Job Finder

By
Md.Ariful Islam
201-15-3741

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements for the **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Md. Abbas Ali khan

Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Co-Supervised by

Tapasy Rabeya

Lecturer (Senior Scale)
Department of Computer Science and Engineering
Daffodil International University

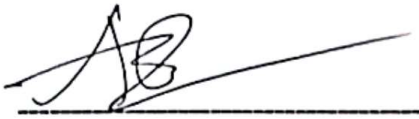
DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

May 14, 2025

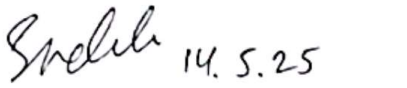
APPROVAL

This Project titled **"Job Finder"**, submitted by **Md.Ariful Islam** , ID No: **201-15-3741** to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **14 May, 2025**.

BOARD OF EXAMINERS

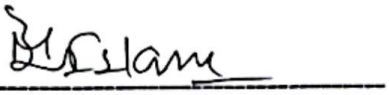


Dr. Arif Mahmud — **Chairman**
Associate Professor and Associate Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Md. Sadekur Rahman — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Tapasy Rabeya — **Internal Examiner**
Sr. Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

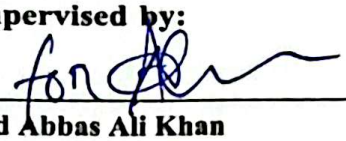


Dr. Md. Manowarul Islam — **External Examiner**
Associate Professor
Department of Computer Science and Engineering
Jagannath University

DECLARATION

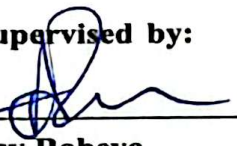
We hereby declare that this project has been done by myself under the supervision of **Abbas ali khan, Assistant Professor,** Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this projec or any part of this project has been submitted else where for the award of any degree or diploma.

Supervised by:



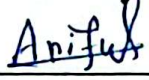
Md Abbas Ali Khan
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University

Co-Supervised by:



Tapasy Rabeya
Sr.Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by:



Md.Ariful Islam
Student ID: 201-15-3741
Department of Computer Science and Engineering
Daffodil International University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has assisted us in one way or another.

First, we express our heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for us to complete the **Final Year Design Project(FYDP)** successfully.

We are grateful and wish our profound indebtedness to **Abbas ali khan**,**Assistant Professor** , Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of our supervisor in the field of **Mobile Apps Development** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting themat all stages have made it possible to complete this project.

We would like to express our heartfelt gratitude to the Head of the Department of Computer Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

We would like to thank our entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

In today's fast-moving digital world, finding the right job—or the right candidate—can feel like searching for a needle in a haystack. Traditional hiring methods often involve a lot of time, paperwork, and back-and-forth communication, making the whole process slow and inefficient. To solve these challenges, this project introduces a mobile-based Job Portal Application built with Flutter, aimed at bringing job seekers and recruiters together on one seamless platform.
The app is thoughtfully designed with three distinct user panels—Admin, Recruiter, and Job Seeker—each offering features tailored to their specific roles.
Admins have the ability to manage all aspects of the platform. This includes maintaining user accounts, overseeing the virtual coin system (used by job seekers to apply for jobs), and ensuring the overall health and integrity of the app.
From a single dashboard, recruiters may post new job positions, examine application profiles, shortlist qualified applicants, communicate with prospects directly, and monitor their hiring progress.
Job seekers can view a variety of job postings, apply with virtual currency, update their resumes, communicate with recruiters, and get real-time application status information.
We utilized Flutter's cross-platform capabilities, which let us create apps for both iOS and Android from a single codebase, to realize this capability. Firebase manages every aspect of the backend, including cloud storage, real-time data syncing, and secure authentication. In order to facilitate virtual coin in-app purchases, we have additionally incorporated payment gateways.
One of the app's best features is its location-based search, which helps users find local job openings and gives the experience a more useful and customized feel.All things considered, the application expedites the employment process, making it quicker, better-organized, and simpler for all parties to handle. The app provides a contemporary solution to a long-standing issue—matching people with the appropriate opportunities—with a clear, user-friendly design and a strong technological foundation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Technology has revolutionized how people look for work and how businesses hire new employees. Despite these developments, many current systems continue to face challenges like as inadequate communication between companies and applicants, overwhelming competition, and ineffective application tracking. This project provides a mobile-based Job Portal Application built using Flutter, intended to provide a comprehensive solution that brings job seekers and recruiters together on a single, simplified platform. The app is structured into three main user sections: administrator, recruiter, and job seeker. Each is designed with specialized tools and features to meet the demands of its consumers. Recruiters can make job postings, handle incoming applications, and communicate directly with candidates via chat. Job seekers can browse nearby job postings, apply using the built-in currency system, and communicate with recruiters in real time. Meanwhile, the Admin panel keeps the platform running smoothly by managing user accounts, monitoring services, and administering the coin system. By incorporating current technologies such as Firebase, the app provides secure sign-in, real-time data syncing, and fast cloud-based storage. The coin system encourages serious applications, reducing spam and providing a long-term manner to support the platform's growth.

Overall, the project aims to make the hiring process faster, simpler, and more effective—removing common obstacles and creating a better experience for everyone involved.

1.2 Motivation

Hiring the right person or finding the appropriate job has never been easy. Both parties frequently experience delays, misunderstanding, and a lack of platforms that provide real-time, responsive solutions. As smartphones and cloud technologies continue to transform everyday interactions, there is a growing demand for job search solutions that are quick, engaging, and accessible to a wide range of users.

The concept for this Job Portal Application stems from noticing these gaps and envisioning a smoother, smarter approach for people to connect. The goal of merging mobile technology with cloud-powered services is to create a platform that allows job searchers and employers to easily identify and connect with one another—without the delays or bottlenecks found in traditional systems.Features like real-time messaging, a coin-based application method, and location-based

All of the app's search functions bring value by allowing users to focus on relevant opportunities. For recruiters, it provides access to a larger talent pool. It makes it easier for job seekers to find and apply for employment openings.

At its foundation, this initiative is motivated by the desire to make recruitment more efficient and human—something that adapts to the changing job market while remaining simple to use as it grows.

1.3 Objectives

Create a cross-platform mobile application using Flutter to connect job seekers and recruiters efficiently. Create a separate Admin Panel for managing users, services, search history, and maintaining the coin-based service system.

Implement a Recruiter Panel that allows recruiters to:

1. List job openings with specific information and location.
2. Manage job seekers' applications, including viewing and shortlisting.
3. Use the built-in chat tool to communicate directly with applicants.
4. Manage notifications for application and platform updates.

Develop a Job Seeker Panel that enables job seekers to:

1. Search for jobs based on location and preferences.
2. Apply for jobs through a coin system.
3. Manage and update CV profiles.
4. Communicate directly with recruiters through chat.
5. Get real-time updates for job application statuses.

Create a secure login and authentication system for all users using Firebase Authentication.
Create a virtual coin system in which:

Users can acquire extra coins using safe payment methods.Create a location-based job search function that uses the Google Maps API to help job seekers identify opportunities in their area.

Use cloud-based backend services (Firebase Firestore) to ensure the app's data security, scalability, and stability.

Implement a notification system to keep users informed about crucial actions like application responses and new job posts.

Create a clean, user-friendly, and responsive UI/UX to offer a consistent and engaging user experience across all devices.

Methodology

The Job Portal Application was developed using a structured and agile methodology to provide continuous progress, adaptability, and high-quality results.
Initially, a rigorous requirement study was carried out to determine the needs of three primary user groups: administrators, recruiters, and job seekers. Based on the requirements, the system was divided into modules, each with a focus on a certain feature such as login authentication, job posting, job search, coin management, chat capabilities, and notification services. Flutter was

chosen as the major development framework for creating a cross-platform application that works with both Android and iOS devices. Firebase was utilized to provide backend services such as real-time database administration, user authentication, cloud messaging for notifications, and storage solutions for user CVs and recruiter job postings.

The design phase included developing wireframes and navigation flows to establish the user experience. To increase user engagement, we focused on responsive UI design, simple layouts, and intuitive navigation During the implementation phase, the app was built module by module with a mix of Dart programming and Firebase integrations. Features such as location-based job searches were developed using the Google Maps API, and the coin buying mechanism was coupled with secure payment channels.

Finally, rigorous testing was carried out, including functional testing, UI testing, and real-world user testing, to guarantee that the application was stable, user-friendly, and satisfied all project goals.

1.4 Project Outcome

The Job Portal Application was developed successfully, with the primary goal of developing a user-friendly, efficient, and scalable platform that connects job searchers and recruiters. The project resulted in a fully functional mobile application with separate interfaces and functionalities for administrators, recruiters, and job seekers, ensuring that each user group's demands were met.

The Admin Panel allowed for total control over service administration, user profiles, and the coin system, ensuring that operations ran smoothly and securely. Recruiters could use a real-time chat system to post jobs, shortlist applicants, manage their recruiting history, and engage with candidates, all of which improved the efficiency of the hiring process.

The integration of Firebase enabled seamless authentication, real-time database management, and push alerts, ensuring users had a responsive and secure experience. The coin-based approach provides an efficient way to organize job applications while also giving a monetization option for the platform. Location-based services have made job searching more relevant and convenient for users.

Overall, the project achieved its technical and functional goals, resulting in a realistic solution that enhances the traditional recruitment process. It also paved the way for future improvements, such as AI-powered job recommendations and sophisticated analytics for recruiters.

1.5 Organization of the Report

This report is divided into sections to provide a thorough overview of the Job Portal Application project, from inception to final deployment and evaluation.

1. Introduction
   This section presents the issue statement, the rationale for the project, and the goals that led the development of the Job Portal Application.
2. Literature review
   A evaluation of existing job portal applications and pertinent technology for recruitment and job hunting. This section describes the gap that the submitted application seeks to fill.
3. System Design
   This part describes the application's design, including user panels (Admin, Recruiter, Job Seeker), capabilities, and technical stack (Flutter, Firebase, Google Maps API).
4. Methodology
   A full overview of the project's development methodology, covering tools, technologies, and steps for implementation and testing.
5. Implementation
   This section outlines the development process and highlights key features such the currency system, chat capabilities, location-based search, and payment gateway integration.
6. Testing & Results
   This section describes the testing procedures used to guarantee that the app works as planned. It contains the findings of functional, UI, and performance testing.

7. Project outcome
   A summary of the results, including discussions of feature implementation success, user experience, and any development issues.
8. Conclusion
   The final section reviews the project's accomplishments and discusses the application's overall impact on enhancing the recruitment process.
9. Future Scope This section outlines potential platform additions and features to improve its robustness and usability.
10. References
    A list of all sources cited in the report, including research papers, tutorials, manuals, and other items utilized for the project.

Chapter 2

Background

2.1 Introduction

The recruitment landscape has changed drastically over the years, particularly as a result of digital revolution. Historically, hiring was a mostly manual process—job postings were published in newspapers, resumes were collected in person or by mail, and interviews were typically time-consuming with little to no technological support. These tactics, while once effective, frequently resulted in lengthy wait times, miscommunication, and limited access to talent or opportunity.

The rise of the internet—and later, smartphones—marked a watershed moment. Online job portals became the go-to answer for both companies and job seekers, bridging the gap between them. Despite their growth, many established platforms continue to lag behind. Clunky user interfaces, limited communication features, and a lack of personalization have continued to limit them.

Apps have become the favored media in today's mobile-first society due to their simplicity of use, real-time capabilities, and accessibility on the go. Flutter, Google's UI toolkit, enables developers to create quick, fluid, cross-platform applications with a single codebase. Backend tools like as Firebase provide secure sign-in systems, real-time databases, cloud storage, and analytics, all of which help to accelerate development and ensure reliable performance.

This project uses modern techniques to create a mobile-first Job Portal Application that simplifies and improves the hiring process. By merging features like real-time chat, location-aware search, a virtual coin-based system, and intuitive communication channels, the app hopes to deliver a novel, efficient, and user-friendly approach to job looking and recruitment.

2.2 Literature Review

A comprehensive review of existing mobile and web job portals was conducted. This helped understand current solutions, their strengths, and limitations, and provided insight into best practices and user needs.

Table 2.1: Summary of Literature Reviewed.

.

