      <Search className='h-5 w-5' />

    </Button>
```

```
                </div>

              </div>

            </div>

        </div>

    );

};

export default HeroSection;
```

Jobs:

```
import React, { useEffect, useState } from 'react'; import Navbar from
'./shared/Navbar'; import FilterCard from './FilterCard'; import Job
from './Job'; import { useSelector } from 'react-redux'; import { motion
} from 'framer-motion'; import axios from 'axios'; import {
JOB_API_END_POINT } from '@/utils/constant'; const Jobs = () =>
{  const { searchedQuery } = useSelector((store) => store.job);  const
[jobs, setJobs] = useState([]);    const [filterJobs, setFilterJobs] =
useState([]);


  // Fetching jobs from API  useEffect(()
=> {    const fetchJobs = async () => {
     try {

       const response = await axios.get(`${JOB_API_END_POINT}`);
console.log('Fetched jobs:', response.data);        if (Array.isArray(response.data))
{          setJobs(response.data);          setFilterJobs(response.data);
       } else {                console.error('Unexpected API response
format');
       }

     } catch (error) {          console.error('Error fetching
jobs:', error);
     }
```

```
    };

    fetchJobs();

  }, []);

  // Filter jobs based on the search query   useEffect(() => {
console.log('Searched  Query:',  searchedQuery);          let
filteredJobs = jobs;


    if (searchedQuery) {

      // Improved regex to handle multiple key-value pairs correctly        const
queryObj = Object.fromEntries(         searchedQuery
          .split(' ')

          .map((item) => item.split(/:(.+)/).map((str) => str.trim()))

      );

      console.log('Query Object:', queryObj);

      const queryLocation = queryObj?.location?.toLowerCase();        const
queryPosition = queryObj?.position?.toLowerCase();      const querySalary
=      queryObj?.salary;                                const      queryJobType      =
queryObj?.job_type?.toLowerCase();
      // Apply filters      filteredJobs = jobs.filter((job) => {       const matchLocation = queryLocation
? job.location?.toLowerCase() === queryLocation : true;
                                        const            matchPosition            =            queryPosition            ?
job.position?.toLowerCase().includes(queryPosition) : true;        const matchSalary = querySalary
? isSalaryInRange(job.salary, querySalary) : true;
                                        const            matchJobType            =            queryJobType            ?
job.jobType?.toLowerCase().includes(queryJobType) : true;



        return matchLocation && matchPosition && matchSalary && matchJobType;
```

```
    });

    console.log('Filtered Jobs:', filteredJobs);

  }

  setFilterJobs(filteredJobs);

}, [searchedQuery, jobs]);

const isSalaryInRange = (jobSalary, queryRange) => { const jobSalaryNumeric = extractSalaryValue(jobSalary); const [minRange, maxRange] = extractSalaryRange(queryRange);


  if (minRange === null || maxRange === null || jobSalaryNumeric === null) {

    return false;

  }

  return jobSalaryNumeric >= minRange && jobSalaryNumeric <= maxRange;

};



const extractSalaryValue = (salaryString) => { const salaryMatch = salaryString.match(/(\d+)\s*Lakh/); return salaryMatch ? parseInt(salaryMatch[1]) * 100000 : null;
};

const extractSalaryRange = (rangeString) => { if (rangeString.includes('Above')) { const min = parseInt(rangeString.match(/(\d+)\s*Lakh/)[1]); return [min * 100000, Infinity];
  }

  const rangeMatch = rangeString.match(/(\d+)\s*Lakh.*?(\d+)\s*Lakh/); return rangeMatch ? [parseInt(rangeMatch[1]) * 100000, parseInt(rangeMatch[2]) * 100000] : [null, null];
};
```

```
  return (

    <div>

      <Navbar />

      <div className='max-w-7xl mx-auto mt-5'>

        <div className='flex gap-5'>

          <div className='w-1/5'>

            <FilterCard />

          </div>

          {filterJobs.length <= 0 ? (

            <span>No jobs found</span>

          ) : (

            <div className='flex-1 h-[88vh] overflow-y-auto pb-5'>

              <div className='grid grid-cols-3 gap-4'>

                {filterJobs.map((job) => (

                  <motion.div                    initial={{
opacity: 0, x: 100 }}                    animate={{
opacity: 1, x: 0 }}                    exit={{ opacity: 0,
x: -100 }}                    transition={{ duration: 0.3
}}                    key={job.id}
                  >

                    <Job job={job} />

                  </motion.div>

                ))}
```

```
            </div>
          </div>
        )}
      </div>
    </div>
  </div>
);
};


export default Jobs;
```

7. References:

Spring Boot Documentation:

Used as a primary reference for creating and managing the backend RESTful APIs, handling HTTP requests, and integrating backend services.

Spring Boot Official Documentation

MySQL Documentation:

Referred for designing the relational database schema, managing database operations, and integrating MySQL with the Spring Boot backend.

React Documentation:

Utilized as a reference for developing the frontend user interface, creating React components, managing state, and handling user interactions.

Redux Toolkit Documentation:

Used for managing global state in the React application, including user authentication, job listings, and search filters.

Redux Toolkit Official Documentation

Axios Documentation:

Referenced for handling HTTP requests and responses between the React frontend and Spring Boot backend.

Axios GitHub Repository

React Toastify Documentation:

Used to implement toast notifications in the application, providing feedback to users on actions like login, logout, job application, and form submissions.

Vite.js Documentation:

Used to set up and manage the development environment for the React frontend, ensuring fast build times and optimized performance.

Vite.js Official Documentation

Personal Development Experience:

Personal knowledge and experience in Java, Spring Boot, MySQL, and React were extensively used to design, implement, and troubleshoot the application.

Project Prompts and Instructions

8. Details

Ramgopalhyndav Bollepalli

Bollepalliramgopalhyndav@gmail.com

9550753429