DESIGN AND DEVELOPMENT OF AI JOB PORTAL

A

MAJOR PROJECT - II REPORT

Submitted in partial fulfilment of the requirements.

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

GROUP NO. 40

Krishna Kant Chaubey	**0187CS211088**
Nitesh Kumar	**0187CS211111**
Priyanshu Kumar Singh	**0187CS211126**
Raj Kumar	**0187CS211128**

Under the guidance of

Dr. Bhavana Gupta

(Associate Professor)

Department of Computer Science & Engineering
Sagar Institute of Science &Technology (SISTec), Bhopal (M.P)

Approved by AICTE, New Delhi & Govt. of M.P.
Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P)

June -2025

Sagar Institute of Science & Technology (SISTec), Bhopal (M.P)
Department of Computer Science & Engineering

CERTIFICATE

We hereby certify that the work which is being presented in the B.Tech. Major Project-II Report entitled **DESIGN AND DEVELOPMENT OF AI JOB PORTAL,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology***, submitted to the Department of **Computer Science & Engineering**, Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.) is an authentic record of our own work carried out during the period from Jan-2025 to June-2025 under the supervision of **Dr. Bhavana Gupta.**

The content presented in this project has not been submitted by us for the award of any other degree elsewhere.

Krishna Chaubey	**Nitesh Kumar**	**Priyanshu Kr. Singh**	**Raj Kumar**
0187CS211088	**0187CS211111**	**0187CS211126**	**0187CS211128**

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date:

Dr. Bhavana Gupta **Project Guide**	**Dr. Amit Kumar Mishra** **HOD, CSE**	**Dr. D.K. Rajoriya** **Principal**

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. D. K. Rajoriya, Principal, SISTec and Dr. Swati Saxena, Vice Principal SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Amit Kumar Mishra, HOD, Department of Computer Science & Engineering** for his kindhearted support

We extend our sincere and heartfelt thanks to our guide, **Dr. Bhavana Gupta**, for providing us with the right guidance and advice at crucial junctures and for showing us the right way.

We are thankful to the **Project Coordinator, Prof. Deepti Jain**, who devoted her precious time in giving us the information about various aspects and gave support and guidance at every point of time.

We would like to thank all those people who helped us directly or indirectly to complete our project whenever we found ourself in any issue.

TABLE OF CONTENTS

ABSTRACT

In today's competitive job market, traditional job portals face significant limitations in efficiently matching job seekers with relevant opportunities. The AI-Based Job Portal aims to revolutionize the recruitment process by leveraging artificial intelligence (AI) and machine learning (ML) algorithms to automate resume screening, provide personalized job recommendations, and enhance employer-employee connections. This system intelligently analyzes candidate profiles, job descriptions, and market trends to ensure precise job matching, thereby reducing recruitment time and improving accessibility. By integrating AI-driven automation and predictive analytics, the proposed solution streamlines the hiring process, optimizes employer decision-making, and enhances the overall job-seeking experience. The implementation of this intelligent job portal not only benefits job seekers by offering tailored career opportunities but also assists recruiters in identifying the most suitable candidates efficiently, leading to a more effective and data-driven recruitment ecosystem.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
API	Application Programming Interface
USDA	United States Department of Agriculture
Junit	Java Unit Testing
ML	Machine Learning
AI	Artificial Intelligence
OCR	Optical Character Recognition
JOBS	Job Opportunity Based System
UI	User Interface

Chapter 1
Introduction

CHAPTER-1
INTRODUCTION

1.1 ABOUT PROJECT

1.2 This project focuses on building an AI-based job portal that efficiently matches job seekers with relevant job opportunities using artificial intelligence and machine learning algorithms. The application enhances the recruitment process by automating resume screening, providing personalized job recommendations, and improving employer-employee connections. The primary objective is to streamline the hiring process, reduce recruitment time, and enhance job accessibility through intelligent tools and resources.

1.2.1 AUTOMATED RESUME SCREENING: Utilizes AI to analyze resumes and match candidates with suitable job opportunities based on their skills and experience.

1.2.2 PERSONALIZED JOB RECOMMENDATIONS: Leverages machine learning algorithms to suggest job listings tailored to the user's profile and preferences.

1.2.3 EMPLOYER DASHBOARD: Provides recruiters with an interface to post job openings, filter candidates, and manage recruitment efficiently.

1.2.4 JOB SEEKER PROFILE OPTIMIZATION: Assists users in enhancing their profiles with AI-driven suggestions to improve their visibility and job prospects.

1.2.5 REAL-TIME NOTIFICATIONS: E Alerts users about new job postings, application status updates, and relevant employer interactions.

1.3 PROJECT OBJECTIVE

The primary objectives of this project are to develop a smart and user-friendly job portal that optimizes job searching, simplifies recruitment for employers, and enhances job accessibility.

1.3.1 ENHANCED JOB MATCHING: Improve the accuracy of job recommendations by leveraging AI to analyze job descriptions and candidate profiles.

1.3.2 AI-DRIVEN RECRUITMENT: Automate the screening process, reducing the time and effort required for employers to find suitable candidates.

1.3.3 EMPLOYER-EMPLOYEECONNECTIVITY: Facilitate seamless communication between job seekers and recruiters through an intuitive interface.

1.3.4 DATA-DRIVEN INSIGHTS: Provide real-time analytics to both job seekers and employers, improving decision-making in the hiring process.

1.4 FUNCTIONALITY

The following point covers the functionality of the project:

1.4.1 AUTOMATED RESUME SCREENING: AI scans and categorizes resumes based on job requirements, streamlining candidate selection.

1.4.2 JOB RECOMMENDATION SYSTEM: Uses AI to analyze user profiles and suggest job listings that align with their qualifications and experience.

1.4.3 EMPLOYER DASHBOARD: A feature-rich panel for recruiters to post job vacancies, manage applications, and interact with potential candidates.

1.4.4 JOB SEEKER PROFILE OPTIMIZATION: AI-driven recommendations help candidates refine their profiles to enhance job search success.

1.4.5 REAL-TIME JOB ALERTS: Sends notifications for job openings, interview invitations, and application status updates.

1.5 INTERFACE

The interface is designed to be user-friendly, intuitive, and visually appealing, providing time.

1.5.1 RESUME SCREENING INTERFACE: Displays AI-generated analysis of resumes, highlighting key qualifications and suitability for job roles.

1.5.2 JOB RECOMMENDATION DISPLAY: An organized list of job postings tailored to the user's skills and experience, with application options.

1.5.3 EMPLOYER DASHBOARD: A comprehensive panel for recruiters to post jobs, filter applications, and engage with candidates.

1.5.4 JOB APPLICATION TRACKER: Allows job seekers to monitor the status of their applications and receive feedback from employers.

1.5.5 NOTIFICATION SYSTEM: Provides real-time updates on job postings, application progress, and recruiter messages.

1.6 DESIGN AND IMPLEMENTATION CONSTRAINTS

The design and implementation constraints of the projects are:

1.6.1 AI ALGORITHM ACCURACY: The job matching algorithm must ensure precise and reliable candidate-job pairings based on skill sets and requirements.

1.6.2 DATA SOURCES FOR JOB POSTINGS: Requires access to updated and accurate job listings from various industries.

1.6.3 USER PRIVACY AND DATA SECURITY: The system must securely store and protect sensitive user data, including resumes and personal details

These constraints highlight factors to consider in ensuring reliability and accuracy.

1.7 ASSUMPTIONS AND DEPENDENCIES

The following are the assumptions and dependencies of the project:

1.7.1 RELIABLE AI MATCHING: Assumes that AI algorithms will effectively match job seekers with suitable job opportunities**.**

1.7.2 DATA SOURCE AVAILABILITY: Relies on continuous access to updated job postings and recruitment data to function optimally.

Chapter 2
Software & hardware requirements

CHAPTER-2
SOFTWARE & HARDWARE REQUIREMENTS

2.1 INTRODUCTION

This chapter focuses on the essential software and hardware required for developing and deploying the AI-Based Job Portal application. These requirements are carefully chosen to enhance the app's functionality, performance, and security. Software requirements include frameworks, databases, and tools that streamline development while ensuring robust data management and seamless integration. Hardware specifications are optimized to support efficient processing, ensuring a smooth user experience across devices. Emphasis is placed on usability, ensuring the app remains intuitive and accessible. By aligning these requirements with the app's goals, the chapter ensures a balance between technical efficiency, user-friendliness, and secure operations.

2.2 SOFTWARE REQUIREMENTS

The software requirements for the AI-Based Job Portal include essential programming tools, robust development frameworks, versatile libraries, APIs for seamless integration, and reliable databases for efficient data management. Together, these components enable the app’s functionality, streamline development, enhance performance, and ensure a secure, user-friendly experience for both developers and end-users.

2.2.1 DEVELOPMENT ENVIRONMENT

- **VISUAL STUDIO CODE:** I It is a widely used Integrated Development Environment (IDE) for web and mobile application development, offering a comprehensive platform for building, testing, and debugging applications. It provides tools for efficient coding, including a powerful code editor, real-time suggestions, and easy project management. Visual Studio Code also supports extensions that enhance development capabilities and provide additional tools for debugging and integration.

- **VERSION CONTROL SYSTEM**: A Version Control System like Git, used via platforms such as GitHub or Bitbucket, manages code changes efficiently. It supports developer collaboration, tracks all modifications, and enables rollback and conflict resolution.

- This ensures seamless integration and a reliable development workflow for the AI-Based Job Portal. Hosting repositories on platforms like GitHub or Bitbucket enhances teamwork with features like issue tracking, pull requests, and code reviews, ensuring a streamlined and organized development lifecycle to the codebase, facilitating seamless integration of contributions. Its version tracking capabilities ensure a complete history of modifications, allowing rollbacks and conflict resolution when necessary. Hosting repositories on platforms like GitHub or Bitbucket enhances teamwork with features like issue tracking, pull requests, and code reviews, ensuring a streamlined and organized development lifecycle.

2.2.2 PROGRAMMING LANGUAGES

- **PYTHON/JAVASCRIPT**: The primary programming languages for AI and web-based development are Python and JavaScript. Python is widely used for AI and machine learning applications due to its extensive libraries, ease of use, and strong community support. JavaScript is essential for front-end development, enabling dynamic and interactive user interfaces. Combining these languages allows the AI-Based Job Portal to offer a seamless experience with powerful backend functionalities and an intuitive user interface.

2.2.3 MACHINE LEARNING FRAMEWORKS: TensorFlow and Scikit-learn are crucial for implementing AI-driven features in the job portal. These frameworks enable the development of intelligent job matching algorithms that analyze user profiles and job descriptions to provide personalized job recommendations.

- **DATABASE MANAGEMENT**: PostgreSQL and Firebase are used to manage job listings, user profiles, and application data efficiently. PostgreSQL provides a reliable relational database system, while Firebase offers real-time data synchronization and authentication services for enhanced user experience.

- **FRONT-END FRAMEWORKS**: React.js is used for building the user interface, ensuring a responsive and dynamic experience for job seekers and employers.It provides reusable components, enhancing development efficiency anmaintainability.

- **TESTING FRAMEWORKS:** Jest and Selenium are essential testing tools for ensuring the functionality and stability of the AI-Based Job Portal. Jest is used for unit testing, validating the behavior of individual modules and methods. Selenium is

- employed for automated UI testing, simulating user interactions to verify that the user interface responds correctly. Together, these tools help catch bugs early, ensuring the app's reliability, smooth performance, and a seamless user experience throughout its development and deployment

2.3 HARDWARE REQUIREMENTS

The hardware requirements for the AI-Based Job Portal are minimal but should meet specific specifications for optimal performance. For features like AI-based job recommendations, powerful processors and sufficient RAM may be needed to ensure smooth operation. These specifications enhance the platform’s performance and user experience, particularly in resource-intensive tasks.

2.3.1 USER DEVICE REQUIREMENTS

- **OPERATING SYSTEM:** Windows, macOS, or Linux with a modern web browser (Chrome, Firefox, Edge, or Safari) for seamless access to the job portal.