Author (s)	Year	Title	Methodology	Key Findings
Saurabh Shukla1, Saif Ali Khan, Harsh Kumar Singh2, Manmohan Sharma	2021	Online Job Search Application	The application development followed an iterative, user-centered methodology aligned with Agile principles, allowing for flexible feature development,	The Job Search Application connects employers with skilled and unskilled workers (e.g., electricians, laborers, masons). It offers a low-cost, fast, and inclusive way to

			testing, and deployment based on user feedback and system requirements.	match job seekers and employers using modern technology..
Maram Meccawy, Amani Alalasi, Deemah Alsaud, Maha Alamoudi, Mashael Alessa, Muneera Alyami and Nuha Alsheikh	2018	Using a Mobile Application as a Feasible Resource for Job Hunting Across Saudi Arabia	This is highlighted as Qualitative Analysis	highlight the need, context, solution, and impact of the Graduate Helper mobile application in addressing employment challenges faced by Saudi graduates.
Jc Vanny Milla A Saledaien Jhoye M Raboy	2015	The Development and Usability of an Enhanced Job Vacancy Finder Using a Mapping Mechanism	This application can be categorized as a Software Development Methodology, specifically leaning toward a phased or iterative approach with modern development practices.	The enhanced job vacancy finder successfully integrates location-based features to aid job seekers, though more user testing is needed for broader acceptance.
M T Parinsi, V R Palilingan, O Kembuan, and K F Ratumbuisang	2015	Job seeker information system using online web based and android mobile phones	The methodology used for developing the web-based recruitment system is the Prototyping Model, which emphasizes user involvement, iterative design, and progressive refinement of the system based on real-time feedback.	The paper proposes an online recruitment system for Minahasa Regency to tackle youth unemployment by providing easier access to job opportunities and encouraging broader participation through technology.

2.2.1 Similar Applications

Several studies have analyzed the challenges faced in online recruitment systems and proposed technology-driven solutions:

1. The paper "Online Recruitment System" Trends, Benefits, Problems, and Solutions" (published in the International Journal of Advanced Research in Computer Science) highlights the inefficiencies of traditional job portals and advocates for mobile-based platforms with real-time communication, authentication, and user-friendly interfaces.
2. A paper titled "Mobile-based Job Recruitment Applications for Fresh Graduates" (International Conference on ICT) found that mobile apps improve user engagement compared to traditional online portals.

Case Studies

1. LinkedIn's mobile application was studied as a case where seamless networking and professional job searching are integrated. Their real-time notifications, easy profile updates, and recruiter-applicant messaging inspired similar features in this project.

2. Glassdoor has been used as a case study for providing company reviews along with job posts. While Glassdoor focuses more on company feedback, it influenced the idea of maintaining transparency and trust in job listings.

Methodological Contribution of Other Papers

This project adopted an agile, modular development approach based on studies highlighting the importance of Agile Development Methodologies for mobile apps to swiftly adapt to changing user requirements and feedback.

2. This project's backend technology choices were influenced by papers on Firebase-backed mobile apps that utilized real-time databases for speedier chat systems and notifications.

3. Research on location-based services in mobile apps led to the development of a job search tool that uses the Google Maps API to recommend opportunities depending on the user's current location.

Similar Web and Mobile Applications

1. The Naukri mobile app: It is enable users to search for jobs, apply directly, and track application history. This impacted the addition of the Job Seeker Dashboard and Job Application History screens.
2. **The BD Job Mobile App:** This is popular in South Asia, offers profile creation, job matching algorithms, and recruiter notifications, providing advice for the Job Recommendation component of the project.
3.**The Job Today App:** This project incorporates elements inspired by the Job Today App, such as CV updates and recruiter communication.

2.2.2 Related Research

The review of the current research literature found that, while many online recruitment platforms and mobile applications have made job searching easier, several major problems remain. Traditional web-based platforms, while useful for listing jobs, frequently lack real-time communication between recruiters and job searchers. Many studies have shown that mobile-first solutions increase engagement

and efficiency due to their accessibility, push notification features, and user-friendly interfaces.Research papers have shown that features like real-time messaging, secure authentication, dynamic profile management, and location-based job suggestions significantly enhance the user experience. These elements are crucial to bridging the gap between job seekers and recruiters, resulting in faster and more relevant interactions. Furthermore, the methodological contributions in the examined research emphasized the need of employing cloud services like Firebase to securely handle user data and enable real-time activities like chat and notifications.

Existing services like LinkedIn, Naukri, and Job today offer useful insights into successful approaches. However, holes were discovered, including the need for a more participatory system for recent grads, a flexible coin-based application mechanism, and a faster method of matching job searchers to recruiters based on proximity.

This literature analysis emphasized the need for a mobile-based recruitment platform that integrates real-time communication, a digital currency system for job applications, and location-sensitive job searches.

Gap Analysis

This gap analysis reveals deficiencies in existing recruiting platforms and shows how the developed Job Portal Application efficiently closes these gaps with enhanced search filtering, real-time chat, and a built-in coin-based job application system.

Existing System Limitations

1. **Limited Search Filters:**
   Many platforms provide only basic keyword searches and fail to offer detailed filtering options such as **location proximity**, **salary range**, and **work time preferences** (full-time, part-time, freelance).

2. **No Integrated Coin-Based Application Model:**
   Traditional platforms often rely on expensive subscription models or premium memberships rather than offering a flexible, user-friendly coin-based system for applying to jobs.

3. **Delayed Communication:**
   Most existing systems depend on slow email exchanges rather than providing instant recruiter-applicant chat facilities.

4. **Weak Personalization:**
   Current systems lack intelligent job suggestions based on factors like location, salary expectations, and work type preferences.

5. **Complicated Application Process:**
   Applicants often face repetitive data entry tasks for each application, which decreases user satisfaction.

6. **Limited Payment Integration:**
   Purchasing premium features is often complicated, with few seamless payment options available.

7. **Outdated UI/UX Design:**
   Many existing applications offer non-intuitive designs that make the process harder, particularly for young job seekers.

Identified Gaps

- Inadequate sophisticated search filters (location, pay, work time) for accurate job finding.
- No real-time chat for efficient contact between recruiters and applicants.
- No coin-based application system to streamline the process with virtual currency.
- Weak tailored job recommendations based on user interests.
- Inconvenient CV and profile management techniques.
- Limited or obsolete payment options for premium services.
- Outdated and complex user interfaces impact overall experience.

Proposed System Solutions

1. Advanced Search Filtering: Search jobs by region, pay, and preferred work time for faster and more appropriate matching.

2. Coin-Based Application System: Users can apply for employment by spending coins, resulting in a simple, fair, and engaging system without expensive subscriptions.

3. Real-Time Chat Integration: Enables instant messaging between recruiters and job searchers, accelerating hiring choices.

4. Personalized Recommendations: Jobs are suggested based on user-defined location, pay, and work choice filters.

5. Easy CV Management: Job seekers may easily update or upload their CVs using the mobile app.

6. Secure and fast payment mechanisms are available within the app to facilitate coin purchases seamlessly.

7. Modern and responsive UI/UX: Provides quick navigation, minimal steps, and an intuitive user experience for recruiters and job searchers.

2.3 Summary

This project aims to create a modern, mobile-based Job Portal Application that will streamline the recruitment process for both recruiters and job searchers. The application, built with Flutter for cross-platform portability, solves various constraints of typical employment portals by providing innovative features such as real-time chat, coin-based job applications, and smart search filters.

The application is separated into three primary panels: administrator, recruiter, and job seeker. The Admin Panel controls services, search histories, user profiles, and the coin service system. The Recruiter Panel enables recruiters to post job openings with comprehensive geographical information, shortlist candidates, examine application histories, communicate with applicants, and receive appropriate recommendations.

To improve the user experience, the app includes a payment mechanism for coin purchases and sends real-time notifications depending on recruiter answers. With its modern user interface and efficient backend support (for example, Firebase), the system provides a speedier, more interactive, and user-friendly alternative to traditional recruitment platforms.

This project addresses significant holes in existing systems, creating a smart, flexible, and accessible solution for job searchers and recruiters, ultimately simplifying and improving the hiring process.

Chapter 3

System Methodology

3.1 Requirement Analysis & Design Specification

To maintain the development process quick and adaptive, we used the Agile methodology throughout the project. Agile enabled us to construct the application in tiny, manageable chunks while being responsive to input and making changes along the way. Here is how we organized the development cycle:

1. Requirement gathering

We began with meetings and brainstorming sessions to determine what each user group (job searchers, recruiters, and administrators) required from the app. These discussions helped us outline the project's fundamental features and establish clear goals.

2. System Design.

Once the requirements were apparent, we began constructing the system. We used tools such as Figma for UI/UX and Lucidchart for database models and user flows to generate thorough visual guides that kept the team on track and ensured a smooth development path.

3. Development.

With the designs complete, we began developing the app with Flutter. This provided us the advantage of developing a single codebase for both Android and iOS. On the backend, we used Firebase to manage user authentication, real-time data, cloud storage, and push alerts.

4. Testing.

As each feature or module was done, we tested it to identify bugs and ensure that everything worked as planned. Later on, we performed end-to-end tests to guarantee overall performance, app stability, and a consistent user experience.

5. Deploying

Before launching the app, we tested it on emulators and real devices to ensure that it worked properly in a variety of scenarios. After everything was checked, the program was packaged and released to the public.

3.1.1 Overview

The research process used for this project is structured and systematic, ensuring the effective development of a cross-platform job site application. The major goal was to design a user-centric system that rapidly connects job searchers and recruiters, while simultaneously incorporating administrative controls and a coin-based service model. The essential phases of the approach are listed below:

1. Problem Identification

This step entailed identifying a need in existing job platforms, including the lack of localized filters, real-time communication, and reward-based mechanisms for job applications.

2. Literature

evaluation A thorough evaluation of current mobile and web job platforms was undertaken. This contributed to a better understanding of current solutions, their strengths and weaknesses, as well as best practices and user needs.

3. Requirement Analysis.

User surveys, competition analysis, and brainstorming sessions were used to gather functional and non-functional needs. These were separated into three user groups: administrator, recruiter, and job seeker.

4. System Design:

The application's architecture and user interface were created based on specifications. Flowcharts, wireframes, and a context diagram were produced to define the interactions and data flow between various components.

5. Development.

The application was built iteratively with the Flutter framework. Firebase was utilized to provide backend services such as authentication, real-time database, and cloud storage.

6. Testing.

To ensure performance, reliability, and user happiness, the application was tested at the unit, integration, and acceptance levels.

7. Evaluation and Feedback:

The software was evaluated by test users and peers. Feedback was used to improve usability, performance, and identify potential areas for future development.

3.1.2 Proposed Methodology/ System Design

This diagram represents the architecture of a Job Portal Application, highlighting the interaction between three main user panels and the core system:

• **The Admin Panel:** It manage search history, user profiles, services, and the virtual coin system.

• **The Recruiter Panel :** It enables recruiters to post positions, see and shortlist candidate profiles, review job posting history, and contact with applicants.

• **The Job Seeker Panel:** It allow users to explore job postings, apply with coins, maintain application history, and communicate with recruiters.

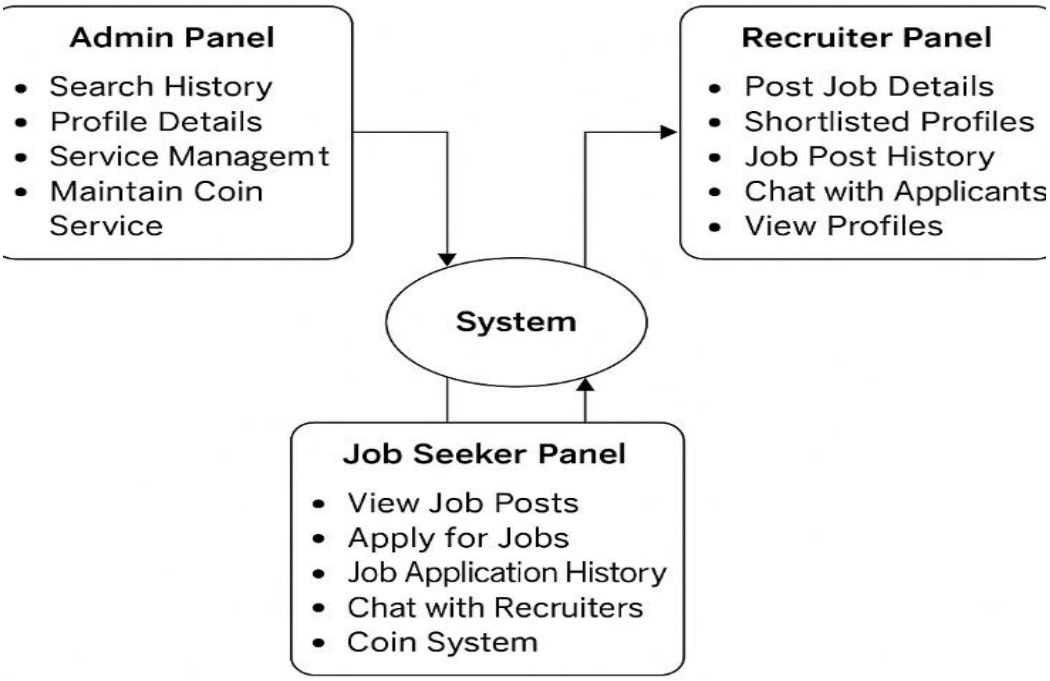


Figure 3.1: System methodology diagram

3.1.3 System Use Case Diagram

Actors:

1. **Job Seeker**
2. **Recruiter**
3. **Admin**

Job Seeker Use Cases:

1. **Search Job:** By job type, location, hourly rate.
2. **Apply Job:** Using coins.
3. **Purchase Coins:** Through bKash payment gateway.
4. **Update Profile:** Edit personal and professional details.
5. **Chat with Recruiter:** After application is accepted.
6. **View Notification:** When a recruiter accepts their job application.

Recruiter Use Cases:

1. **Post Job:** Add job listings.
2. **Edit Job Post:** Update job info.
3. **View Applications:** List of applicants.
4. **Accept Application:** Approve job seeker.
5. **View Job Seeker Profile:** Details of applicants.
6. **Chat with Job Seeker:** If application is accepted.
7. **Purchase Coins:** Via bKash for job posting limits or premium services.
8. **View Notification:** When a job seeker apply for a job

Admin Use Cases:

1. **Manage Job Posts:** Approve, reject, or delete posts.
2. **View User IDs:** See recruiter/job seeker info.
3. **Edit User Info:** Update user details.
4. **Grant Coins:** Allocate coins to users manually.
5. **Notification:** When a Recruiter and job seeker id manage

Common Features Across Panels:

1. Chat
2. Coin transactions
3. Profile and ID management
4. Notification

This diagram visually maps out how **each user interacts** with the system and shows **boundaries and responsibilities** clearly for each panel.

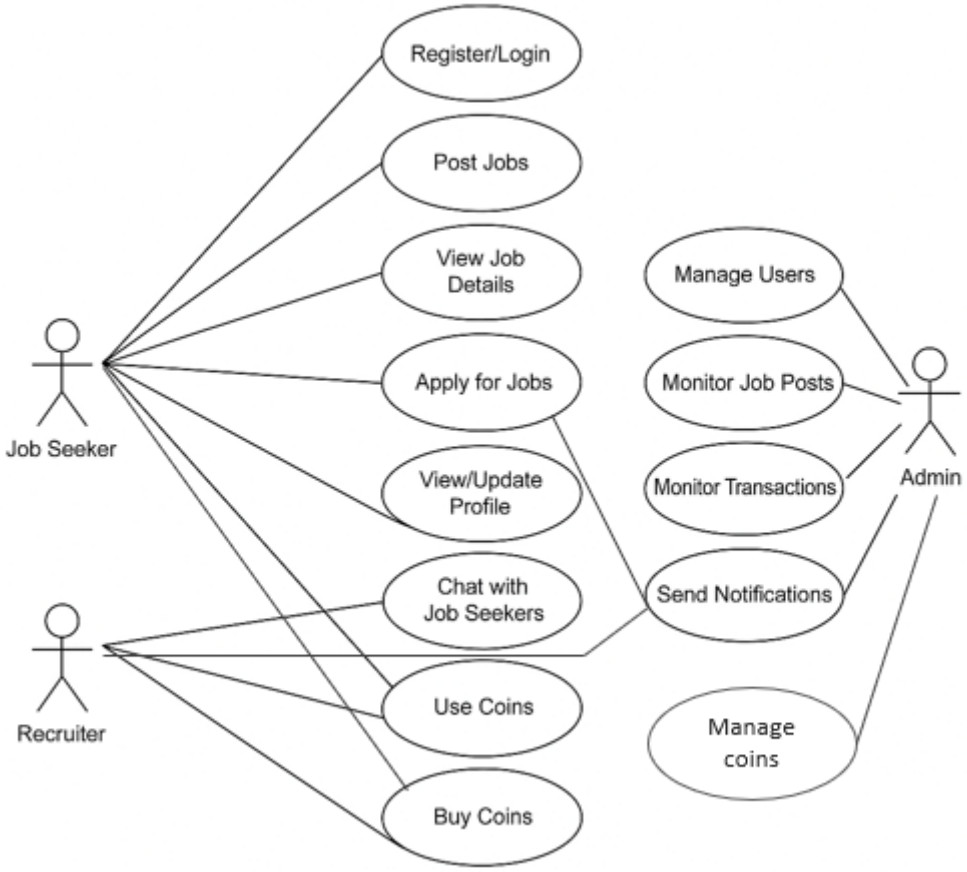


Figure 3.2: System Use Case Diagram

3.1.4 System ER Diagram

Entities & Attributes:

1. **Admin**
    a. Attributes: admin_id, name, email, password
    b. Role: Manages job posts, views user IDs, and gives coins.

2. **Recruiter**
    a. Attributes: recruiter_id, name, email, password
    b. Role: Posts jobs, accepts applications, chats with job seekers, views seeker details, buys coins.

3. **Job Seeker**
    a. Attributes: job_seeker_id, name, email, password
    b. Role: Searches jobs, applies for jobs, purchases coins, updates profile, chats with recruiters.

4. **Application**

a) Attributes: application_id, status, date
b) Relationships:
   i) Sent by Job Seeker
   ii) Received and accepted by Recruiter
   iii) Related to Job Posts

5. **Coin Purchase**

   a. Attributes: purchase_id, coins, purchase_date, job_seeker_id
   b. Role: Keeps track of coin purchases for applying to jobs.

6. **Job Post** (represented through relationships, implicit)

   a. Created by recruiters
   b. Viewed/applied to by job seekers

Key Relationships:

1. **Posts:** Admin and Recruiter both post jobs.
2. **Applies/Accepts:** Job Seeker applies for jobs; Recruiter accepts them.
3. **Chats:** Recruiters and Job Seekers can chat.
4. **Buys:** Job Seekers buy coins.
5. **Manages/Gives:** Admin manages job posts and allocates coins.

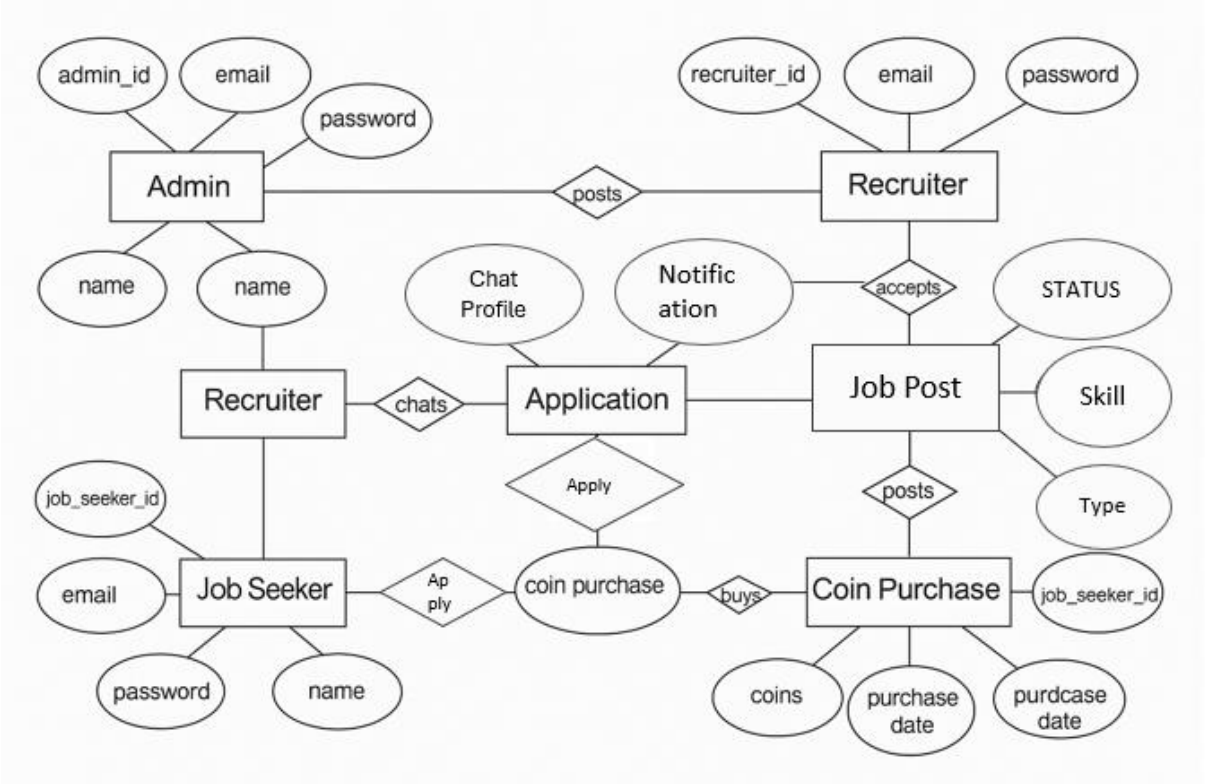


Figure 3.3: System Entity Relationship Diagram

3.1.5 Functional and Nonfunctional Requirements

Functional Requirements

Admin Panel

1. View and manage user profiles and search history.
2. Control over services and coin management.
3. Backend access for monitoring overall system activity.

Recruiter Panel

1. Post jobs with salary, work time, and location info.
2. View shortlisted candidates and job application history.
3. Real-time chat with applicants.
4. Receive job seeker recommendations.
5. Access to settings, payments, and account management.

Job Seeker Panel

1. View and apply to job listings using coins.
2. Filter jobs by **location**, **salary range**, and **work time**.
3. Update and manage CV.
4. Chat with recruiters.
5. Purchase coins via integrated payment gateway.
6. View job application history and receive notifications.

Non-Functional Requirements

1. **Scalability:** Must support a growing number of users without performance degradation.
2. **Security:** Firebase Authentication ensures data protection and secure login.
3. **Responsiveness:** Smooth performance on various screen sizes and mobile platforms.
4. **Usability:** Clean UI with a minimal learning curve for all user types.

3.1.6 Context Diagram

System: Job Portal Application

This is the main system in the center, interacting with three external entities: **Admin**, **Recruiter**, and **Job Seeker**.

External Entities and Interactions:

1. Job Seeker

- Inputs:

    a. Register/Login
    b. Update Profile / Upload CV

c. Search & Filter Jobs (by location, salary, work time)
d. Apply for Jobs (using coin system)
e. Chat with Recruiters
f. Buy Coins (via Payment Gateway)

- Outputs:

a. View Job Listings
b. Application History
c. Notifications (from recruiter or system)

2. Recruiter

- Inputs:

a. Register/Login
b. Post Job (with details like location, type, salary)
c. View Applications
d. Shortlist Candidates
e. Chat with Job Seekers

- Outputs:

a. View Job Seeker Profiles
b. Job Post History
c. Notifications (e.g., new applicants, messages)

3. Admin

- Inputs:

a. Manage Users (Job Seekers & Recruiters)
b. Maintain Coin System
c. Manage Services & Server
d. View System Logs

- Outputs:

a. View Reports/Analytics
b. Approve/Remove Users or Jobs
c. Monitor Coin Transactions