- **PROCESSOR:** A processor with a minimum speed of 2.0 GHz and above for optimal performance when interacting with AI-driven job recommendations and filtering options.

- **RAM:** At least 4GB of RAM to handle AI-based data processing and ensure smooth operation. This is particularly important for real-time job search and filtering functionalities.

- **INTERNET CONNECTIVITY:** A stable internet connection is required to fetch job listings, update user profiles, and communicate with the AI-driven recommendation system.

2.3.2 REQUIREMENTS AT DEVELOPER SIDE

- **COMPUTER (PC OR MAC)**: A modern computer is essential for running development tools. It should have at least 8GB RAM (16GB preferred) and a multi-core processor (Intel i5 or equivalent) to handle the development workload efficiently.
- **RAM:** A minimum of 8GB RAM is recommended for smooth development, with 16GB or more providing better performance, especially when running multiple applications like IDEs, machine learning frameworks, and web servers concurrently.
- **STORAGE SPACE**: Sufficient disk space (at least 100GB) is required to install development tools, databases, libraries, and machine learning models. Additional space is needed for project files, source code, and testing environment.
- **GRAPHICS CARD (GPU)**: A dedicated GPU is beneficial for AI-based training and processing tasks, particularly when working with machine learning models.
- **DISPLAY RESOLUTION**: A physical Android device (smartphone or tablet) is necessary for real-world testing. Emulators are useful, but testing on an actual device ensures better accuracy in performance and usability, particularly for app- specific features like camera access or sensors.
- **DISPLAY RESOLUTION**: A high-resolution display (1080p or better) is helpful for accurately designing and debugging apps. It allows clear visibility of UI elements.

Chapter 3

Problem Description

CHAPTER-3
PROBLEM DESCRIPTION

3.1 OVERVIEW

In today's competitive job market, job seekers and employers face significant challenges in finding the right opportunities and candidates efficiently. Traditional job search methods often involve lengthy application processes, outdated job postings, and a lack of personalized recommendations, making it difficult for both parties to connect effectively. Furthermore, candidates struggle with optimizing their resumes and applications to match the expectations of recruiters and applicant tracking systems (ATS). Employers, on the other hand, face difficulties in sorting through vast numbers of applications to find the most suitable candidates.

This AI-powered job portal is designed to address these challenges by leveraging artificial intelligence to streamline the recruitment process. The platform provides job seekers with AI-driven resume optimization, personalized job recommendations, and real-time career insights to enhance their chances of securing the right job. Employers benefit from intelligent candidate screening, automated shortlisting, and skill-based matching, significantly reducing hiring time and improving recruitment efficiency. By integrating these features, the AI job portal fosters a smarter and more effective hiring ecosystem, ensuring both job seekers and employers make informed decisions with ease.

3.1.1 THE CHALLENGES OF TRADITIONAL JOB SEARCH METHODS

Traditional job search and recruitment methods come with significant inefficiencies. Job seekers often spend hours browsing job boards, submitting applications, and waiting for responses without any feedback. Many job postings are outdated or do not accurately reflect current hiring needs, leading to wasted efforts. Moreover, generic job recommendations fail to consider an individual's skills, experience, and career aspirations, resulting in irrelevant job suggestions.

For employers, manually screening thousands of applications is time-consuming and prone to bias. Many companies rely on keyword-based applicant tracking systems (ATS), which often filter out highly qualified candidates due to mismatched formatting or missing keywords. Additionally, the absence of data-driven hiring insights makes it difficult for recruiters to identify the best-fit candidates efficiently. These limitations highlight the need for an AI-driven solution that enhances job matching accuracy, streamlines the hiring process, and provides valuable career insights in real time.

3.1.2 THE IMPORTANCE OF REAL-TIME JOB MATCHING AND PERSONALIZED CAREER TRACKING

Traditional job search platforms lack real-time insights and personalization, making it difficult for job seekers to adjust their applications based on market demand. Without real-time feedback, job seekers may unknowingly apply for jobs they are underqualified or overqualified for, leading to missed opportunities. Additionally, many job portals do not offer career development insights, leaving users without guidance on skill improvement or career progression.

Employers also suffer from a lack of real-time analytics when assessing job market trends and candidate availability. AI-driven tools can bridge this gap by offering instant recommendations, candidate rankings, and market insights, ensuring that both job seekers and recruiters make data-informed decisions. The integration of AI-powered resume optimization, skill assessment, and automated shortlisting enhances efficiency, leading to faster hiring and better job placements.

3.2 KEY FEATURES AND FUNCTIONALITY

1. **AI-POWERED JOB MATCHING:**
    - The platform uses machine learning algorithms to match job seekers with roles based on their skills, experience, and preferences.
    - Real-time job recommendations ensure that users receive relevant job postings aligned with their career goals.

2. **RESUME OPTIMIZATION TOOL:**
    - AI analyzes resumes and suggests improvements to enhance ATS compatibility.
    - Provides keyword recommendations to increase visibility in recruiter searches.

3. **AUTOMATED CANDIDATE SCREENING:**
    - Employers can leverage AI to filter applications based on required skills, experience, and qualifications.
    - Intelligent shortlisting reduces hiring time and increases recruitment efficiency.

4. **SKILL ASSESSMENT AND TRAINING SUGGESTIONS:**
    - Job seekers receive insights into skill gaps and recommendations for online

- courses or certifications.
- AI-generated career roadmaps help users stay competitive in their industry.

5. REAL-TIME MARKET INSIGHTS:

- Users gain access to labor market trends, salary benchmarks, and industry demands.
- Employers can analyze hiring trends and optimize job postings accordingly.

6. SCALABILITY AND CUSTOMIZATION:

- The platform caters to a broad range of users, including fresh graduates, mid-career professionals, and executives.
- Customizable settings allow users to set job preferences, salary expectations, and preferred industries.

7. CAREER PROGRESS TRACKING:

- Users can track job applications, interview schedules, and employment status through an interactive dashboard.
- Personalized career development plans keep users motivated and goal-oriented.

8. PERSONALIZED RECOMMENDATIONS:

- AI provides tailored job alerts, interview tips, and employer insights based on user behavior and preferences.
- Smart notifications ensure users never miss critical opportunities.

9. USER-FRIENDLY INTERFACE:

- A clean, intuitive design ensures seamless navigation, enabling users to easily access job listings, career tools, and employer information.
- Mobile-friendly design enhances accessibility for users on the go.

Chapter 4

Literature Survey

CHAPTER-4
LITERATURE SURVEY

Android Studio is the official IDE for Android development, offering developers a robust and user-friendly environment to build, test, and deploy Android applications. It combines the power of IntelliJ IDEA with tools specifically designed for Android, such as the Gradle build system, a feature-rich layout editor, and an advanced code editor. Developers can take advantage of intelligent code completion, refactoring tools, and lint warnings to write clean and optimized code. The integrated emulator allows testing apps on a variety of virtual devices, supporting multiple screen sizes, API levels, and configurations, making it easier to ensure compatibility across devices. Additionally, Android Studio includes profiling tools to monitor performance metrics like CPU, memory, and battery usage, helping developers optimize their apps. The layout editor, combined with live previews, enables developers to design user interfaces visually and test changes in real-time without needing to deploy the app repeatedly. Android Studio also provides robust debugging features, such as a unified debugging window and support for Java, Kotlin, and C++. For testing, the IDE integrates with frameworks like JUnit and Espresso to streamline unit and UI testing. The official documentation serves as a comprehensive resource, covering everything from setting up the IDE and creating your first app to advanced topics like Jetpack libraries and app architecture. By providing extensive resources, tutorials, and guides, Android Studio documentation empowers developers to create high-quality applications that adhere to Android's design and performance standards.[1]

Firebase is a powerful development platform by Google designed to help developers build apps faster and more efficiently. With a wide range of services like authentication, database management, cloud storage, and machine learning, Firebase enables developers to focus on creating features rather than managing backend infrastructure. Firebase Authentication provides secure sign-in methods, supporting email, Google, Facebook, and other providers. The Realtime Database and Cloud Firestore are cloud-hosted NoSQL databases that allow synchronized data sharing across users in real-time or offline. Cloud Storage offers scalable file storage for user-generated content like images and videos. Firebase Analytics helps developers track user engagement and behavior, providing insights that can inform design and feature decisions. For app performance and reliability, tools like Firebase Crashlytics offer real-time crash reporting and diagnostics, helping teams quickly identify and fix issues. Firebase ML extends the platform's capabilities by integrating machine learning, enabling apps to perform tasks like image recognition or language translation without building models from

scratch. Its flexible APIs and SDKs support Android, iOS, and web apps, making Firebase suitable for cross-platform development. The official documentation is a rich resource containing tutorials, API references, and integration guides for every service, helping developers get started quickly. Whether you're building a simple app or a complex, multi-platform solution, Firebase provides the tools needed to deliver engaging and scalable applications while minimizing the challenges of backend management. [2]

Mobile health (mHealth) applications are becoming vital tools for promoting healthy behaviors and improving overall well-being. Studies highlight the positive impact these apps have on users, particularly in supporting healthy eating habits and offering diet management tools. By providing personalized tracking, nutritional insights, and reminders, mHealth apps help users make informed decisions about their diet and lifestyle. This increased accessibility to health data and guidance plays a significant role in encouraging healthier eating patterns and long-term health improvements. Image recognition technology is advancing rapidly, allowing mobile applications to analyze and interpret visual information effectively. In the context of SugarCare, this technology is crucial for its product label scanning feature, which extracts and analyzes text from food packaging. By accurately identifying ingredients, nutritional facts, and sugar content, SugarCare can provide users with valuable insights about their food choices. This functionality enhances the app's ability to help users manage their sugar intake by easily scanning and understanding product labels.

USDA FoodData Central is an extensive resource maintained by the U.S. Department of Agriculture, offering a centralized database for food composition and nutrient information. This platform serves a wide audience, including developers, researchers, and consumers seeking reliable food data. It provides detailed information across various categories, such as branded foods, foundational ingredients, and experimental data sets. The branded foods section includes nutritional profiles for commercially available products, while the foundation foods category focuses on raw and minimally processed ingredients. Additionally, the SR Legacy database offers historical data for commonly consumed foods, supporting studies and applications requiring older references. One of the standout features of FoodData Central is its API, which allows developers to integrate rich food and nutrient data into their applications.

This can be especially beneficial for health and fitness apps, dietary planners, or any tool aimed at promoting better nutrition. By providing access to data on calories, macronutrients, and micronutrients, the platform helps users make informed dietary choices. FoodData Central also includes data on food additives and alternative sweeteners, addressing growing consumer concerns over processed foods and hidden sugars. The documentation accompanying the platform is comprehensive, covering topics like API authentication, query structures, and data parsing, making it easy for developers to incorporate this resource into their projects. Whether for app development, scientific research, or consumer education, FoodData Central serves as an invaluable tool in promoting nutritional awareness and healthier lifestyles. [3]

Jetpack Compose is Android's modern, declarative UI toolkit, revolutionizing how developers build user interfaces for applications, including job portals. Unlike the traditional View system, Jetpack Compose simplifies UI development by using Kotlin's declarative syntax, enabling developers to build visually appealing interfaces with significantly less code. The toolkit revolves around "composable" functions, which are reusable UI components that update dynamically as the app state changes, eliminating the need for manual updates and improving code maintainability. Its tight integration with Material Design ensures that apps built with Jetpack Compose adhere to Android's design principles, offering built-in support for themes, typography, and pre-designed components. One of its most powerful features is interoperability with the existing View system, allowing developers to incrementally migrate their apps to Compose without starting from scratch. Jetpack Compose also includes tools like live previews and real-time recomposition, enabling developers to test and tweak their designs instantly without recompiling the app. For advanced features, Compose supports animations, navigation, and complex layouts, all while maintaining excellent performance. The official documentation provides a wealth of resources, including tutorials, sample projects, and API references, to help developers master the framework. Its simplicity, flexibility, and scalability make it ideal for developing AI job portals that require dynamic UIs for candidate profiles, job listings, application tracking, and recruiter dashboards. [4]

Chapter 5

Software Requirement Specifications

CHAPTER-5
SOFTWARE REQUIREMENTS SPECIFICATION

5.1 FUNCTIONAL REQUIREMENTS

- **USER REGISTRATION AND AUTHENTICATION:** The application must allow users to securely register and log in using methods such as email/password or social media integration. Both job seekers and recruiters should have dedicated roles during registration, enabling role-based access and features.