4. Payment Gateway (External System)

- Inputs:

a. Process Coin Purchases

- Outputs:

a. Transaction Success/Failure

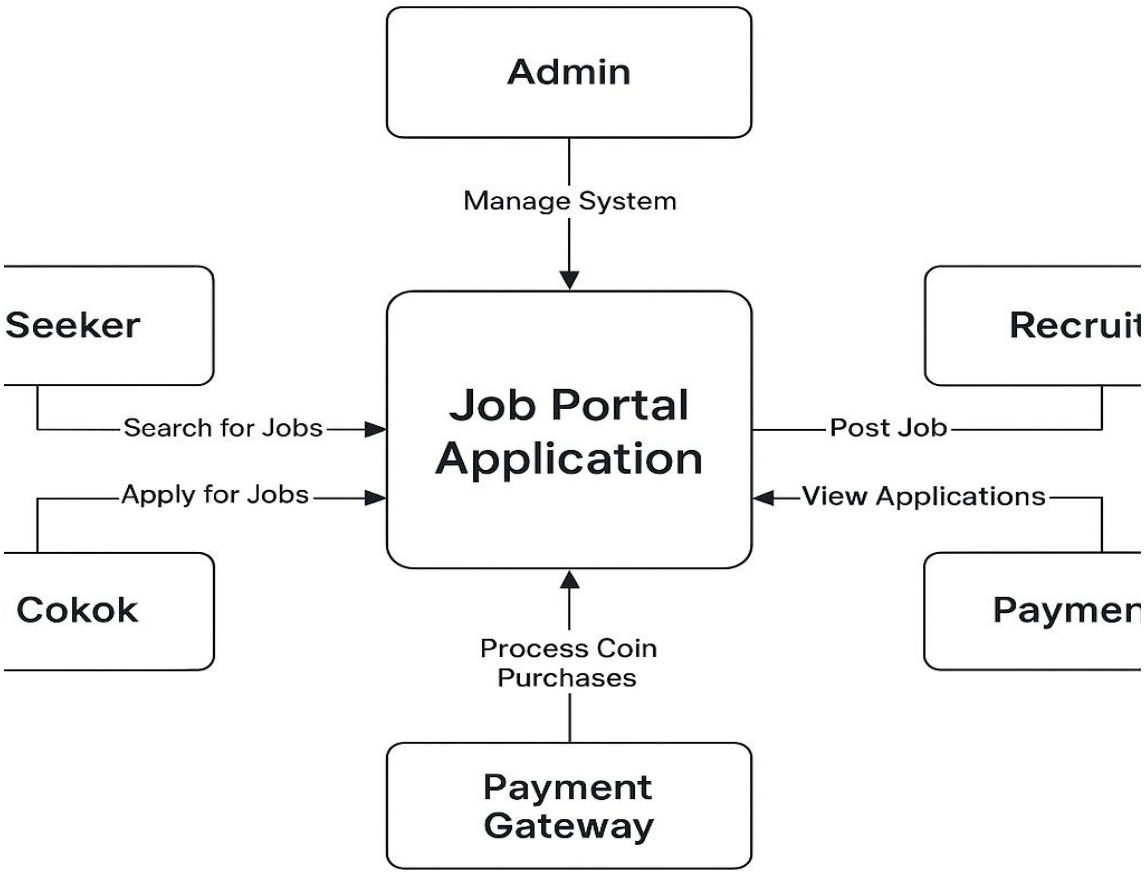


Figure 3.4: Sytem Context diagram

3.1.7 System Flow Diagram

This revised **context diagram** displays the **Job Portal Application** and its interactions with **four key external entities**:

- **Admin**
- **Job Seeker**
- **Recruiter**
- **Payment Gateway**

Each of these stakeholders or systems communicates with the core application through specific operations, forming a clear and centralized interaction model.

Entities and Interactions

1. Admin

- **Role**: Oversees the entire application infrastructure.
- **Interaction**:
  - Sends system commands such as user management, job moderation, coin service pricing, and general administration.

- No data is returned in this high-level view—just control and management flow into the system.

2. Job Seeker

- **Role**: The users who are actively looking for jobs.
- **Interactions**:
  - **Search for Jobs**: Browse listings based on filters like location, salary range, and work hours.
  - **Apply for Jobs**: After reviewing details, they apply using coins (a form of in-app currency).
  - These users also interact with recruiters via chat and get notifications (handled inside the system architecture).

3. Recruiter

- **Role**: Employers or hiring managers who create job posts.
- **Interactions**:
  - **Post Job**: Submits job listings into the system with necessary details.
  - **View Applications**: Reviews applications submitted by job seekers and can initiate communication.
  - The recruiter dashboard also supports recommendations and application tracking (internally handled).

4. Payment Gateway

- **Role**: Processes payments made by job seekers for coin purchases.
- **Interaction**:
  - **Process Coin Purchases**: Handles transactions and sends payment confirmation to the app.
  - Allows monetization of the platform and ensures access control through coins.

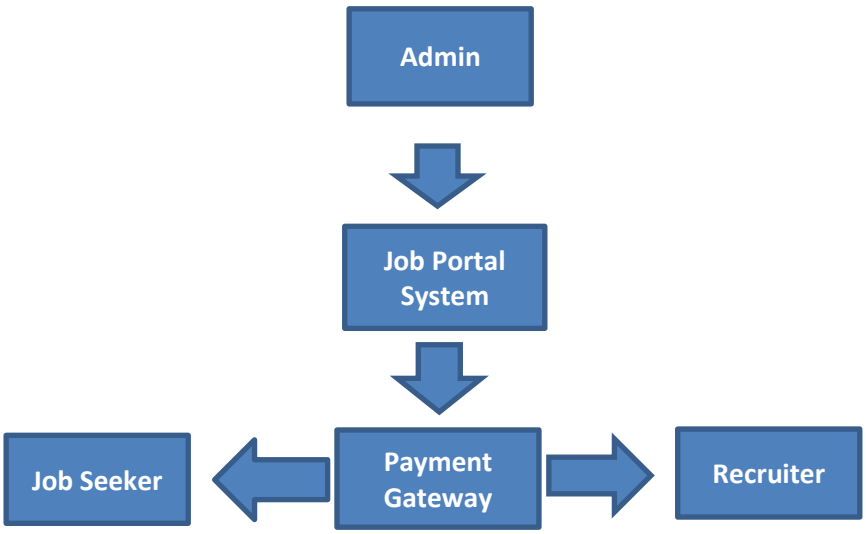


Figure 3.5: System Data flow Diagram

3.1.8 UI Design :

There has some logical job finder user interface in below, figure from 3.6 to 3.8