- **JOB SEARCH AND FILTERING FEATURES:** The application should provide robust features for job seekers to search, filter, and sort job listings based on various parameters such as location, salary range, job type, skills, and experience level. It must also allow recruiters to search and filter candidate profiles based on specific criteria.

- **RESUME UPLOAD AND PARSING:** The app should provide a feature for users to upload resumes in various formats. It will extract key professional information such as skills, work experience, education, and certifications using AI-powered parsing, which will then be used for job matching and profile enhancement.

- **PROFILE MATCHING ENGINE:** The application must include an AI-driven profile matching engine that compares job requirements with user resumes and preferences. It should provide personalized job recommendations for users and potential candidate recommendations for recruiters based on compatibility scores.

- **PERSONALIZED DASHBOARDS:** Based on user type (job seeker or recruiter), the app should generate personalized dashboards. Job seekers should be able to track job applications, saved jobs, and interview schedules, while recruiters should have access to applicant tracking, job post status, and hiring analytics.

- **INTEGRATED JOB POSTING:** The app must feature a job posting system for recruiters. Employers can create, edit, and manage job listings, including setting role

descriptions, required qualifications, salary, and location. The system should also support scheduling application deadlines and receiving candidate applications in real-time

- **SKILL DEVELOPMENT AND TRAINING MODULES:** The app should provide access to a curated list of training resources and skill-development courses. Users can select training based on their career goals and track their progress, helping them become more competitive in the job market.

- **NOTIFICATIONS AND ALERTS**: The application should send reminders and alerts to users about job matches, application status updates, recruiter messages, and interview reminders. Users should have customizable settings for managing the frequency and type of notifications they receive.

5.2 NON-FUNCTIONAL REQUIREMENTS

- **USABILITY:** The app must have a clean, intuitive, and user-friendly interface, ensuring users can easily navigate through features such as resume parsing, job search, profile updates, and the dashboard. The design should reduce cognitive load and offer a seamless user experience for both job seekers and recruiters.

- **PERFORMANCE:** The application should offer fast response times, with smooth transitions between screens and features. It should efficiently handle large datasets such as user resumes, job listings, and recruiter communications, ensuring high performance even during peak usage.

- **SECURITY:** User data, including resumes, personal details, and hiring records, must be stored securely. The application should use encryption to protect sensitive information. Privacy policies must comply with regulations such as GDPR, ensuring user confidentiality and data integrity.

- **SCALABILITY:** The app must be reliable and have minimal downtime. It should handle errors gracefully and ensure that core features (job searching, resume parsing, job posting, and user communication) function correctly under various conditions.

- **RELIABILITY:** The app must be reliable and have minimal downtime. It should handle errors gracefully and ensure that core features (job searching, resume parsing. job posting, and user communication) function correctly under various conditions.

- **COMPATIBILITY:** The app should be compatible across a wide range of devices and operating systems, including Android and iOS, ensuring broad accessibility. It should work seamlessly on smartphones and tablets of various sizes and resolutions.

- **MAINTENANCE AND SUPPORT:** Regular maintenance should be provided to fix bugs, introduce new features, and improve app performance based on user feedback. An in-app support system should be available for users to get help with any issues they encounter

.

- **BATTERY AND RESOURCE EFFICIENCY:** The app should be optimized for minimal battery consumption and efficient use of system resources like CPU and memory.

- **BACKUP AND DATA RECOVERY:** The app should have data backup mechanisms in place, ensuring that user data can be recovered in case of system failures.

- **LOCALIZATION AND INTERNATIONALIZATION:** The app should support multiple languages and regions, ensuring accurate translations and date/time formats.

Chapter 6

Software & Hardware Design

CHAPTER-6
SOFTWARE DESIGN

6.1 SOFTWARE DESIGN

Software Design is a critical phase in software development that involves planning and defining the system's architecture, components, user interfaces, and data structures. It acts as a blueprint for the development process, ensuring that the software meets both functional requirements (what the software should do) and non-functional requirements (how the software should perform). This stage focuses on creating a robust, scalable, and maintainable system, guiding the development team in implementing a solution that aligns with user needs, business goals, and technical constraints.

6.1.1 ACTORS

- Job Seeker: The job seeker is a primary end-user of the mobile app, interacting with features like searching for jobs, uploading resumes, receiving job recommendations, and applying for positions. They have personalized access to manage their profile, saved jobs, and application history.
- **Recruiter**: The recruiter manages job postings and interacts with candidate profiles. They have access to create and manage listings, review applicants, and communicate with potential hires through the platform.
- **Admin**: The admin manages the mobile app’s backend, overseeing user accounts, app settings, and platform content. They have elevated access to monitor performance, ensure compliance, provide user support, and maintain the app’s security and smooth operation.

6.1.2 USE CASES

- Upload Resume and Profile Creation: Job seekers can upload their resumes, which the system processes using AI-based parsing to extract relevant details like experience, skills, and education to build a professional profile automatically.
- Search and Apply for Jobs: Job seekers can search for jobs based on filters such as job type, location, salary, and experience level. They can view job descriptions and apply directly through the app.

- Post a Job Listing: Recruiters can create and manage job postings, input role requirements, set deadlines, and review submitted applications within the app.

- Dashboard Access: Both job seekers and recruiters have access to dashboards.

- Job seekers see saved jobs, applications, and matches, while recruiters manage listing.

- Job Matching and Recommendations: Users can view detailed nutritional information for various fruits, including sugar content, calories, vitamins, and other essential nutrients, to make informed choices based on their dietary needs.

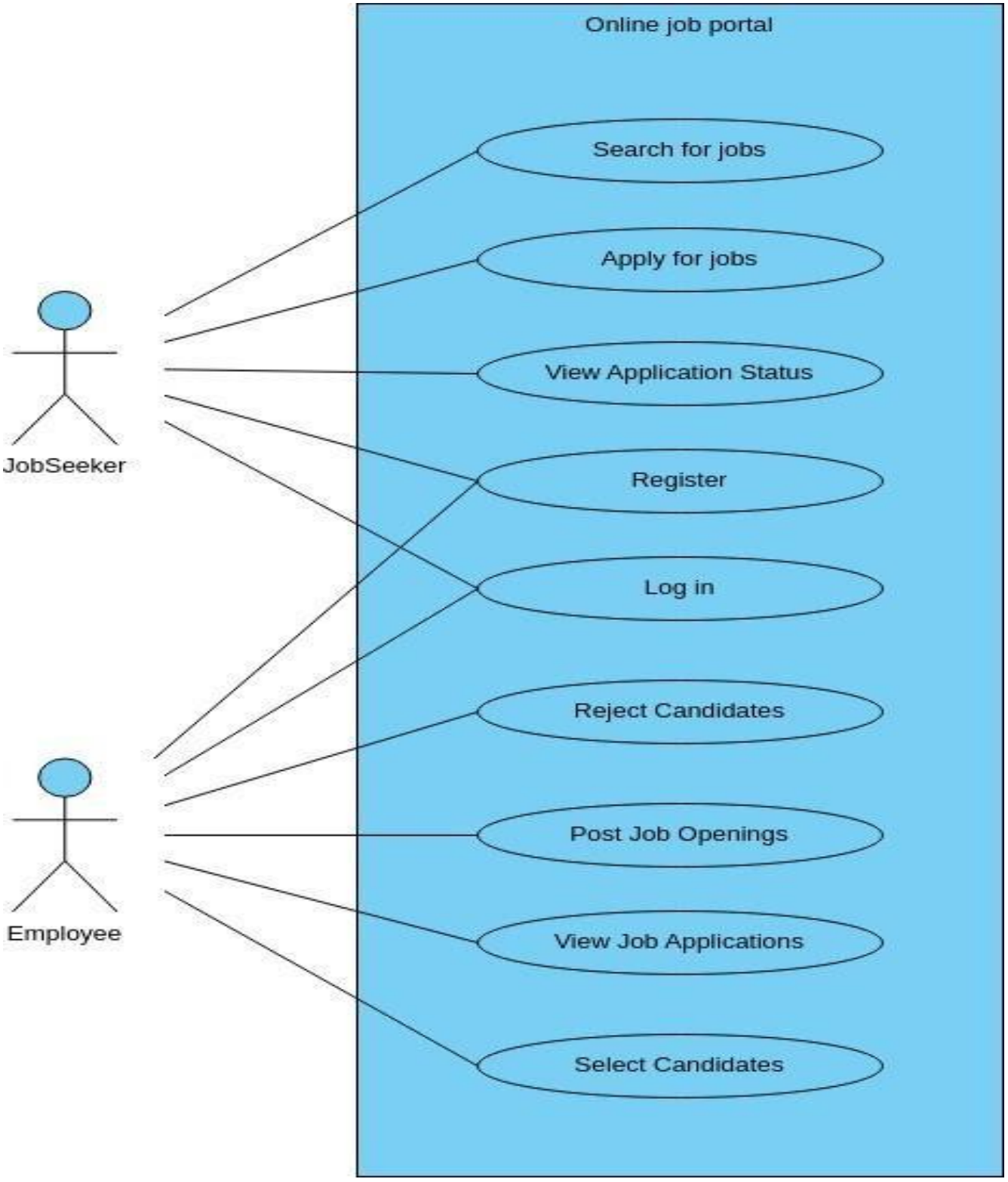


Figure 6.1: Use case diagram

6.2 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system. It illustrates how input data is processed, stored, and transformed into output data. DFDs help visualize the movement and transformation of information within a system, highlighting how different components interact with each other.

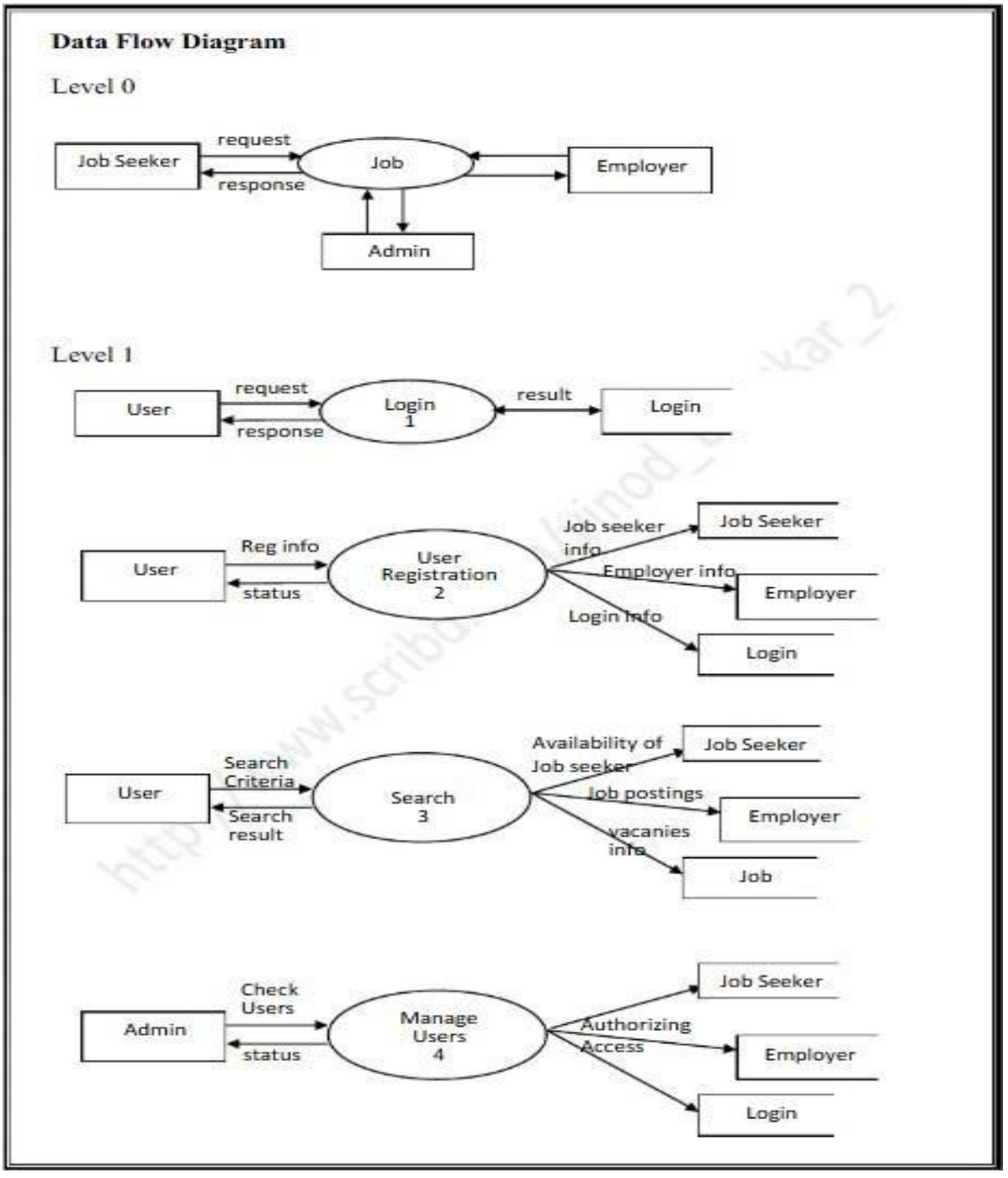


Figure 6.2: Dataflow diagram

Chapter 7
Module

CHAPTER-7
ANDROID MODULE

7.1 INTRODUCTION

In the context of Android development, a module represents a distinct part of an application or project that can function independently or as part of a larger application. Modules help in structuring, organizing, and reusing code effectively, particularly for large projects. This chapter provides an in-depth look at Android modules, their types, structure, and practical applications, especially for a multi-module Android project like the AI Job Portal, where different functionalities like user profiles, job listings, and AI-based recommendations are separated for better scalability and performance.

7.1.1 WHAT IS AN ANDROID MODULE?

An Android module is a collection of source code, resources, and build configurations that work together to perform specific functionalities. Modules can be thought of as building blocks of an Android project, allowing developers to compartmentalize and modularize code. Each module can be independently built, tested, and debugged. For an AI Job Portal, modularizing the project allows functionalities such as job search, resume parsing, and recruiter dashboards to be developed independently. Android Studio supports various types of modules, including app modules, library modules, and more, which can be combined to create a feature-rich application.

7.1.2 TYPES OF ANDROID MODULES

1. **App Module**
    - This is the main module that acts as the entry point of the AI Job Portal app.
    - Contains essential logic, resources, and the AndroidManifest.xml for the entire application.
    - It is the installable component that integrates features like job search, user login, and recommendation engine.
2. **Library Module**

- Includes reusable components such as UI components, utilities for parsing resumes, or AI logic.

- Outputs an Android Archive (.AAR) file.
- Useful for maintaining consistency across features like profile rendering and job cards.

3. **Dynamic Feature Module**
   - Supports downloading features like job interview prep tools or resume building guides on demand.
   - Reduces initial app size by delivering only required features.
   - Ideal for AI-powered recommendation tools or optional recruiter tools.

4. **Java/Kotlin Module**
   - Contains core logic for AI algorithms or job-matching engines, independent of Android APIs.
   - Outputs a Java Archive (.JAR) file.
   - Useful for encapsulating AI logic, job recommendation models, or language processing functions.

7.1.3 STRUCTURE OF AN ANDROID MODULE

An Android module contains the following key components:

- src/main/java: Includes Java or Kotlin source code such as login controllers, job search handlers, and resume parsers.
- src/main/res: Stores UI resources like XML layouts for job listings, drawables for icons, and strings for localization.
- src/main/AndroidManifest.xml: Declares app components such as activities, services, and permissions.
- build.gradle (Module Level): Configures dependencies and SDK versions specific to each module (e.g., Firebase for auth module).
- build.gradle (Project Level): Configures global settings and dependencies shared across all modules.

7.1.4 ADVANTAGES OF USING MODULES

- **Code Reusability:** Library modules can be reused in multiple projects.
- **Scalability:** Modularization makes it easier to scale projects by separating concerns.
- **Parallel Development:** Teams can work on different modules simultaneously without conflicts.
- **Faster Build Times:** Gradle builds smaller, independent modules instead of the entire project.
- **Testing and Debugging:** Modules can be individually tested and debugged.

7.1.5 CREATING A MODULE IN ANDROID STUDIO

1. **Steps to Create a Module:**
    - Open the AI Job Portal project in Android Studio.
    - Go to File > New > Module.
    - Choose the appropriate module type (e.g., Android Library for resume parser, Dynamic Feature Module for optional AI tools).
    - Define the module name, package structure, and minimum SDK.
    - Click Finish.
2. **Setting Module Dependencies:**
    - Open the build. gradle file of the app module.
    - Add a dependency on the new module using:
    - implementation project(':module name')

7.1.6 BEST PRACTICES FOR MODULARIZATION

- **Feature-Based Modules:** Create modules such as jobListing, resumeAnalysis, auth, and recruiter Panel for separation of concerns**.**
- **Domain-Specific Modules:** Useseparatemodules fornetworking, AI logic, anddatastorageto ensurecleanarchitecture.
- **Keep Modules Lightweight:** Preventtight coupling byminimizing directdependenciesbetween modules.

- **Use Dependency Injection:** Tools like Hilt or Dagger help manage shared objects across modules like repositories or data services**.**

7.1.7 MODULARIZING A PROJECT

For the AI Job Portal, modularization may look like:

1. **app Module:** Main navigation, app-wide configurations, and UI shell**.**
2. **auth Module:** User registration, login, and social authentication features**.**
3. **jobSearch Module:** Handles job browsing, filtering, and advanced search**.**
4. **resumeParser Module:** Parses uploaded resumes using AI and NLP techniques**.**
5. **recruiterDashboard Module:** Manages job postings, applicant tracking, and analytics.
6. **recommendationEngine Module**: AI models for job-candidate matching.

This modular approach enables independent development and testing of each feature.

7.1.8 TESTING AND DEBUGGING MODULES

- **Unit Testing:** Write tests for modules like recommendation Engine to verifyprediction accuracy.
- **Instrumentation Testing:** Use UI tests for modules like job Search or recruiter Dashboard**.**
- **Gradle Tasks:** Use Gradle commands such as assemble Debug or test to isolate and test modules during CI/CD processes**.**

7.1.9 CHALLENGES IN MODULARIZATION

- **Complexity in Setup:** Requires careful planning of dependencies and module boundaries.
- **Build Time:** Over-modularization may increase build complexityandduration**.**
- **Inter-Module Communication:** Requires clean architecture practices like interfaces, shared view models, or event buses to handle module interaction without tight coupling.

Chapter 8
Coding

CHAPTER-8
CODING

8.1 OVERVIEW

The coding phase highlights the development process of the **AI Job Portal Android application**, built using **Kotlin**, **Jetpack Compose**, **Firebase**, and **machine learning-based job recommendation models**. This chapter explains how each technology was utilized to implement the application's features and ensure efficient development practices.

Kotlin served as the primary programming language, offering concise, expressive syntax and seamless compatibility with Jetpack Compose. **Jetpack Compose** was used for building responsive, dynamic UIs, supporting components such as job listings, candidate profiles, and recruiter dashboards through reusable and modular design.

Firebase was integrated for backend services including **authentication**, **Cloud Firestore** for real-time job and user data storage, and **Cloud Storage** for resume uploads and media management. It provided secure, scalable infrastructure and real-time data synchronization across user roles like job seekers and recruiters.

For intelligent job recommendations, **machine learning models** were implemented and connected via a modular backend, helping match candidates with the most relevant job postings based on their resumes, skills, and preferences.

Version control was handled with **Git**, enabling team collaboration and codebase management. Error-handling strategies, modular design patterns, and architectural best practices (such as MVVM) ensured stability, maintainability, and scalability of the application.

8.1.1 FUTURE SCOPE OF CODING

The future scope of coding in the **AI Job Portal** domain is immense, driven by rapid advancements in artificial intelligence, data analytics, and cloud computing. With continued evolution in **AI and machine learning**, the application can expand into **predictive job matching**, **interview preparation AI assistants**, and **automated skill gap analysis**.

Features such as **voice-enabled job search**, **multi-language support**, **blockchain-based verification for credentials**, and **real-time chatbots for recruitment support** can further enhance the platform. Additionally, integration with **IoT for wearable-based productivity tracking** and **edge computing** for resume parsing in offline scenarios may broaden its utility.

As the job market continues to digitize and adopt AI-driven solutions, coding will remain essential in building **scalable, intelligent, and adaptable platforms** like the AI Job Portal that empower both job seekers and recruiters in a seamless ecosystem.