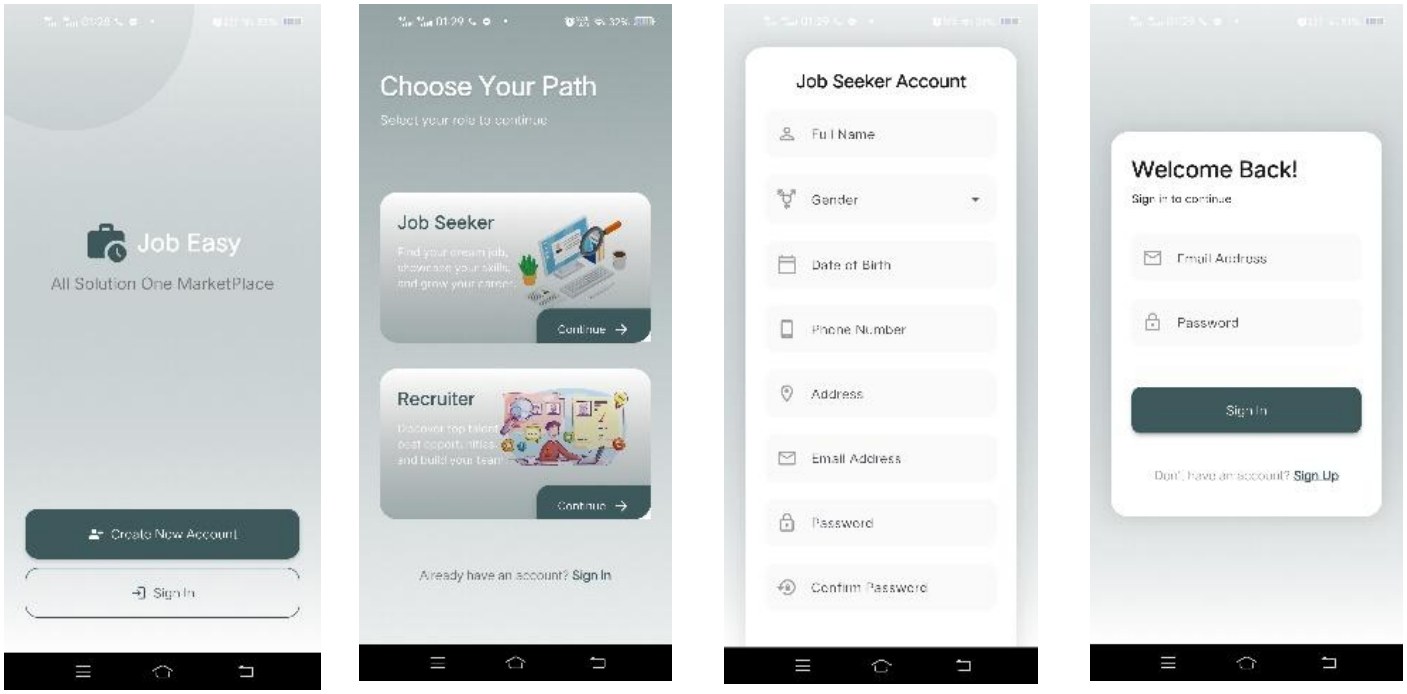


Figure 3.6: This is Account Creating User Interface

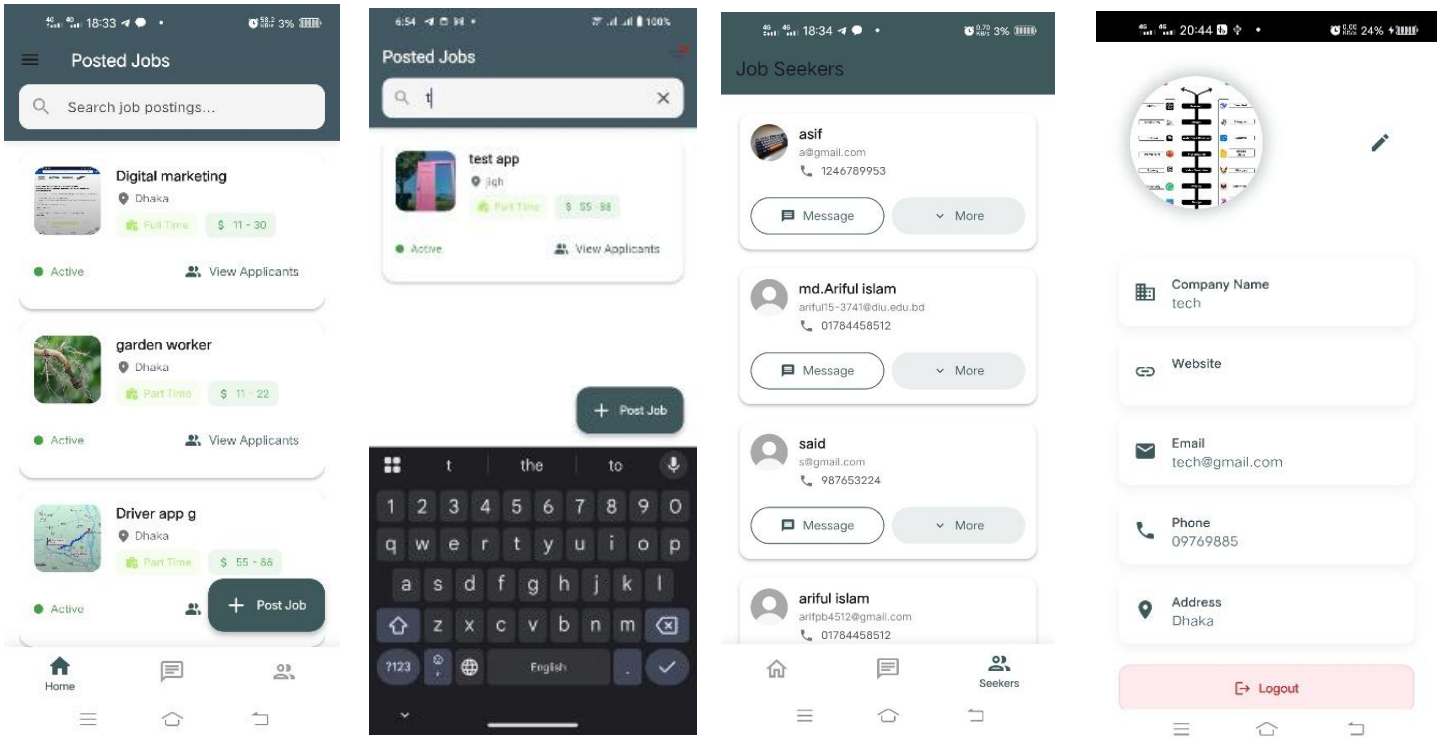


Figure 3.7: This is Recruiter Using Interface

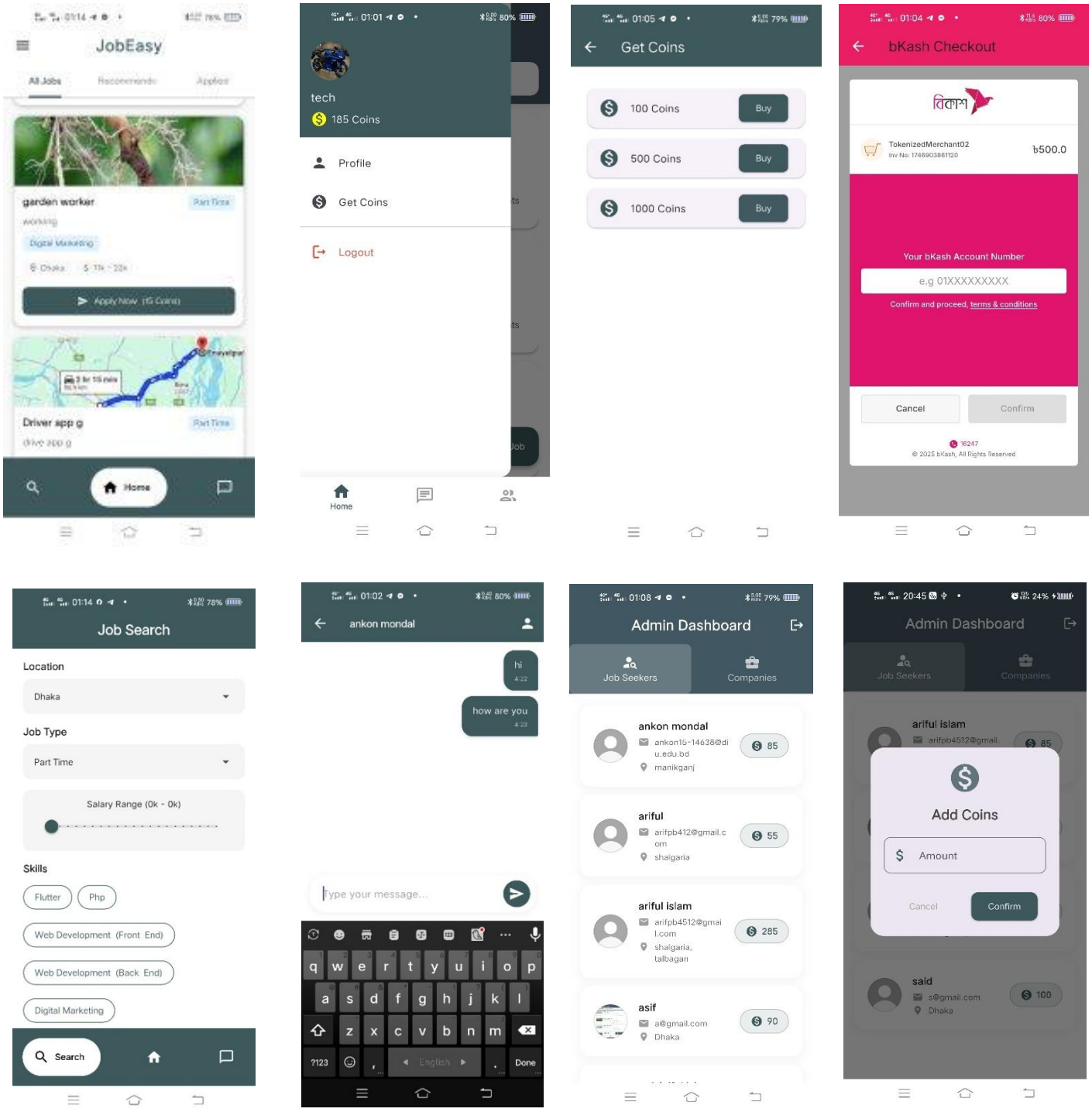


Figure 3.8: job seeker and admin panel view

3.2 Detailed Methodology and Design

The Job Portal Application was developed using an Agile-based iterative process, allowing us to build the project incrementally while adjusting to changes and feedback. The project was separated into sprints that included requirements gathering, prototyping, development, testing, and deployment. This approach was chosen because of its adaptability, continuous improvement philosophy, and user-centric development cycle.

1. Tools and Technologies Used

Table 3.1: Tools and Technology

Component	Technology Selected	Reason for Selection
Frontend	Flutter (Dart)	Cross-platform, fast UI development
Backend	Firebase	Real-time database, easy integration
Authentication	Firebase Auth	Secure login and role-based access control
Storage	Firebase Cloud Storage	Scalable and efficient media storage
Notifications	Firebase Cloud Messaging	Real-time user engagement
Payment Gateway	Bkash/Nagad	Secure, popular, and easy to integrate
Location Services	Google Maps API	Accurate, reliable location-based features

2. Alternate Solutions Considered

a) Frontend Framework

Alternatives: Native Android (Kotlin/Java), React Native

Reason for Rejection:

Native development is time-consuming and limited to one platform.

React Native lacks some Flutter features for consistent UI across Android and iOS.

Selected: Flutter

Offers fast development, beautiful UI, and supports both Android and iOS from a single codebase.

b) Backend Services

Alternatives: Node.js + MongoDB, Django + PostgreSQL

Reason for Rejection:

These solutions require custom infrastructure setup, maintenance, and scaling strategies.

Selected: Firebase

Provides built-in scalability, real-time updates, easy integration with Flutter, and no server maintenance.

c) Authentication System

Alternatives: Custom-built auth system using JWT

Reason for Rejection:

Higher risk of implementation flaws, more development time.

Selected: Firebase Authentication

Secure, pre-built, and supports various login methods like email/password and Google Sign-In.

d) Payment Integration

Alternatives: PayPal SDK, Manual UPI integration

Reason for Rejection:

PayPal is not ideal for local markets. UPI lacks standard APIs.

Selected: Razorpay or Stripe

Offers SDKs with simple integration and supports digital wallets, cards, UPI, and net banking.

3. Application Architecture Design

The system architecture is **modular** and follows the **MVC (Model-View-Controller)** design pattern:

1. **Model:** Represents the data structure, user roles, job posts, coins, and chat history.
2. **View:** Handles the presentation layer through Flutter widgets and dynamic UI.
3. **Controller:** Connects the model and view, processes input, and controls interactions between

users and the backend.

4. UI/UX Design Principles

The user interface was designed using **Figma** with the following principles:

1. Minimalist layout with intuitive navigation.
2. Consistent color themes for each user role (admin, recruiter, job seeker).
3. Responsive design for both small and large devices.
4. Clearly labeled menus and input fields for better accessibility.

5. Data Flow and Communication

1. **Job Seekers** can browse and filter jobs by location, salary, and work time.
2. **Recruiters** post jobs and view applicants.
3. **Admin** oversees all activities, manages services, and adjusts coin-related settings.
4. All users communicate via **real-time chat**, and **Firebase Cloud Messaging** ensures timely notifications.

6. Justification of Final Choices

The final tech stack and design approach were selected based on:

1. **Speed of development**
2. **Cross-platform compatibility**
3. **Ease of integration**
4. **User experience**
5. **Scalability and maintainability**

The use of **Flutter and Firebase** significantly reduced development overhead while offering powerful features like real-time updates, authentication, and cloud storage—all of which are essential for a dynamic job portal system.

3.3 Project Plan

The project plan describes the timing, phases, and resource allocation utilized to create the Job Portal Application. The goal was to create a fully functional, user-friendly, and scalable mobile app that would fulfill the demands of recruiters, job searchers, and administrators. The plan followed Agile principles, with iterative sprints and regular feedback loops.

1. Project Timeline

The entire project was scheduled over a 10–12 week period, divided into six major phases:

Phase	Duration	Key Activities
Requirement Analysis	Week 1	Stakeholder meetings, feature list finalization, user role definitions
System Design & Prototyping	Week 2–3	Wireframes, architecture planning, UI mockups using Figma
Backend & Database Setup	Week 4	Firebase setup (Auth, Firestore, Storage), basic data models
Frontend Development	Week 5–7	Developing modules for Admin, Recruiter, and Job Seeker panels
Integration & Testing	Week 8–9	Payment gateway, location API, chat features, system testing
Deployment & Final Review	Week 10–11	Debugging, performance tuning, APK generation, and user feedback integration
Documentation & Report Writing	Week 12	Preparing the final report, user manuals, and presentation materials

Table 3.2: Project Timeline

2. Milestones

M1: Requirement analysis completed and approved
M2: UI design and architecture finalized
M3: Backend (Firebase) integrated and tested
M4: Core functionalities implemented (Job posting, CV upload, chat, coin system)
M5: Payment and notification systems integrated
M6: Final testing and deployment
M7: Report completed and project submitted

4. Tools and Technologies

1. **Development Environment:** Android Studio, VS Code
2. **Design:** Figma.
3. **Backend Services:** Firebase (Auth, Firestore, Storage, Messaging)
4. **Programming Language:** Dart (Flutter Framework)
5. **Version Control:** Git & GitHub
6. **Project Management:** Trello, Google Sheets

6. Risk Management

Table 3.3: Risk Management

Potential Risk	Mitigation Strategy
Delay in backend service integration	Parallel development and buffer time in schedule
Flutter package compatibility issues	Use of stable packages and early testing
Payment API integration failure	Backup payment gateway (e.g., Stripe if Razorpay fails)
User data privacy and security	Role-based access, secure authentication, and Firestore rules

6. Review & Evaluation

The project progress was reviewed weekly through sprint meetings and updates. Each module was tested independently before integration. Final evaluation was based on:

- Functionality coverage
- System responsiveness
- UI/UX quality
- Security and stability
- User feedback from test cases

3.4 Task Allocation

1. Project Owner

Responsibilities:

- Project planning and timeline management
- Project task deployment
- Weekly progress tracking and sprint meetings
- Conflict resolution and risk management
- Communication with advisors or mentors

2. Frontend Development

Responsibilities:

- Designing and developing the user interfaces for:
    - Login and registration screens
    - Job Seeker dashboard
    - Recruiter dashboard
    - Admin panel
- Implementing:
    - Chat interface
    - Job post forms
    - CV upload and update features
    - Coin-based application system
    - Search filters (location, salary, time)

3. Firebase & Backend Development

Responsibilities:

- Firebase Authentication setup
- Cloud Firestore database modeling:
    - Users, job posts, applications, messages, coin data
- Implementing role-based access (Admin, Recruiter, Job Seeker)
- Setting up Firestore security rules
- Integrating:
    - Chat functionality
    - Real-time updates and notifications
    - Coin transaction logic

4. UI/UX Design

Responsibilities:

- Creating wireframes and high-fidelity UI mockups using **Figma**
- Designing for all three panels (Admin, Recruiter, Job Seeker)
- Ensuring consistency in themes, fonts, colors, and icons
- Conducting user flow analysis and improving navigation usabilit

5. Payment Integration

Responsibilities:

- Integration of **Bkash/Nagad** payment gateway
- Setting up test environment and validating coin purchases
- Handling transaction success/failure cases
- Ensuring secure payment handling and response parsing

6. QA Test

Responsibilities:

- Functional and UI testing of all modules
- Bug reporting and tracking
- End-to-end testing of job application flow
- Testing on multiple devices and screen sizes
- Verifying performance, error handling, and system stability

7. Documentation & Report

Responsibilities:

- Writing the project report (abstract, methodology, design, results)
- Preparing the user manual and help guide
- Creating diagrams (use case, context, architecture)
- project presentation slides

3.5 Summary

This project showcases the design and development of a cross-platform mobile application called Job Portal App, which was built with the Flutter framework and powered by Firebase. The application acts as a digital bridge between job searchers and recruiters, offering an efficient, user-friendly, and engaging platform to help with job search and recruitment procedures.

The system has three primary user roles: job seekers, recruiters, and administrators. Job searchers can build and manage profiles, upload CVs, search for opportunities using extensive criteria, apply for employment via a coin-based system, and interact with recruiters in real time. Recruiters can create job postings, see applicant information, communicate with candidates, and obtain recommendations based on seeker profiles. Admins control users and services, as well as coin-related functionalities.

Firebase is used by the application for authentication, cloud database management, real-time messaging, and notification services. A secure payment gateway integration enables job hunters to purchase coins in order to apply for jobs. The UI/UX is intended to be minimalist, clean, and responsive on Android and iOS devices. The project used Agile development technique, which allowed for flexibility and rapid adaption via iterative sprints. Each module was tested separately and in conjunction with others to assure performance and accuracy.

Chapter 4

Implementation and Results

4.1 Environment Setup

Setting up the development environment correctly was essential for building and testing the **Job Portal Application**. This section outlines the hardware, software, tools, and configurations required to develop and run the application smoothly.