8.2 HOME PAGE

```
package com.example.sugarfree

import android.os.Bundle
import
 androidx.activity.ComponentActivity
import
androidx.activity.compose.setContent
import
androidx.compose.foundation.Image
import
androidx.compose.foundation.background
import
androidx.compose.foundation.clickable
import
androidx.compose.foundation.layout.*
import
 androidx.compose.foundation.lazy.LazyRow
 import
 androidx.compose.foundation.lazy.items
import androidx.compose.foundation.rememberScrollState
 import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
 import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import
 androidx.compose.material3.MaterialTheme
 import androidx.compose.runtime

@Composable// Home page
fun HomeScreen(navController: NavController)
{
Scaffold(
 bottomBar = {
```

```
BottomNavigationBar(navController) }
                ) {
innerPadding
Column(
modifier = Modifier
.fillMaxSize()
.padding(innerPadding)
.background(Color(0xFFF8F8F8))
.verticalScroll(rememberScrollState())
)
Composable
fun HeaderSection() {
Column(modifier =
Modifier.padding(16.dp)) { Text(
text = "Find Your Dream Job",
fontSize = 24.sp,
fontWeight = FontWeight.Bold,
 color = Color(0xFF1A237E)
)
Text(
text = "Powered by AI for smarter matches", fontSize
 = 14.sp,
color = Color.Gray
)
}
}


@Composable
fun SearchSection(navController:
NavController) { Column(modifier =
Modifier.padding(16.dp)) {
OutlinedT
 extField(
 value =
 "",
 onValue
 Change =
 {},
placeholder = { Text("Job title, keywords, or company") },
leadingIcon = { Icon(Icons.Default.Search,
contentDescription = "Search") }, modifier =
Modifier.fillMaxWidth()
)
Spacer(modifier =
 Modifier.height(12.dp))
```

```
OutlinedTextField(
value = "",
onValueC
 hange =
 {},
placeholder = { Text("Location") },
leadingIcon = { Icon(Icons.Default.LocationOn,
 contentDescription = "Location") }, modifier =
 Modifier.fillMaxWidth()
)

Spacer(modifier =

 Modifier.height(12.dp))

 Button(
onClick = { navController.navigate("jobSearch") },
modifier = Modifier.fillMaxWidth()
) {
Text("Search Jobs")
}
}
}

@Composable
fun FeaturedJobsSection(navController:
 NavController) { Column(modifier =
 Modifier.padding(16.dp)) {
Text("Featured Jobs", fontSize = 18.sp, fontWeight =
 FontWeight.SemiBold) Spacer(modifier =
 Modifier.height(8.dp))

LazyRow(horizontalArrangement =
 Arrangement.spacedBy(12.dp)) {
 items(sampleJobList) { job ->
JobCard(job = job, navController = navController)


@Composable
fun HeaderSection() {
Column(modifier =
Modifier.padding(16.dp)) { Text(
text = "Find Your Dream Job",
fontSize = 24.sp,
fontWeight = FontWeight.Bold,
 color = Color(0xFF1A237E)
)
Text(
text = "Powered by AI for smarter matches", fontSize
 = 14.sp,
color = Color.Gray
)
}
}
```

```
@Composable
fun SearchSection(navController:
NavController) { Column(modifier =
Modifier.padding(16.dp)) {
OutlinedT
 extField(
 value =
 "",
 onValue
 Change =
 {},
placeholder = { Text("Job title, keywords, or company") },
leadingIcon = { Icon(Icons.Default.Search,
 contentDescription = "Search") }, modifier =
 Modifier.fillMaxWidth()
)

Spacer(modifier =

 Modifier.height(12.dp))

 OutlinedTextField(


value = "",
onValueCh
ange = {},
placeholder = { Text("Location") },
leadingIcon = { Icon(Icons.Default.LocationOn, contentDescription =
 "Location") }, modifier = Modifier.fillMaxWidth()




)

Spacer(modifier =

 Modifier.height(12.dp))

 Button(
onClick = {
 navController.navigate("jobSearch
 ") }, modifier =
 Modifier.fillMaxWidth()
) {
Text("Search Jobs")
}
}
}


@Composable
fun FeaturedJobsSection(navController:
 NavController) { Column(modifier =
```

```
Modifier.padding(16.dp)) {
Text("Featured Jobs", fontSize = 18.sp, fontWeight =
 FontWeight.SemiBold) Spacer(modifier =
 Modifier.height(8.dp))
        }
}
}
}


@Composable


fun
 ResumeUploadSection(navController:
 NavController) { Card(
  modifier =
    Modifier
.fillMaxWidt
        h()
 .padding(16.
        dp)
.clickable {
 navController.navigate("resumeUpload")
 }, backgroundColor =
 Color(0xFFE8EAF6),
elevation = 4.dp
) {
Row(
modifier =
 Modifier.padding(16.dp),
 verticalAlignment =
 Alignment.CenterVertically
) {


Icon(Icons.Default.UploadFile, contentDescription = "Upload", tint =
Color(0xFF3F51B5)) Spacer(modifier = Modifier.width(16.dp))
Column {
Text("Upload Your Resume", fontWeight =
 FontWeight.Bold) Text("Get job matches based
 on your



skills", fontSize = 12.sp)
}
}}}

@Composable
fun
```

```
AIRecommendationsSection(navController:
 NavController) { Card(
  modifier =
   Modifier
.fillMaxWidt
      h()
.padding(16.
      dp)
.clickable {
 navController.navigate("aiRecommendatio
 ns") }, elevation = 4.dp
) {
Column(modifier = Modifier.padding(16.dp)) {
Text("AI Job Recommendations", fontWeight =
 FontWeight.Bold) Text("Smart suggestions tailored to
 your profile", fontSize = 12.sp)
}
}
}

@Composable
fun CareerTipsSection() {
Column(modifier = Modifier.padding(16.dp)) {
Text("Career Tips", fontWeight = FontWeight.SemiBold, fontSize = 16.sp)

val tips = listOf(
"
 description", "    Use keywords
 for better visibility",
      Apply actively and
 follow up", μ H      Keep
 learning new skills"
)

Text(text = it, fontSize = 14.sp, modifier = Modifier.padding(vertical = 4.dp))
}
}
}
@Composable
fun BottomNavigationBar(navController:
NavController) { BottomAppBar(containerColor =
MaterialTheme.colorScheme.primary) {
IconButton(onClick = {
navController.navigate("home") }) {
Icon(Icons.Default.Home,
 contentDescription =

"Home")
}
IconButton(onClick = {
 navController.navigate("jobSearch") }) {
```

```
Icon(Icons.Default.Search,
 contentDescription = "Search")
}
IconButton(onClick = {
 navController.navigate("profile") }) {
 Icon(Icons.Default.Person,
 contentDescription = "Profile")
}
}
}

import Cookies from "js-cookie";

// post job api
//Index.js

export const post_job =

 async (formData) => { try

 {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/postAJob
 `, { method: 'POST',
headers: {
'Content-Type': 'application/json',
 'Authorization': `Bearer
 ${Cookies.get('token')}`
},
body: JSON.stringify(formData),
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in post job (service) => ', error);
}}


// get job api
export const
get_job = async
() => { try {
const res = await

fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getAllJobs
 `, { method: 'GET',
headers : {
'Authorization': `Bearer ${Cookies.get('token')}`
}
```

```
})
const data
 =

res.json();
 return
 data;
} catch (error) {
console.log('error in getting job (service) => ', error);
}
}

// get specified job api
export const get_specified_job = async (id) => {



try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getSpecifiedJob?id=${
 id}`, { method: 'GET',
headers : {'Authorization': `Bearer ${Cookies.get('token')}`}
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in getting  specified job (service) => ', error);
}
}


// apply  job api

export const apply_job =
 async (formData) => {
 try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/applyJob
 `, { method: 'POST',
headers : {'Authorization': `Bearer
 ${Cookies.get('token')}`}, body: formData,
});
const data =
 await
 res.json();
 return data;
} catch (error) {
console.log('error in apply job (service) => ', error);
```

```
}
}

// get my all applied job api

export const
 get_my_applied_job = async
 (id) => { try {
const res = await



fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getAppliedJobs?id=
 ${id}`, { method: 'GET',
headers : {'Authorization': `Bearer ${Cookies.get('token')}`}



})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in getting  getting my all job (service) => ', error);
}
}

// get my all posted job api

export const
 get_my_posted_job = async


(id) => { try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getPostedJobs?id=$
 {id}`, { method: 'GET',
headers : {'Authorization': `Bearer ${Cookies.get('token')}`}
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in   getting my all job (service) => ', error);
}
}

// get my all application of
```

specified jobs api export

```
const get_all_applications =

async (id) => {
try {


const res = await
fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getAllApplica
tionsOfSpecifiedJob?id=${id}`, {
method: 'GET',
headers : {'Authorization': `Bearer ${Cookies.get('token')}`}
})
const data
=
res.json();
return
data;
} catch (error) {
console.log('error in   getting my all application of specified jobs (service) => ', error);
}
}

// change application status api

export const change_application_status
= async (formData) => { try {
const res = await
fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/responseOfApplicat
ion`, { method: 'PUT',
headers: {
'Content-Type': 'application/json',
'Authorization': `Bearer
${Cookies.get('token')}`
},
body: JSON.stringify(formData),
})
const data
=
res.json();
return
data;
} catch (error) {
console.log('error in   getting my all application of specified jobs (service) => ', error);
}
}



export const
get_application_details = async
(id) => { try {
const res = await
```

```
fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/getApplicationDetail?id=$
 {id}`, { method: 'GET',
headers : {'Authorization': `Bearer ${Cookies.get('token')}`}
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in   getting my all application of specified jobs (service) => ', error);
}
}
```

```
//import Cookies from "js-cookie";

// bookmark job api
export const book_mark_job = async (formData) => {

try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/bookmark
 `, { method: 'POST',
headers: {
'Content-Type': 'application/json',
 'Authorization': `Bearer
 ${Cookies.get('token')}`
},
body: JSON.stringify(formData),
})
const data
 =
 res.json();
 return
 data;
} catch (error) {


console.log('error in bookmark job (service) => ', error);
}
}

// get bookmark job api

export const
 get_book_mark_job = async
 (id) => { try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/bookmark?id
```

```
=${id}`, { method: 'GET',
headers: {
'Authorization': `Bearer ${Cookies.get('token')}`
},
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in getting bookmark job (service) => ', error);
}
}

// delete bookmark job api

export const
 delete_book_mark_job = async
 (id) => { try {
const res = await
 fetch(`${process.env.NEXT_PUBLIC_API_BASE_URL}/api/job/bookmark
 `, { method: 'DELETE',



headers: {
'Authorization': `Bearer
 ${Cookies.get('token')}`, 'Content-
 Type': 'application/json',
},
body : JSON.stringify(id),
})
const data
 =
 res.json();
 return
 data;
} catch (error) {
console.log('error in deleting bookmark job (service) => ', error);
}
}



//AppliedjobSlice.js
import { createSlice } from '@reduxjs/toolkit'

const
 initialS
 tate = {
 applied
 Job :
```

```
[],
 bookM
 ark :
 [],


}
export const
 appliedJobSlice =
 createSlice({ name:
 'AppliedJob',
in
 {
setAppliedJob : (state,
 action) => {
 state.appliedJob =
 action.payload
},
setBookMark : (state ,
 action) => {
 state.bookMark =
 action.payload
}
},
})



// Action creators are generated for each case reducer function
export const { setAppliedJob , setBookMark} =

 appliedJobSlice.actions export const

 AppliedJobReducer = appliedJobSlice.reducer


//jobSlice.js




import { createSlice } from '@reduxjs/toolkit'

const
 initialS
 tate =
 {
 JobDat
```

```
a:   null,
 matchi
 ngData
 : null,
 myJob
 s :
 null,
}

export const
jobSlice =
createSlice({
name: 'Job',
in
 :
 {
setJobData :
 (state, action)
 => {
 state.JobData =
 action.payload
},
setMatchingJobDat : (state , action)
 => { state.matchingData =
 action.payload
},
setMyJobs :
 (state , action)
 => {
 state.myJobs =
 action.payload
}
},
})

// Action creators are generated for each case reducer function


export const { setJobData , setMatchingJobDat ,

 setMyJobs } = jobSlice.actions export const JobReducer

 =  jobSlice.reducer


//userSlice.js

import { createSlice }

 from '@reduxjs/toolkit'

 const initialState = {
```

```
export const
userSlice =
createSlice({
name: 'User',
in
 :
 {
setUserData :
 (state, action)
 => {
 state.userData
 =



action.payload
},
setUserToken : (state,
 action) => {
 state.userToken =
 action.payload
}

},
})

// Action creators are generated for each
 case reducer function export const {
 setUserData , setUserToken } =
 userSlice.actions


export const UserReducer = userSlice.reducer

//gitingore

# See https://help.github.com/articles/ignoring-files/

 for more about ignoring files. # dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# next.js
/.next/
/out/

# production
```

build

misc
.DS_Store
*.pem

debug

.pnpm-debug.log*

local env files
.env*.local

vercel

.vercel
This App is Live and Running At the following #### https://job-portal-teal.vercel.app
adding my code

- don't forget to leave a star ! :)

Authors

- [Abdullah Moiz](https://www.github.com/Abdullah-moiz)

Features

- SignIn / SignUp
- Forget Password
- Post A Job
- View All Jobs
- View Details of Any Job
- BookMark Jobs
- Track Your BookMark Jobs in DataTable View
- Track Your Applied Jobs in DataTable View
- Track Your Posted Jobs and view Submitted Application
- Accept and Reject Different Application
- Status Updated for applied Job based on Job poster action
- JWT validation on each Authorized Request

(Below Feature works only in Local Environment as Vercel Don't allow write operation in free plan so CV are unable to saved in production while you can use firebase , or aws s3 bucket to store , but works fine in local App)

- View or download Applicant CV's

Tech

- Nextjs
- tailwind css
- Redux toolkit
- joi validation
- mongoDB
- SWR hooks for fetching API

Environment Variables

To run this project, you will need to add the following environment variables to your .env file

`DB_URI` = Your mongoDB URL

`JWT_SECREAT` = Your custom JWT_SECREAT key

`NEXT_PUBLIC_API_BASE_URL` = Base URL for localhost => http://localhost:3000

Installation

Install my-project with npm

```bash
npm install
npm run dev (for development server) npm run build (for Production)
npm run preview (To View Production Server )
```

🛠 Tools

Redux toolkit (for state management),
MongoDB,

Screen shots

Loading Screen

![image](https://user-images.githubusercontent.com/90745903/235368351-699df61b-15bb-429d-9387-c724cc4c0d75.png)

Home Page

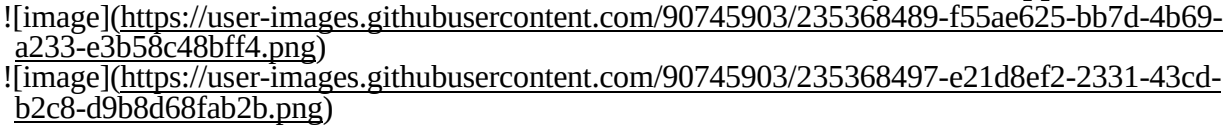
search Job based on tags

login Page

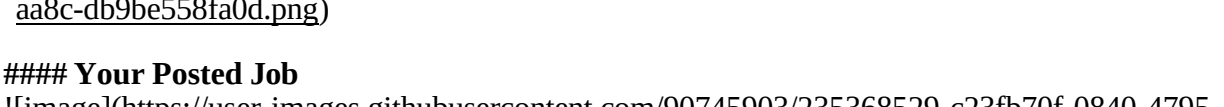
Register Page

Forget Password

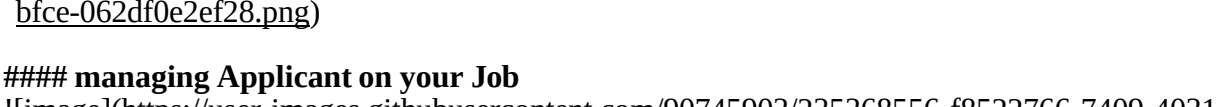
Dashboard to trace Your Favorite Jobs and Jobs on which you have Applied



Post Job



Your Posted Job



managing Applicant on your Job

View Job

//tailwind.config.js

```
/** @type
 {import('tailwindcss').Conf
 ig} */ module.exports = {
```

```
content: [
  "./app/**/*.{js,
  ts,jsx,tsx}",
"./pages/**/*.{js,ts,jsx,tsx}",
"./components/**/*.{js,ts,jsx,tsx}",

// Or if using
  `src` directory:
  "./src/**/*.{js,t
  s,jsx,tsx}",
],
: {
'fade-
  in-
  down'
  : {
  '0%':
  {
opacity: '0',
transform: 'translateY(-10px)'
{
opacity: '1',
transform: 'translateY(0)'


},
}
},
boxShadow: {
'neon': ' 0 0 10px #06d6a1, 0 0 20px #06d6a1, 0 0 40px #06d6a1, 0 0 80px #06d6a1',
},
animation: {
'fade-in-down': 'fade-in-down 0.5s ease-out'
},

},
},
plugins: [],

}
```

```
import { configureStore }
 from '@reduxjs/toolkit'
 import { UserReducer }
 from '@/Utils/UserSlice'
 import { JobReducer }
 from '@/Utils/JobSlice'
import { AppliedJobReducer } from '@/Utils/AppliedJobSlice'

export const store
 =
 configureStore({
 reducer: {
User: UserReducer, // UserReducer is a function that
 returns a slice of state Job : JobReducer, //
 JobReducer is a function that returns a slice of state
AppliedJob : AppliedJobReducer, // AppliedJobReducer is a function that returns a slice of state
},
}

//applyjob.js

import ConnectDB from
 '@/DB/connectDB'; import
 Joi from 'joi';
import AppliedJob from
 '@/models/ApplyJob';
 import formidable from
 'formidable';
import
 fs
 from
 'fs';
 impor
 t path
 from
 'path'
import crypto from 'crypto';
import validateToken from '@/middleware/tokenValidation';

const
schema =
Joi.object({
name:
Joi.string().
required(),
email:
 Joi.string().email(
 ).required(),
 about:
 Joi.string().requir
 ed(),
```

```
job:
 Joi.string(
 ).required(
 ), user:
 Joi.string(
 ).required(
 ),
});

export
 const
 config
 = {
 api: {
bodyParser: false,
},
};

export default
 async (req, res)
 => { await
ConnectDB();
const {
method }
 = req;
 switch
 (method)
 {
case 'POST':



await validateToken(req, res,
 async ()

=> { await applyToJob(req,
 res);

});
:
res.status(400).json({ success: false, message: 'Invalid Request' });
}
}

const applyToJob =
 async (req, res) => {
 await ConnectDB();

try {
const form = new
 formidable.IncomingForm();
 form.parse(req, async (err, fields,
```

```
files) => {
if (err) {

console.error('
Error', err)

throw err
}

const oldPath = files.cv.filepath;
const originalFileName  = files.cv.originalFilename


const fileExtension =
 path.extname(originalFileName); const
 randomString =
 crypto.randomBytes(6).toString('hex');
const fileName = `${originalFileName.replace(fileExtension,

 ")}_${randomString}${fileExtension}`; const newPath =

 path.join(process.cwd(), 'public', 'uploads', fileName);

// Read the file
fs.readFile(oldPath,
function (err, data) { if
(err) throw err;
fs.writeFile(newPath, data,
 function (err) { if (err) throw
 err;
});
fs.unlink(oldPath,
 function (err) { if (err)
 throw err;
});
});
const
jobApplicati
on = { name:
fields.name,
email:
fields.email,
about:
fields.about,
job:
fields.job,
user:
 fields.u
 ser, cv:
 fileNa
```