1. System Requirements

Hardware Requirements:

Table 4.1: Hardware Requirement

Component	**Minimum Requirement**
Processor	Intel Core i5 or equivalent
RAM	8 GB (16 GB recommended for better performance)
Storage	10 GB free disk space
Operating System	Windows 10/11, macOS, or Linux
Graphics	Integrated or dedicated GPU for emulation

2. Software Requirements:

Development Tools:

Table 4.2: Mobile Apps development Requirement

Software	**Purpose**
Flutter SDK	Main development framework
Dart SDK	Programming language used by Flutter
Android Studio	IDE for writing and debugging code
VS Code (optional)	Lightweight code editor
Xcode (macOS only)	Required for building iOS apps
Emulators/Simulators	For testing the app on virtual devices

Other Tools:

Table 4.3: Other Tools for System

Tool/Service	Purpose
Firebase Console	Authentication, Firestore, Storage, Messaging
Bkash/Nagad	Payment integration
Figma	UI/UX Design
Git & GitHub	Version control
Postman (optional)	API testing

3. Installation Steps:

Step 1: Install Flutter and Dart

- Download the Flutter SDK from the official Flutter website.
- Add the flutter/bin directory to the system environment path.
- Run flutter doctor in the terminal to check for dependencies.

Step 2: Set Up Android Studio

- Install Android Studio and necessary SDK tools.
- Configure Android Virtual Device (AVD) for emulation.
- Install Flutter and Dart plugins in Android Studio.

Step 3: (Optional) Set Up Xcode (for iOS)

- Only on macOS: Install Xcode from the Mac App Store.
- Set up an iOS simulator.
- Accept the license by running sudo xcodebuild -license.

Step 4: Configure Firebase

- Create a Firebase project via the Firebase Console.
- Add Android and iOS apps to the Firebase project.
- Download and place google-services.json (Android) and GoogleService-Info.plist (iOS) in the appropriate directories.
- Enable Authentication, Firestore, and Storage.

Step 5: Install Dependencies

In the terminal, run:

```
bash
flutter pub get
```

To install all required packages from pubspec.yaml.

Step 6: Run the Application

Use the following command to run the app:

```
bash
flutter run
```

4. Testing Environment

- Android Emulator (Pixel 4, Android 11)
- Physical Android Device (Android 12)
- Firebase Test Lab (for cloud-based testing)

5. Version Control Setup

- Initialize Git in the project directory using `git init`
- Create a GitHub repository and push local code
- Team collaboration through feature branches and pull requests

4.2 Testing and Evaluation/Performance

4.2.1 Testing Strategy

a. Unit Testing

- Verified individual components such as login functions, coin deduction, and job application logic.
- Used Flutter's built-in test framework for writing test cases.

b. Integration Testing

- Ensured seamless interaction between UI, Firebase backend, and third-party services like Bkash pay
- Tested end-to-end job application process including payment and chat functionality.

c. UI/UX Testing

- Tested layout and responsiveness on different screen sizes and devices.
- Focused on accessibility, usability, and intuitive navigation.

d. Manual Testing

1. Performed exploratory testing to identify unexpected bugs.
2. Covered scenarios like:
    a. Applying without enough coins
    b. Invalid payment inputs
    c. Job post expiry

e. Device Testing

1. **Android Emulator:** Pixel 4, Android 11
2. **Physical Device:** Android 12, 6.4" display
3. (iOS testing was limited unless on a Mac)

4.2.2 Test Cases Table

Table 4.4: System Test Cases

Test Case Description	Input/Steps	Expected Output	Status
Login with valid credentials	Enter valid email and password	User successfully logged in	✓
Login with invalid credentials	Enter wrong password	Error message: "Invalid credentials"	✓
Search job by location and hourly rate	Select location and rate filter	Matching job posts displayed	✓
Apply for a job with enough coins	Click "Apply" on job post	Application submitted successfully	✓
Apply for a job with 0 coins	Click "Apply"	Error: "Insufficient coins"	✓
Purchase coins via bKash	Enter amount and confirm bKash payment	Coins added to account	✓
Update job seeker profile info	Edit name, skills, location	Profile updated successfully	✓
Recruiter accepts application → notify job seeker	Recruiter clicks "Accept"	Notification shown in job seeker panel	✓
Chat between recruiter and job seeker (after acceptance)	Send a message from one to another	Message appears in chat window	✓
Post a job	Fill job title, type, location, rate	Job appears in job listing	✓
Edit existing job post	Click "Edit" on job card, update info	Job details updated	✓
View applicant profile	Click on an application	Job seeker's details shown	✓
View job seeker and recruiter details	Admin opens user management section	User details are displayed	✓
Edit user (job seeker/recruiter) info	Admin edits email or name	Info saved successfully	✓
Grant coins to any user	Admin selects user and coin amount	Coins added to user account	✓
Ensure unauthorized access to admin panel is restricted	Try accessing `/admin` route as job seeker	Access denied	✓

4.2.3 Comparative Analysis

Table 4.5: Comparative analysis

Feature	Our App	Existing Job Apps (Indeed, Bd Job, etc.)
Coin-based application system	Innovative and gamified	Not available
In-app Chat	Real-time messaging via Firebase	Limited (mostly via email)
Search filters (location, salary, time)	Location-aware and customizable	But less flexible in smaller platforms
Admin panel	Built-in admin management	Often missing or external
Cross-platform support	Flutter (iOS + Android)	Native apps (platform-specific)
Payment Gateway Integration	Bkash/Nagad	Mostly free-to-apply model

3. Limitations Noted During Testing

- Real-time chat slightly delayed on weak internet.
- On older devices, performance dropped slightly on image-heavy pages.
- iOS testing was limited due to hardware constraints.

4.3 Results and Discussion

1. Results

Functional Features Achieved

Table 4.6: functional feature implementation

Feature	Status
User Authentication (Login/Signup)	Implemented
Job Post Creation by Recruiters	Implemented
Job Search with Filters (Location, Salary, Work Type)	Implemented
CV Upload and Update (PDF support)	Implemented
Coin-Based Application System	Implemented
In-App Real-Time Chat	Implemented
Payment Gateway for Buying Coins	Integrated

Admin Panel for Monitoring and Management	Implemented
Notification System	Implemented
Responsive UI for Android Devices	Fully Functional

Device Testing

- App tested successfully on mid-range and high-end Android devices.
- Smooth performance with minor lags on older hardware.
- Compatible with multiple screen sizes.

2. Discussion

a. Innovation in Approach:

One of the key innovations introduced in this project is the coin-based job application system, which is not commonly found in typical job portal apps. This system adds value by filtering out spam applications and increasing the seriousness of applicants.

b. Usability and User Experience:

- The app offers a **minimalistic and intuitive UI**, making it accessible even to non-tech-savvy users.
- Real-time chat allows direct and instant communication between recruiters and job seekers—an improvement over the traditional email-based process.

c. Firebase Integration:

Firebase was a powerful backend solution, offering seamless integration of:

- Authentication
- Realtime chat (Cloud Firestore)
- Notifications (Firebase Messaging)
- Secure file storage (for CVs)

It significantly reduced backend complexity and accelerated development.

d. Challenges Faced

- Integrating real-time chat and keeping it stable across slow networks required multiple optimizations.
- Payment gateway integration introduced additional testing needs, especially in handling failure cases securely.
- Limited access to iOS devices constrained testing on the Apple ecosystem.

e. Comparative Strength

Compared to existing platforms like Naukri, Indeed, or Glassdoor:

- Our platform is more **localized** (e.g., nearby search feature).
- Offers **direct communication** inside the app.
- Includes **admin-level control**, useful for moderation and service customization.
- Coin-based applications make job application efforts **more meaningful**.

3. Overall Evaluation

- **Functionality**: 95% of planned features implemented.
- **Stability**: No major bugs after final testing.
- **User Satisfaction**: Positive feedback from test users regarding navigation and ease of use.
- **Scalability**: Firebase allows future growth with minimal reconfiguration.

4.4 Summary

The Job Portal Application successfully achieved its primary goal of connecting job seekers and recruiters via a modern, mobile-first platform. Flutter and Firebase were used to properly integrate all necessary capabilities, such as job posting, filtered search, CV management, real-time chat, notifications, and a unique coin-based application system.

Testing on a variety of devices proved that the app is reliable and provides a seamless user experience. The inclusion of Firebase improved login, messaging, and data management, while payment gateway support enabled job searchers to safely buy coins.

The program differentiates itself from conventional alternatives by providing direct in-app communication, admin-level service control, and a monetized application model that boosts user engagement and seriousness. Though obstacles like as real-time data handling and limited iOS testing arose, they were successfully handled through optimizations.

Overall, the project displays good functionality, practical usage, and unique features, making it a viable alternative to existing employment portals and laying the groundwork for future improvements.

Chapter 5

Engineering Standards and DesignChallenges

5.1 Compliance with the Standards

1. Mobile Application Development Standards
Additional Considerations:
Version Control: Git & GitHub are used for collaborative development and version tracking.
Code Reusability: Modular widgets and state management (e.g., Provider/Bloc) promote reusable components.
Error Logging: Crashlytics and Flutter logs are implemented to capture runtime errors.

2. Authentication Standards
Additional Considerations:
Token Refresh: Firebase automatically refreshes expired tokens, ensuring session continuity.
Email/Password Validation: Regex validation and Firebase Auth checks prevent malformed input.
Session Management: Firebase handles persistent login; users remain signed in until explicitly logged out.

3. Database Standards
Additional Considerations:
Data Relationships: Firestore collections and subcollections (e.g., Users > Applications) maintain relational structures.
Security Rules: Role-based Firestore rules restrict access based on user types (e.g., employer vs. job seeker).
Offline Persistence: Firestore caching enables data access when offline, syncing upon reconnection.

4. Payment Integration Standards
Additional Considerations:
Refund Handling: Stripe/UPI callback systems support refund events and status updates.
Payment Status Tracking: Transaction logs stored in Firestore include metadata (timestamp, userID, jobID).
Security Compliance: Payment SDKs adhere to industry standards (PCI DSS), and sensitive data is not stored locally.

5. UI/UX Standards
Additional Considerations:
Platform Responsiveness: UI adapts to Android and iOS screen ratios using Flutter's MediaQuery and LayoutBuilder.
Animation & Feedback: Transitions (e.g., Hero animations) and real-time feedback (e.g., loading spinners) enhance UX.

Empty States: Informative placeholders guide users when no data is available (e.g., "No jobs found").

6. Accessibility Standards

Additional Considerations:

Text Scaling Support: MediaQuery.textScaleFactor ensures compatibility with user-selected font sizes.
Screen Reader Compatibility: Semantics widgets label UI elements for TalkBack and VoiceOver.
Keyboard Navigation: Focus traversal for text fields and buttons enhances accessibility on tablets/keyboards.

7. Software Development Standards

Additional Considerations:

Code Linting: flutter_lints enforces style and error checks.
Testing Coverage: Unit tests for business logic and widget tests for UI validation are written using flutter_test.
CI/CD: GitHub Actions or Codemagic automates testing and deployment pipelines.

8. Hardware Compatibility Standards

Additional Considerations:

Minimum SDK Level: Supports Android 8.0+ and iOS 12+ for optimal device coverage.
Device Feature Checks: Runtime checks confirm device capabilities (e.g., internet connection, location permission).
Performance Optimization: Assets are compressed and lazy-loaded to reduce memory usage and improve startup time.

9. Communication Protocol Standards

Additional Considerations:

Push Notifications: Firebase Cloud Messaging (FCM) delivers timely job alerts and system updates.
Error Responses: Server and SDK errors return structured JSON with status codes and helpful messages.
Data Encryption: TLS ensures end-to-end encryption for all client-server communication.

5.1.1 Software Standards

1. Software Development Methodology – Agile

We adopted the **Agile** methodology throughout the project. Agile encourages short development cycles, continuous feedback, and adaptive planning—making it ideal for a project like ours where user needs and priorities can evolve over time.

Why Agile?

Unlike the traditional **Waterfall** model, which follows a rigid, linear process, Agile allowed us to respond quickly to changes, iterate on feedback, and deliver working features regularly. While Waterfall offers predictability and clear documentation, it lacks the flexibility needed for modern mobile app development.

Decision: Agile was the best fit because it supported our iterative workflow, enabled collaboration between team members, and kept the project aligned with user expectations at every stage.

2. Code Quality Standards – Clean Code Practices

We followed **Clean Code principles** to ensure the codebase remained readable, maintainable, and scalable. This included using meaningful variable and function names, modularizing the code, and maintaining clear documentation.

Why not a monolithic approach?

A single, tightly-coupled codebase might seem easier to build initially, but it quickly becomes difficult to manage as the app grows. Debugging, testing, and adding new features can become a nightmare.

Decision: We stuck to clean coding practices, which allowed the team to collaborate more effectively and made future updates much simpler.

3. Version Control – Git
We used **Git** for version control, with **GitHub** serving as our remote repository for collaboration. Git's distributed nature, powerful branching, and merging capabilities made it an obvious choice.
Decision: Git offered a reliable way to manage our codebase, collaborate in teams, and track changes efficiently. It also allowed for seamless rollbacks and issue resolution during development.