Chapter 9
Result & Output Screens

CHAPTER-9
RESULT AND OUTPUT SCREEN

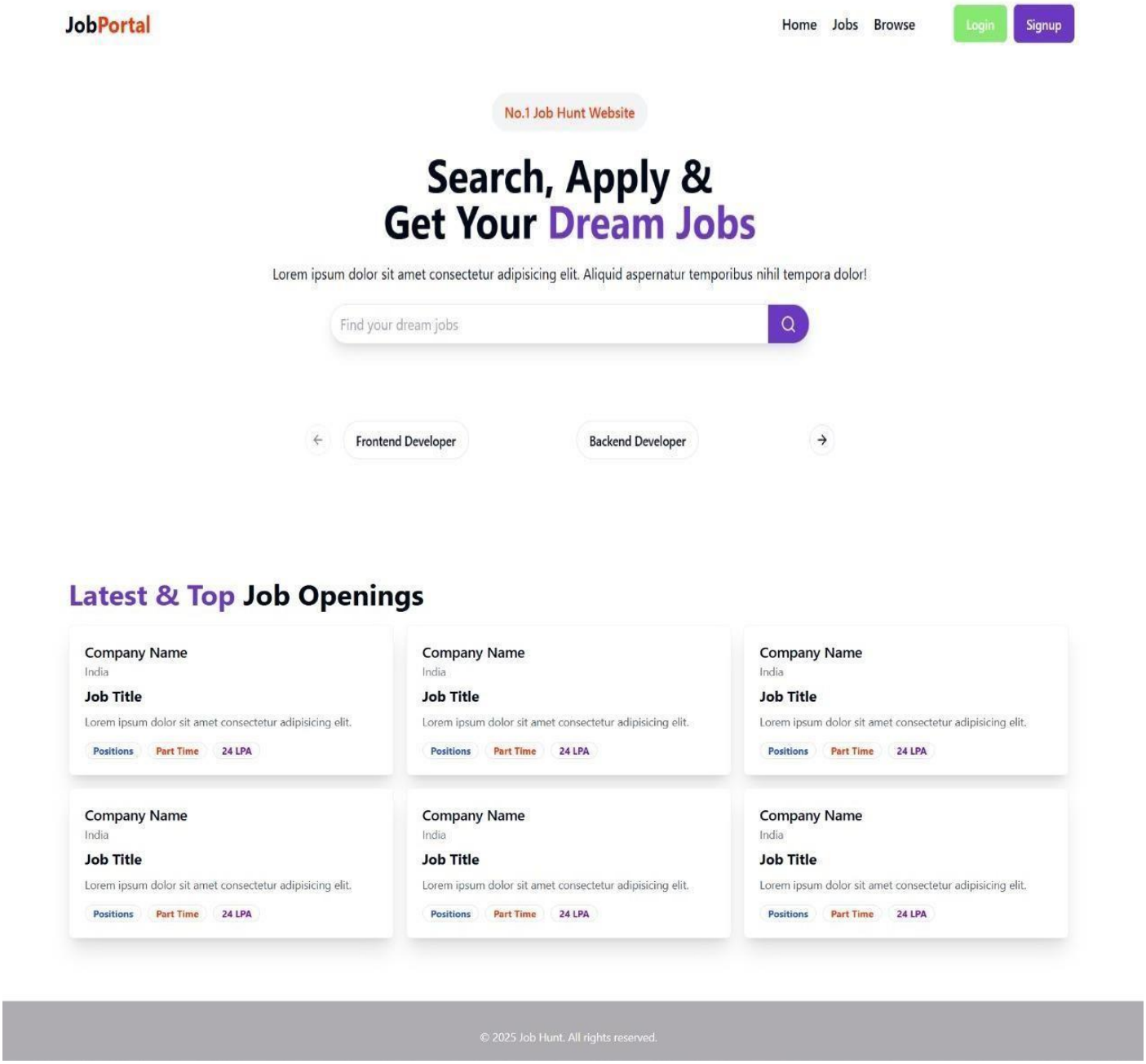


Figure 9.1: Home Page & Footer Page

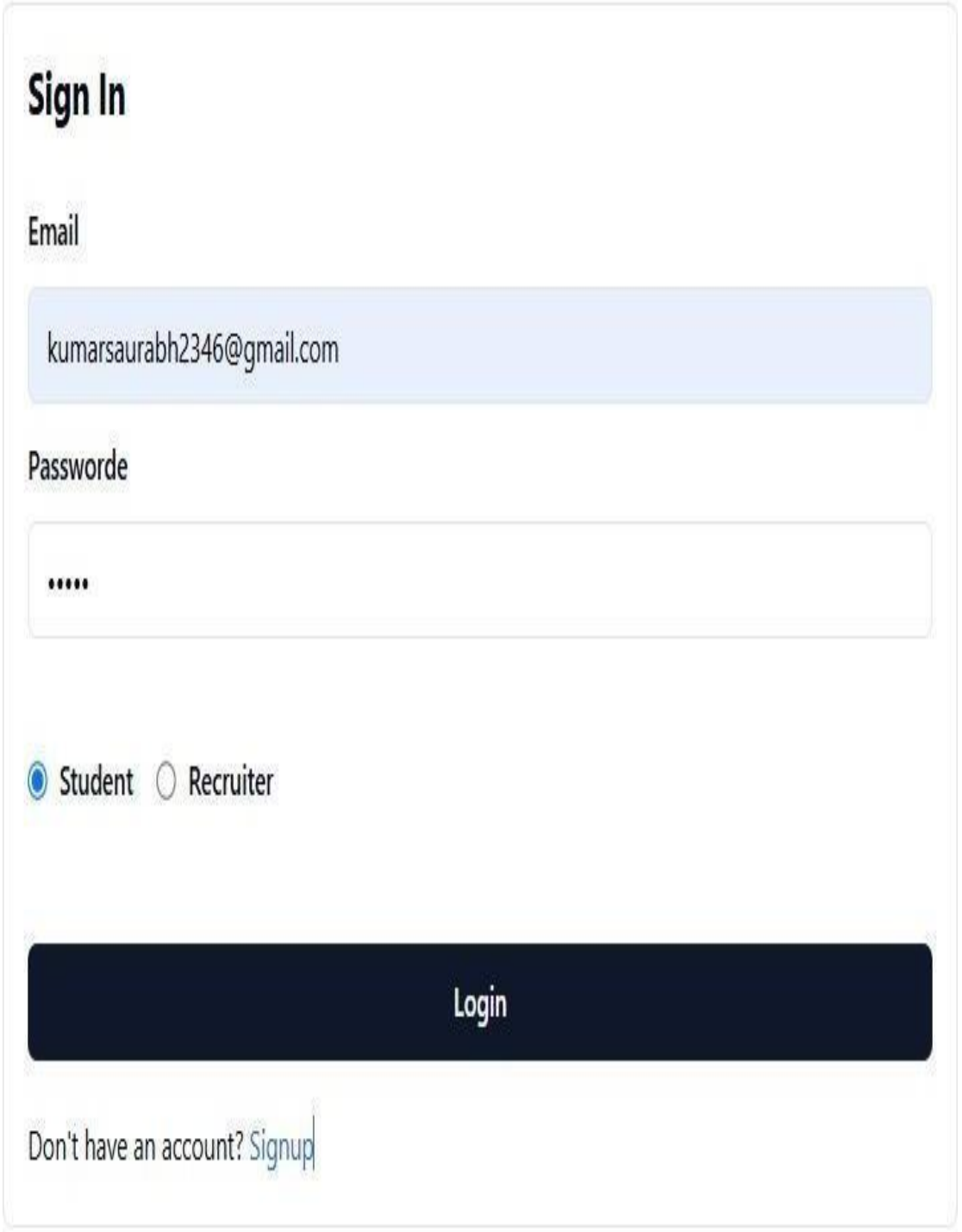


Figure 9.2: Sign In Page

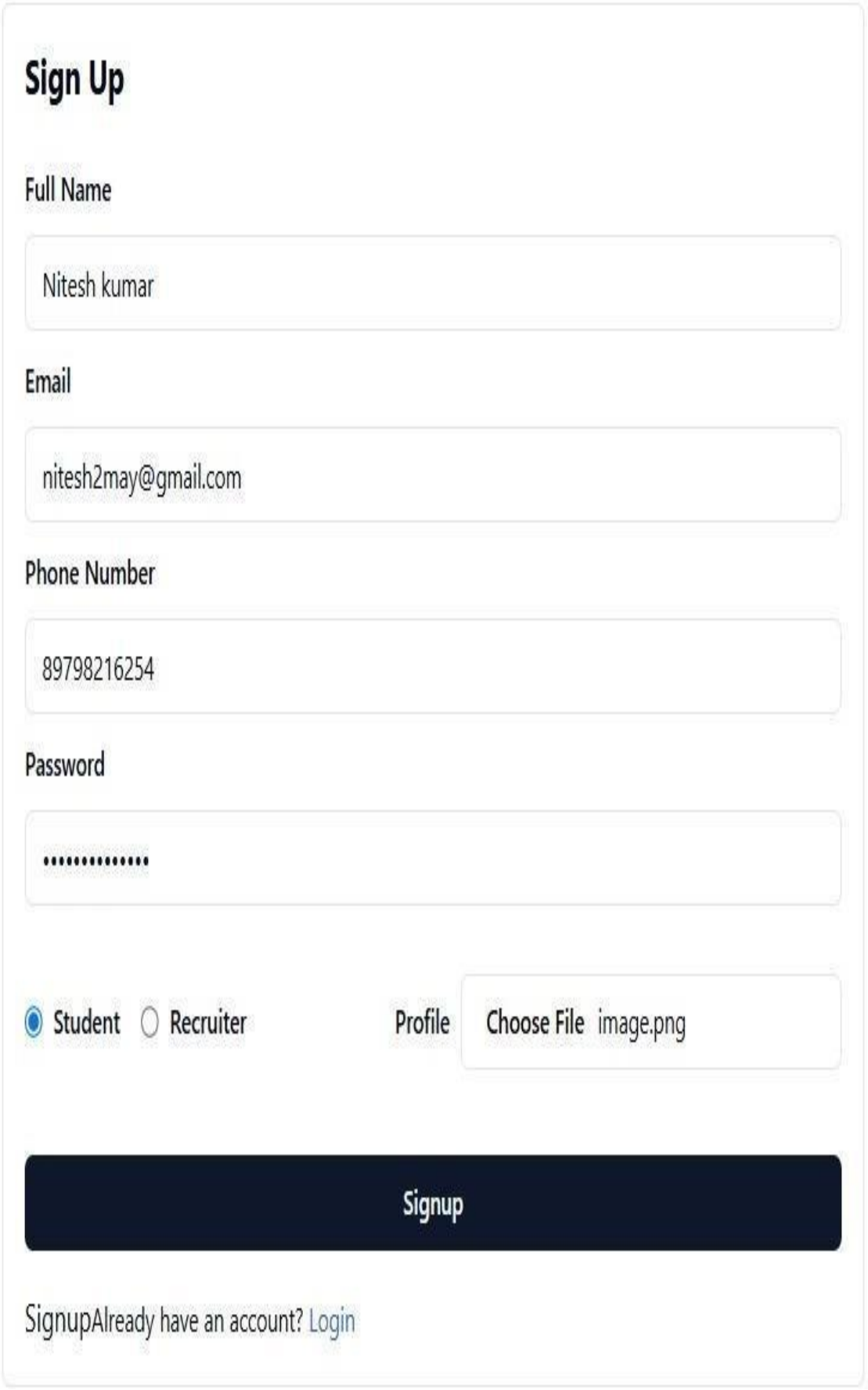


Figure 9.3: Sign Up

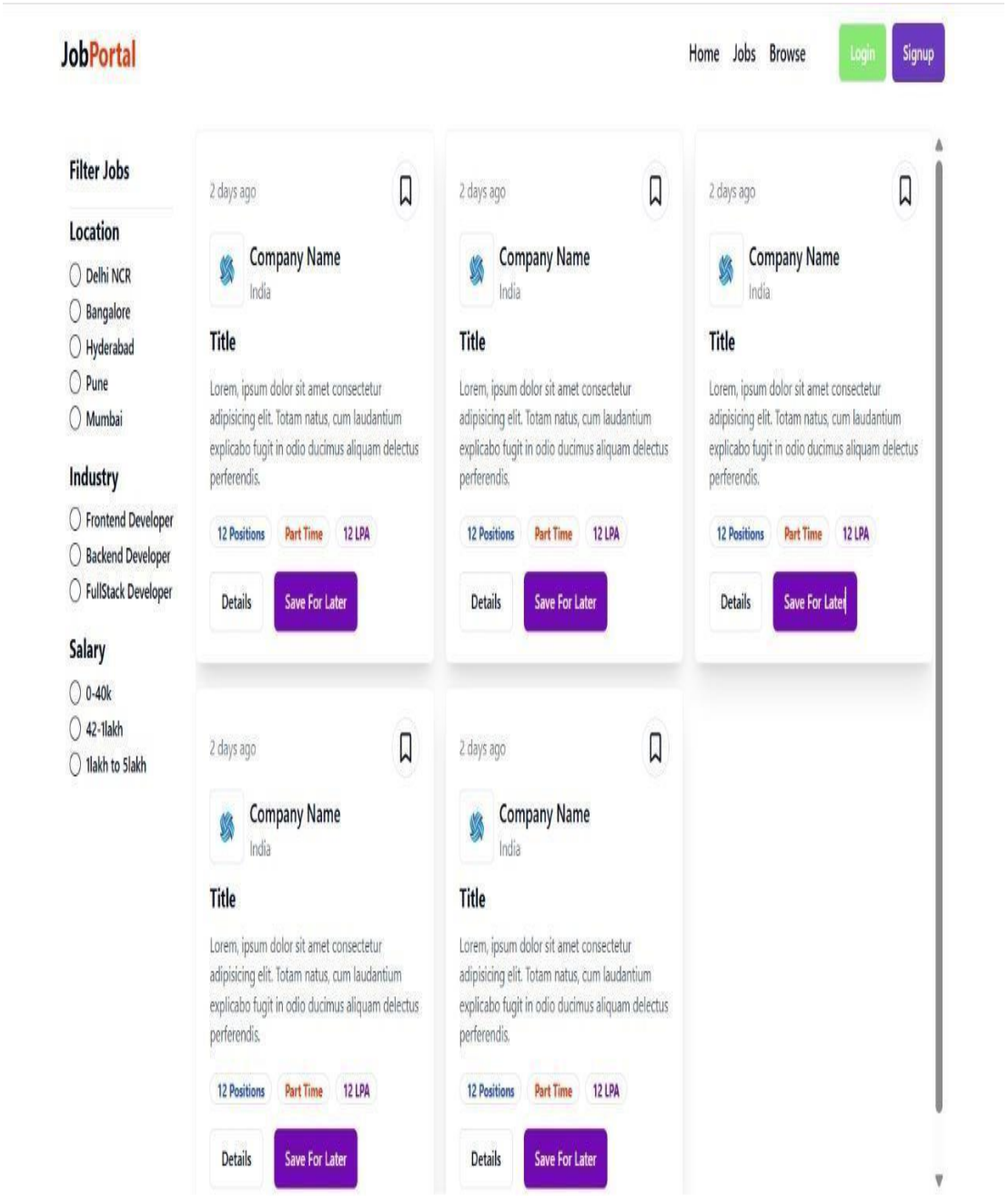


Figure 9.4: Opportunities

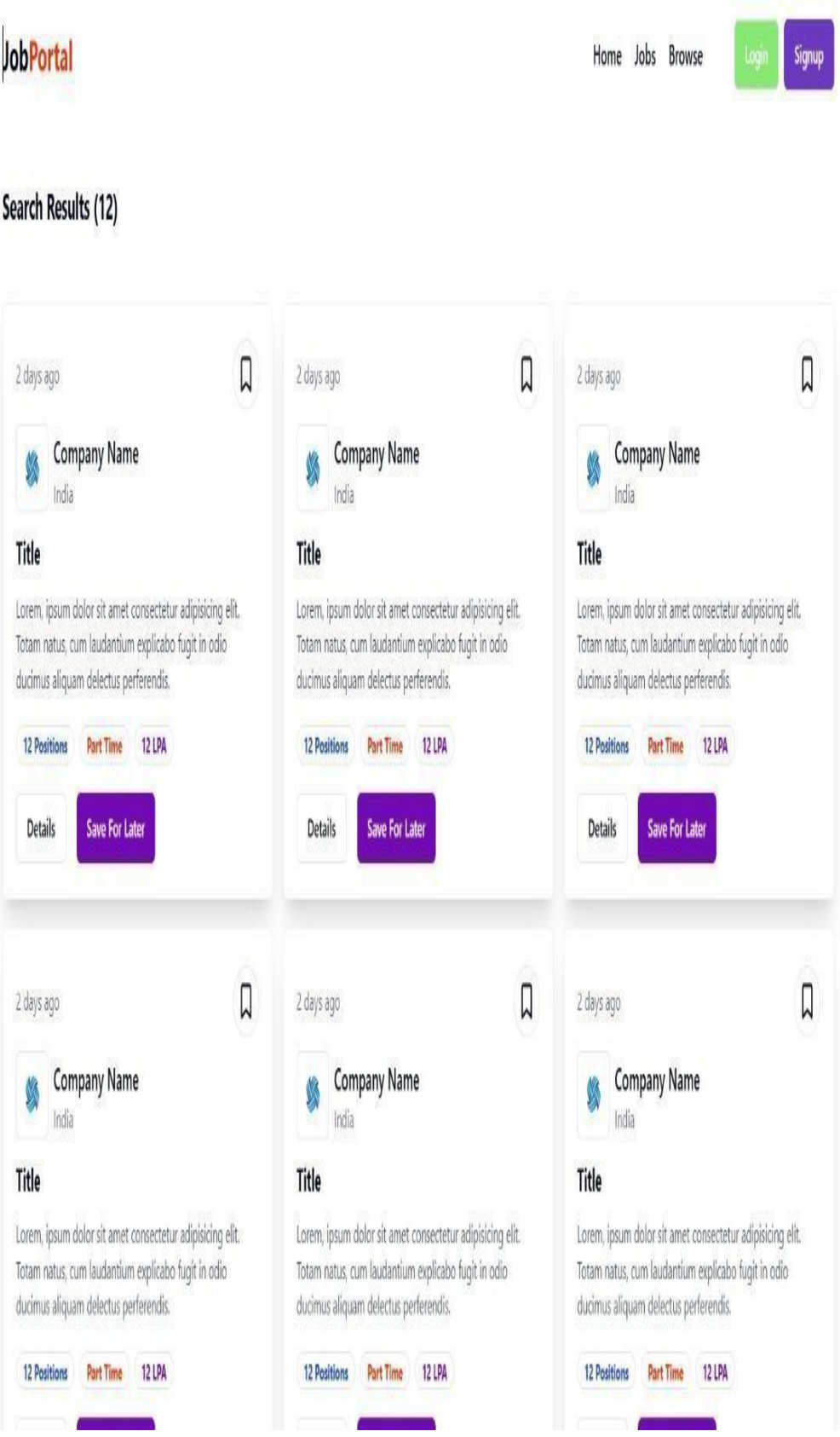


Figure 9.5: Search Result

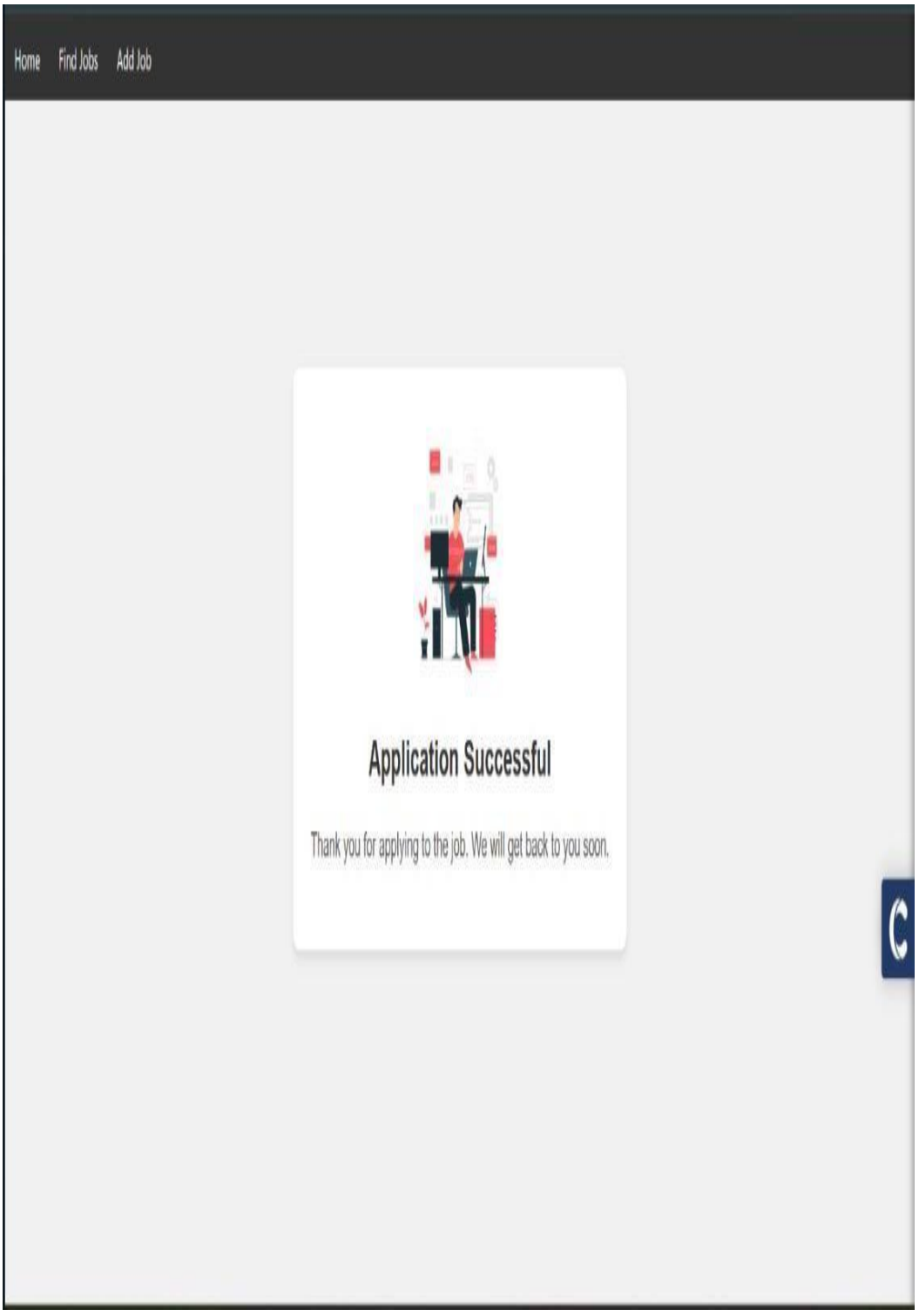


Figure 9.6: Applied Job

Chapter 10

Conclusion & Future Work

CHAPTER-10
CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

In conclusion, the development of our AI-driven job portal application represents a significant step forward in redefining how individuals discover and connect with career opportunities in today's rapidly evolving job market. By integrating critical features such as AI-powered job matching, resume scanning, skill-gap analysis, personalized recommendations, and a curated job search experience, our platform offers a holistic solution to the challenges job seekers face in navigating complex employment landscapes.

Through thorough research and analysis of recruitment trends and emerging technologies, we identified key components that shaped the design and functionality of our application. By prioritizing user experience, data security, and intelligent automation, we created a tool that supports individuals in making informed career decisions based on their unique profiles and preferences.

As we move forward, our application will continue to adapt and grow based on user feedback and the advancement of AI and recruitment technologies. We remain committed to enhancing the platform by introducing innovative features, optimizing performance, and ensuring it remains aligned with current hiring practices and professional development strategies.

Ultimately, our AI Job Portal is more than just a job search tool—it is an empowering career companion. By offering personalized insights, continuous support, and intelligent job discovery, we aim to make a meaningful impact in the lives of job seekers globally, improving employment outcomes and fostering career growth.

10.2 FUTURE WORK

1. AI-POWERED INSIGHTS AND RECOMMENDATIONS

- Integrate advanced machine learning models to analyze user profiles and application history for more precise job suggestions.
- Offer dynamic career guidance based on market trends, skills in demand, and individual preference.

2. INTEGRATION WITH PROFESSIONAL NETWORKS AND TOOL

- Enable syncing with platforms like LinkedIn, GitHub, and online portfolios to enhance profile visibility and data accuracy.
- Support real-time updates of achievements, certifications, and work history.

3. GAMIFICATION

- Introduce gamified elements such as career growth badges, milestone tracking, and goal challenges to boost user engagement.
- Implement leaderboards to encourage healthy competition among job seekers based on activity and achievements.

4. MULTI-LANGUAGE SUPPORT

- Expand accessibility by introducing multi-language support to cater to diverse job seekers across different regions.

5. INTEGRATION WITH RECRUITERS AND COMPANIES

- Add functionality for recruiters to post jobs, view applicant insights, and conduct AI-assisted shortlisting.
- Enable live chat or video call functionality for virtual interviews and direct engagement.

6. OFFLINE FUNCTIONALITY

- Allow users to browse saved job listings and prepare applications offline, with automatic synchronization upon reconnecting to the internet.

REFERENCES

WEBSITES

[1] **Android Studio Documentation:** https://developer.android.com/develop

[2] **Firebase Documentation:** Firebase - https://firebase.google.com/

[3] **USDA FoodData Central**: https://fdc.nal.usda.gov/

[4] **Jetpack Compose: ht**tps://developer.android.com/develop/ui/compose/ documentation

[5] **You tube:** https://www.youtube.com

PROJECT SUMMARY

About Project

Title of the project	DESIGN AND DEVELOPMENT OF AI JOB PORTAL
Semester	8th
Members	4
Team Leader	Nitesh Kumar
Describe role of every member in the project	Krishna Kant Chaubey – User Interface & Database Management. Nitesh Kumar– User Interface & Database Management. Priyanshu Kumar Singh– User Interface & Database Management. Raj Kumar– User Interface & Deployment.
What is the motivation for selecting this project?	The motivation for this project stems from the growing global challenge faced by job seekers in navigating the increasingly competitive and dynamic employment landscape. With rising unemployment rates, rapidly changing skill requirements, and the disconnect between job seekers and recruiters, there is a pressing need for intelligent solutions that can bridge this gap. This project aims to address that need by providing an accessible, AI-driven platform that personalizes the job search experience, enhances employability, and empowers users to make informed career decisions—ultimately promoting better employment outcomes and supporting career growth in today's digital age.
Project Type **(Desktop Application, Web Application, Mobile App, Web)**	Web Application.

Tools & Technologies

Programming language used	React JS, Tailwind CSS, Bootstrap, Node.js, Express.js, MongoDB
Compiler used **(with version)**	Vite