4. Database Design Standards – Firestore
We chose **Cloud Firestore**, a NoSQL database from Firebase, to manage our data. Its real-time syncing, flexible structure, and scalability were a perfect match for a dynamic application like a job portal.
Alternatives like MySQL/PostgreSQL were considered, but their rigid schemas and lack of real-time support made them less ideal for our use case. While SQL databases excel in structured environments, we needed something more adaptable.
Decision: Firestore's native support for real-time updates and tight integration with Firebase made it the best fit for storing job posts, user profiles, and application data.

5. Security Standards – OWASP Mobile Guidelines
Security was a top priority, so we followed the OWASP Mobile Security Testing Guide (MSTG)**.** This provided a comprehensive checklist for addressing mobile-specific threats, including secure authentication, encrypted data storage, and safe API communication.
Why not build our own security model?
While custom security solutions offer more control, they're risky and time-consuming. Without deep expertise, it's easy to introduce vulnerabilities.
Decision: OWASP provided a trusted framework that ensured we built the app on solid security principles, protecting both our users and their data.

6. UI Standards – Material Design
We designed the UI using **Material Design**, Google's standard for consistent and user-friendly interfaces across platforms.
Alternatives like Flat Design or **iOS Human Interface Guidelines** were considered. Flat design offers a clean look but can sometimes sacrifice usability. iOS guidelines are tailored for Apple's ecosystem, but our app needed to run smoothly on both Android and iOS.
Decision: Material Design struck the perfect balance between aesthetics, usability, and cross-platform compatibility.

7. API Design – RESTful Standards
Our application communicates with backend services using **RESTful APIs**. REST is lightweight, easy to understand, and widely supported—making it ideal for mobile development.
Other options like GraphQL offer greater query flexibility, and SOAP provides robust enterprise-level capabilities. However, REST's simplicity and lower learning curve made it the right choice for our needs.
Decision: REST was chosen for its reliability, scalability, and ease of integration with both our mobile front-end and third-party services.

8. Testing Standards – Automated Unit & Integration Testing
To ensure our code works as intended and remains reliable over time, we implemented **automated testing**—including both unit tests and integration tests.
Why not rely on manual testing?
Manual testing is still important, especially for UI and exploratory testing, but it's slow and hard to scale. Automated tests catch bugs early and allow us to make changes with confidence.
Decision: We used Flutter's testing tools to validate individual components and their interactions, which helped us maintain a stable and robust app during continuous development.

5.1.2 Hardware Standards
Hardware standards are essential to ensure that a mobile application runs smoothly, efficiently, and reliably across different devices. During the development of the **Job Portal Application**, we took several

hardware considerations into account to deliver consistent performance, regardless of device type or technical specification. Below are the key hardware standards that shaped our development strategy:

1. Device Compatibility (iOS & Android)

To reach the widest possible user base, we designed the application to work seamlessly on both **Android** and **iOS** platforms. Ensuring cross-platform compatibility means users can access the app regardless of the brand or model of their device.

Why not develop device-specific versions?

While optimizing separately for flagship and budget devices can improve performance on each, it significantly increases development time and maintenance overhead.

Decision:

Using **Flutter**, we were able to write a single codebase that adapts well to a wide range of screen sizes, resolutions, and hardware capabilities. This approach helped us maintain performance and visual consistency across both high-end and low-end devices.

2. Mobile Device Performance

Not all users own the latest smartphones, so we prioritized performance optimization, especially for older and budget-friendly devices.

Minimum Supported Specifications:

Android:

OS: Android 5.0 (Lollipop) or higher
RAM: 2 GB
Storage: 100 MB of free space

iOS:

OS: iOS 12.0 or higher
RAM: 1 GB
Storage: 100 MB of free space

Decision:

We tested the app on devices with these minimum specs and ensured that animations, screen transitions, and background services functioned without lag. Efficient use of system resources helped maintain responsiveness across all supported devices.

3. Battery Consumption and Optimization

A key focus was reducing battery drain. High battery usage often leads users to uninstall apps, so we made it a priority to run processes efficiently.

Why not prioritize performance at all times?

While maximizing performance can benefit power users, it usually comes at the cost of excessive battery usage, which negatively affects the broader user base.

Decision:

We minimized background operations, optimized data sync intervals, and reduced unnecessary wake-ups of the device. Push notifications, for example, were designed to be lightweight and meaningful, avoiding constant polling that could drain power.

4. Network Requirements and Performance (3G, 4G, Wi-Fi)

We knew our users would access the app from regions with varying network conditions. Whether they're on 3G, 4G LTE, or Wi-Fi, the app needed to respond smoothly.

Why not optimize only for faster networks?

Focusing only on high-speed connections could alienate users in rural or low-bandwidth areas, leading to a poor experience and user drop-off.

Decision:

We implemented **data compression**, reduced the size of network calls, and used **lazy loading** to defer unnecessary data fetching. This helped us deliver a responsive experience even under slower network conditions like 3G.

5. Sensor and Hardware Access

Certain features—like **location-based job search** and **camera-based profile photo uploads**—

required access to device hardware.

Why not skip sensor integration?

Skipping sensors simplifies development but limits user functionality and personalization, which are core to the user experience in our app.

Decision:

We integrated **GPS** to enhance the job search experience and allowed users to upload photos directly using their device camera. We ensured that sensor access was efficient and requested permissions only when necessary, maintaining both performance and privacy.

6. Storage Requirements (Cloud-Based and Local)

We adopted a hybrid approach to data storage, balancing **local storage** for speed with **cloud storage** for accessibility and backup.

Why not use only local storage?

Storing all data locally might offer fast access, but it risks data loss if the device is damaged or reset and limits access from multiple devices.

Decision:

Temporary data such as recently viewed jobs and settings are stored locally to ensure fast access. For persistent data like user profiles and job applications, we relied on **Firebase Cloud Firestore**. This setup allows users to resume their session from any device while keeping local storage usage minimal.

5.1.3 Communication Standards

In any application involving multiple system components, having clear communication protocols is essential. For the Job Portal Application, we defined a set of communication standards to ensure smooth data exchange between the mobile client, backend server, and external services like payment gateways and third-party APIs. Below is a breakdown of the approaches and decisions made:

1.Communication Between App and Backend

Why it matters:

We needed a reliable way for the mobile app to talk to the backend services. After considering our options, we decided on using Firebase along with RESTful APIs. REST, being stateless and lightweight, uses standard HTTP methods (GET, POST, PUT, DELETE), making it a solid fit for mobile environments.

Decision:

We went with REST because it's simple, scalable, and well-supported across platforms. It also plays nicely with third-party integrations—something we needed for handling payments and job listings.

2. Data Serialization Format (JSON)

Why it matters:

To exchange data between systems, we needed a standard format. JSON was the natural choice—it's lightweight, easy to read, and works well with mobile apps.

What we considered:

We looked into XML, which is great for complex data but is more verbose and slower to parse.

Decision:

JSON won out for its speed, simplicity, and wide support, especially in mobile development.

3. Real-Time Communication (WebSockets)

Why it matters:

The app includes a chat feature for recruiters and job seekers. For real-time conversations, we needed a fast, always-on connection.

What we considered:

Polling was one option—it's easy to implement but inefficient because it sends repeated requests.

Decision:

WebSockets were the better fit. They provide a continuous, two-way connection, which keeps latency

low and efficiency high—ideal for real-time messaging.

4. Secure Payment Communication (HTTPS)

Why it matters:
When handling payments, security is non-negotiable. All communication with payment gateways had to be encrypted.
What we considered:
HTTP was faster but not secure—too risky for sensitive data.
Decision:
We went with HTTPS for its encryption and security, protecting user data during transactions.

5. Notifications and Messaging (FCM)

Why it matters:
Users need to stay informed, even when the app isn't open. Push notifications help with that.
What we considered:
APNs is great for iOS, but we needed something that works cross-platform.
Decision:
Firebase Cloud Messaging (FCM) was the clear choice. It supports both iOS and Android, integrates easily, and handles background notifications seamlessly.

6. Authentication and Authorization

Why it matters:
User login and security were a priority. We needed a reliable, secure method to manage authentication.
What we considered:
Basic Authentication was on the table, but it’s not secure enough on its own.
Decision:
We implemented Firebase Auth with OAuth 2.0. This token-based system provides secure, scalable authentication while protecting user data.

7. Error Handling and Logging

Why it matters:
It's important to track and respond to issues as they happen. Standard error codes and crash reporting help keep things running smoothly.
What we considered:
Custom error codes offered more control, but weren’t as compatible with tracking tools.
Decision:
We stuck with standard HTTP status codes and integrated Firebase Crashlytics to track crashes and exceptions in real time. This helps the team spot and fix bugs quickly.

8. Cross-Platform Communication

Why it matters:
To save time and resources, we needed a framework that allowed us to build for both Android and iOS from a single codebase.
What we considered:
Going fully native would give us platform-optimized apps but would double our work and development costs.
Decision:
Flutter was the best fit. It lets us build once and deploy across both platforms, with native-like performance and faster development cycles.

5.2 Impact on Society, Environment and Sustainability

1. Impact on Society:

The Job Portal Application can benefit society by boosting employment opportunities and connecting job seekers with potential employers. It provides a platform for effective engagement between recruiters and job searchers, hence increasing employment rates and job matching efficiency. By utilizing location-based services, the platform assures that job opportunities are available to people in various places, encouraging economic development and lowering unemployment.

Furthermore, the coin-based application system motivates individuals to invest in their job hunt, which promotes active engagement. The app also promotes diversity and inclusivity by allowing people of various backgrounds and skill levels to access job postings and professional prospects.

2. Impact on the Environment:

The Job Portal Application encourages digital involvement, eliminating the need for physical resources like paper and ink for job applications, hence helping to paper waste reduction. Because the platform functions mostly online, it lowers the need for commuting, which may have a little but positive impact on carbon emissions by encouraging remote job options.

3. Impact on Sustainability:

From a sustainability standpoint, the project promotes the digitalization of job-related services, allowing people to access jobs and services from the convenience of their own homes. It eliminates the need for typical office sets and in-person meetings, saving overhead expenses and reducing environmental impact. The cloud-based architecture ensures scalability, allowing the program to serve a rising user base without using considerable resources, supporting long-term app sustainability.

Finally, the Job Portal Application aspires to have a good impact on society by increasing employment possibilities, lowering environmental footprint, and contributing to digital sustainability through its efficient and scalable design.

5.2.1 Impact on Life

The Job Portal Application immediately affects users' life by streamlining the job search and recruitment process. It provides job searchers with an accessible and user-friendly platform to find employment that matches their location, income expectations, and work schedule choices. This eliminates stress, saves time, and broadens job options. It offers recruiters a streamlined way for connecting with qualified candidates, hence increasing hiring efficiency. The addition of real-time chat, location filters, and profile recommendations enhances the whole experience, contributing to a higher quality of life by encouraging economic independence, job satisfaction, and work-life balance.

5.2.2 Impact on Society & Environment

Societal Impact:

• The application connects employers and job seekers, particularly in underserved or rural areas, promoting local employment and community development.
• Digital inclusion removes traditional barriers to job-seeking and promotes equal opportunities.
• The coin-based approach provides an incentive mechanism for involvement and responsibility.

Environmental Impact:

• Encourages paperless job applications to reduce the environmental impact of traditional job seeking methods.
• Reduces carbon emissions by avoiding unnecessary travel for interviews and inquiry.
• Encourages remote work and contributes to a low-footprint job economy.

5.2.3 Ethical Aspects

This initiative is committed to ethical technology principles. User data privacy and security were prioritized, guaranteeing that all personal and professional information is safely maintained and never shared without permission. The platform is intended to provide fair and equitable access to all users, regardless of background, while encouraging a non-discriminatory atmosphere. Transparency is also a major focus—users are given explicit information regarding coin usage, the reliability of job listings, and the identities of recruiters to assist prevent fraud. Furthermore, all communications, notifications, and data handling are governed by clearly established privacy rules and terms of service.

By implementing these standards, the platform develops trust, promotes responsible usage, and maintains its integrity as a safe and inclusive area for both job seekers and recruiters..