IDE used	Visual Studio Code

(with version)	
Front End Technologies (With version, wherever Applicable)	React.js, Tailwind CSS, Bootstrap
Back End Technologies (With version, wherever applicable)	Node.js, Express.js
Database used (with version)	MongoDB

Software Design & Coding

Is the prototype of the software developed?	Yes
SDLC model followed (Waterfall, Agile, Spiral etc.)	Agile SDLC model
Why is the above SDLC model followed?	The **Agile model** is ideal for dynamic, evolving projects like the AI Job Portal, which integrates multiple features such as intelligent job matching, resume parsing, and real-time user feedback. It emphasizes **iterative development and continuous collaboration**, allowing for greater flexibility and adaptability throughout the project lifecycle. This approach works best when requirements may change or expand over time, ensuring the product remains aligned with user needs and market trends. By incorporating regular updates and incremental improvements, Agile minimizes risk, improves usability, and supports the integration of advanced technologies like AI, ensuring the successful delivery of a robust and user-centric application.
Justify that the SDLC model mentioned above is followed in the project.	The AI Job Portal project follows the Agile SDLC model through iterative development, frequent user feedback, and continuous improvement. Key phases include dynamic requirement gathering, modular implementation, regular testing, and incremental deployment. Agile ensures adaptability, allowing the team to respond quickly to changing needs and deliver a user-centric platform with evolving features like AI-based job recommendations and resume analysis.

Software Design approach followed(Functional or Object-oriented)	Object Oriented
Name the diagrams developed (According to the Design approach followed)	Use Case Diagram, Data Flow Diagram

In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	Encapsulation, Inheritance
No. of Tiers (example 3-tier)	3-tier
Total no. of front-end pages	10
Total no. of tables in database	4
Database in which Normal Form?	3NF
Are the entries in the database encrypted?	Yes
Front end validations applied (Yes / No)	Yes
Session management done (in case of web applications)	
Is application browser compatible (in case of web applications)	
Exception handling done (Yes / No)	Yes
Commenting done in code (Yes / No)	Yes

Naming convention followed (Yes / No)	Yes
What difficulties faced during deployment of the project?	
Total no. Of Use-cases	1
Given titles of Use-cases	Use Case Diagram

Project Requirements

MVC architecture followed (Yes / No)	No
If yes, write the name of MVC architecture followed (MVC-1, MVC-2)	N/A
Design Pattern used (Yes / No)	No
If yes, write the name of Design Pattern used	N/A
Interface type (CLI / GUI)	GUI
No. of Actors	2
Name of Actors	User, Admin
Total no. of Functional Requirements	12
List few important non-Functional Requirements	Performance, Security, Compatibility

Testing

Which testing is performed? (Manual or Automation)	Manual
Is Beta testing done for this project?	Yes

Write project narrative covering above mentioned points

The project addresses the growing challenge in today's job market of effectively connecting job seekers with the right opportunities. With increasing competition and skill mismatches, traditional portals often fall short. This AI-based job portal provides intelligent features like resume analysis, personalized job recommendations, and real-time alerts to bridge that gap. It simplifies hiring for recruiters and empowers candidates to make informed career choices, promoting smarter hiring, reduced unemployment, and improved job satisfaction through a user-friendly, tech-driven platform. The application also supports features like profile building, skill-based filtering, chat assistance, and document uploads, making it a comprehensive solution that caters to both job seekers and recruiters in a highly efficient way.

Krishna Kant Chaubey	0187CS211088
Nitesh Kumar	0187CS211111
Priyanshu Kr. Singh	0187CS211126
Raj Kumar	0187CS211128

Guide Signature
(Dr. Bhavana Gupta)

APPENDIX 1 GLOSSARY OF TERM

A	
AI (Artificial Intelligence)	Artificial Intelligence refers to the simulation of human intelligence processes by machines. In this project, AI is used to analyze user profiles and resumes to provide personalized job recommendations, improving job matching efficiency.
I	
IDE	An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application.
J	
Jetpack Compose	Jetpack Compose is a modern UI toolkit from Google used for building native Android user interfaces. It simplifies and accelerates UI development with a declarative approach, enhancing the performance and maintainability of the AI Job Portal app.
F	
Firebase	Firebase is a platform developed by Google for creating mobile and web applications. It provides backend services such as real-time database, authentication, cloud storage, and hosting, used extensively in this project for app function.
K	
Kotlin	Kotlin is a statically typed programming language used for Android development. It is the primary language used in this AI Job Portal project, favored for its modern syntax, safety features, and full interoperability with Java.