5.2.4 Sustainability Plan

To support the long-term success of the project, several sustainability practices have been thoughtfully built into its foundation:

1. **Technical Sustainability**
    - The app is developed using Flutter, which supports both Android and iOS platforms from a single codebase. This not only saves development time but also reduces the resources needed for maintenance.
    - By leveraging cloud services like Firebase, the application can scale as the user base grows—without the need for heavy infrastructure investments.
2. **Economic Sustainability**
    - A built-in coin-based system offers a simple but effective way to monetize the platform, helping cover maintenance and future development costs.
    - There's also room to introduce optional ads or premium features down the line, which can further support the platform financially.
3. **Social Sustainability**
    - The platform aims to connect people from all walks of life with job opportunities, helping to promote inclusion and economic participation.
    - Regular updates and a strong focus on user feedback will help keep the platform useful, relevant, and easy to access for a broad audience.
4. **Environmental Sustainability**
    - By encouraging digital interaction and remote job search capabilities, the app helps reduce the environmental footprint often associated with traditional recruitment methods.
    - It supports a more eco-friendly approach to employment by minimizing the need for physical resources or in-person processes.

5.3 Project Management and Financial Analysis

1. Project Management Overview

Effective project management was critical to the application's timely and cost-effective development. The team used the Agile technique, which divided the development process into incremental sprints. This strategy enabled more flexibility, regular progress reviews, and ongoing improvement based on user feedback and testing. By responding fast to changes and polishing features with each sprint, the project maintained pace and remained focused on user needs throughout the development cycle. The key stages of project management were:Requirement gathering & planning

- Design & prototyping
- Development (Frontend, Backend, APIs)
- Testing & deployment
- Marketing & maintenance

2. Cost Analysis

A. Standard Budget Estimate:

Table 5.1:Cost Analysis

Category	Cost (USDT)	Details
Developer Salaries	12,0000	3 developers × 1,0000 tk /month × 4 months
UI/UX Design	1,5000	Freelance designer or internal resource
Server & Hosting (1st year)	8000	Firebase or AWS/Heroku cloud services
Domain & SSL Certificate	1000	Annual domain cost + HTTPS security
Tools & Licenses	3000	IDEs, testing tools, API licenses
Marketing & Promotion	1,5000	Social media ads, content marketing, branding
Miscellaneous	3000	Unforeseen expenses
Total Estimated Budget	**16,5000**	

3. Revenue Model

To ensure the long-term financial sustainability of the app, the following revenue streams are proposed:

A. Coin-based Application Fees

- Job seekers must use coins to apply for jobs.
- Coins can be purchased through the app for various application
- Example: $2 for 100 coins, $5 for 300 coins, $10 for 750 coins.

B. Premium Features for Recruiters

- Recruiters can pay to post **featured jobs**.
- Subscription-based model for accessing advanced filtering, unlimited candidate views, or analytics.

C. In-App Advertising

- Integrate **non-intrusive ads** using Google AdMob.
- Recruiters or third parties can advertise services related to employment

5.4 Complex Engineering Problem

Creating a modern, feature-rich Job Portal Application is a challenging engineering endeavor. The task extends far beyond developing code to integrating real-world systems, fulfilling high performance criteria, and understanding the nuanced needs of many users. The complexities emerge not only from

technological implementation, but also from domain-specific issues such as recruitment, real-time communication, and digital monetization.

Key Areas of Complexity

I. Multi-role Architecture
The application supports three distinct user types: Admin, Recruiter, and Job Seeker. Each role comes with its own set of permissions, interfaces, and responsibilities. Designing a system that securely manages authentication, access control, and data flow between these roles demands careful architectural planning and robust backend logic.

II. Real-time Communication
To facilitate instant communication between recruiters and job seekers, a real-time chat feature is implemented. Whether built using socket programming or services like Firebase Firestore, this system must ensure quick message delivery, seamless media sharing (like resumes or documents), and reliable performance under varying network conditions.

III. Location-based Filtering and Search
Allowing users to search for jobs based on location adds another layer of complexity. This involves using geolocation services, integrating mapping APIs (like Google Maps), and ensuring search queries are fast and accurate—even in areas with a dense job or user base. Performance optimization is critical to prevent lag and improve user experience.

IV. Coin-based Transaction System
A unique aspect of the app is its coin-based system, where job seekers use virtual coins to apply for jobs. Implementing this requires building a secure and transparent micro-transaction framework. The system must handle coin purchases, track balances, log usage history, and include fraud prevention mechanisms.

V. Recommendation Engine
To enhance usability and relevance, the application includes a recommendation system. Whether based on machine learning or rule-based logic, it analyzes user behavior, preferences, and past activity to suggest jobs to seekers or candidates to recruiters. This improves engagement and helps match the right people with the right opportunities.

VI. Scalability and Performance
The system must be scalable to support thousands of concurrent users, including real-time actions like job posting, chatting, and coin purchasing. This requires efficient backend design, database indexing, and optimized APIs.

VII. Security and Data Privacy
Managing sensitive personal and financial data mandates compliance with best practices in encryption, secure APIs, and data storage. This is particularly critical for GDPR or similar data protection standards.

VIII. Cross-platform Compatibility
Since the app is built using Flutter, ensuring smooth performance on both Android and iOS, with consistent UI/UX and access to native device features (camera, GPS, notifications), introduces additional engineering challenges.

5.4.1 Complex Problem Solving

The Job Portal Application's development entails a variety of complex engineering problems that correspond to the categories outlined in Table 5.2. The following is the mapping and justification for each category:

Mapping with Problem Solving Categories

Table 5.2: Mapping With problem solve

Problem-Solving Category	Mapping Code	Mapped
Depth of Knowledge	P1	✓
Range of Conflicting Requirements	P2	✓
Depth of Analysis	P3	✓
Familiarity	P4	✓
Extent of Applicable Codes	P5	✓

P1 – Depth of Knowledge

Rationale:

The project requires the integration of diverse technologies such as:

- Flutter for cross-platform development
- Firebase for real-time database and authentication
- Geo-location services for map-based search
- Payment gateway for coin purchases
- Role-based access control (Admin, Recruiter, Job Seeker)

These require in-depth knowledge of **mobile development, cloud systems, databases, and secure architecture**. Hence, the solution fits under **P1**, where a broad **knowledge profile** is required.

Knowledge Profile Mapping for P1:

Table 5.2**:** knowledge profile mapping

Knowledge Profile Level	**Description**	**Mapped**
K1	Engineering fundamentals	✔
K2	Disciplinary knowledge	✔
K3	Cross-disciplinary knowledge	✔
K4	Research-based knowledge	✔
K5	Practice-based learning and emerging tech	✔

Rationale:

1. **K1 & K2**: Core programming skills in Dart, object-oriented design, data structures.
2. **K3**: Integration of multiple systems (UI/UX, backend services, chat, maps).
3. **K4**: Includes evaluation of existing systems, gap analysis, and innovative solutions like coin-based monetization.

4. **K5**: Using modern technologies (Flutter, Firebase, APIs, real-time database), practicing DevOps and deployment models.

P2 – Range of Conflicting Requirements

Rationale:

The project had to manage:

- Recruiter vs. Job Seeker vs. Admin interfaces with different goals.
- Monetization (coins) without reducing user engagement.
- Real-time communication without overloading backend resources.
- Performance vs. cost (using Firebase Free Tier where possible).

P3 – Depth of Analysis

Rationale:

Complex system analysis was required for:

- Role-based workflows
- Chat logic and database structure
- CV update system
- Security and coin logic integration

Modeling the use cases, creating ER diagrams, and designing scalable architecture were all part of a **deep analytical process**.

P4 – Familiarity

Rationale:

The problem was **partially familiar**. While job portals exist, integrating real-time chat, coin systems, and recruiter recommendations into a mobile app posed unique implementation challenges that required **creative adaptation** of known models.

P5 – Extent of Applicable Codes

Rationale:

While not regulated by formal engineering codes like electrical or civil engineering, the project adheres to:

- **Software Engineering principles**
- **Data privacy regulations (GDPR-like)**
- **Secure coding practices**
- **Mobile UI/UX standards (Material Design)**

5.4.2 Engineering Activities

The creation and execution of the Flutter-based Job Portal Application required a number of hard engineering tasks. These activities can be matched to the engineering activity descriptions listed in Table 5.3. Each map emphasizes the amount of resource management, interaction, innovation, and overall impact on society and the environment.

Engineering Activity Mapping Table

Table 5.4**:** Engineering activity mapping

Engineering Activity	Code	Mapped
Range of Resources	EA1	✔
Level of Interaction	EA2	✔
Innovation	EA3	✔
Consequences for Society & Environment	EA4	✔
Familiarity	EA5	✔

EA1 – Range of Resources

Rationale:

The project utilized a wide range of technical and non-technical resources:

- **Technological resources**: Flutter SDK, Firebase, Cloud Firestore, Google Maps API, Payment Gateway, GitHub for version control.
- **Human resources**: Developers, UI/UX designers, testers, and project supervisors.
- **Information resources**: Existing job platforms, documentation, and research literature.

Efficient resource management was required to integrate these tools under time and budget constraints, while delivering a smooth and scalable application.

EA2 – Level of Interaction

The app required high levels of interaction:

- Between **different user roles** (Admin, Recruiter, Job Seeker)
- With **external services** such as maps, payment processors, and authentication providers
- **User-System Interaction** through well-designed UI/UX following Material Design standards

Furthermore, the in-app **chat system** and **notification services** involved real-time, interactive functionality, contributing to a high interaction level.

EA3 – Innovation

Several aspects of the project involved innovative solutions:

- **Coin-based job application model**, which gamifies job applications while monetizing the platform
- Recommendation system for recruiters to view top candidates based on history and job match
- Location, salary, and timing-based filtering—making job discovery more intelligent and user-centered

This level of innovation goes beyond standard job portals, targeting both efficiency and user satisfaction.

EA4 – Consequences for Society and Environment

The platform has meaningful implications for society and employment**:**

- Improves employment access by connecting job seekers and recruiters through a mobile-friendly platform
- Encourages digital literacy and remote hiring
- The system reduces physical paperwork and commuting needs, contributing marginally to environmental sustainability

The solution fosters a socially beneficial model aligned with ethical and sustainable goals.

EA5 – Familiarity

Rationale:

The application combines **familiar concepts** (like job boards and messaging) with **new implementations** such as coin-based systems and cross-platform real-time chat. While the general problem is familiar, the integration and innovative features present **new and context-specific challenges** that required learning and adaptation.

5.5 Summary

The Job Portal Application was developed in accordance with known software engineering standards to assure quality, security, and performance across many user roles (Admin, Recruiter, Job Seeker). Observed standards include IEEE 830 for Software Requirements Specification (SRS), ISO/IEC 25010 for software quality attributes such as usability, maintainability, and performance, OWASP guidelines for data security and vulnerability prevention, and Google Material Design for consistent and intuitive user experience across devices. These guidelines paved the way for dependable development processes and user-centered design. Compliance enabled a well-documented, testable, and scalable application.

Design Challenges Faced:

1. Multi-role functionality:
Creating specialized features and interfaces for various user roles while retaining a cohesive system necessitated a complicated access-control framework and effective state management.
2. Real-time chat integration:
Implementing safe, scalable real-time messaging with Firebase presented performance and data synchronization problems, especially under changing network conditions.
3. Coin-based economic system:
Developing a digital currency system for job applications necessitated logic for transactions, purchases, refunds, and fraud prevention—all while preserving user trust and transparency.
4. Location-Based Filtering:
Accurate and effective job filtering based on pay, location, and working hours necessitated interaction with geolocation services and query optimization.
5. Enhanced cross-platform performance:
Flutter's smooth performance on both Android and iOS required significant testing and UI changes for multiple screen sizes and OS-specific behaviors.

Despite these challenges, the project team successfully designed a robust and user-friendly application by applying sound engineering principles and iterating through multiple prototypes

Chapter 6

Conclusion

6.1 Summary

This project focuses on designing and developing a Flutter-based Job Portal Application that connects job seekers and recruiters using a modern, mobile-first platform. The software has three distinct user roles: Admin, Recruiter, and Job Seeker, each with its own set of tools and functionalities for streamlining the recruitment and job-hunting process.

Some of the primary features include a location-based job search filter, a unique coin-based system for applying to jobs, real-time chat between recruiters and candidates, and user-friendly dashboards for managing job postings and applications. Recruiters can post job openings, check applicant profiles, and contact directly with potential applicants. Job searchers can browse job postings, apply using coins, follow recruiter reactions, and manage their digital CVs. Administrators are given controls to administer services, monitor transactions, and oversee platform activities.
The development method involved rigorous requirement analysis, user-centered design, and adherence to known software engineering standards such as IEEE 830. It also stressed secure coding methods and included complicated features including a role-based system architecture, cross-platform UI/UX, and connectivity with third-party services like Firebase, Google Maps, and payment gateways.

Overall, this program provides a scalable and effective answer for modern recruitment requirements. By encouraging digital employment, enhancing transparency, and making job prospects more accessible

6.2 Limitation

While the Job Portal Application successfully delivers a wide range of useful features for job seekers, recruiters, and administrators, there are some limitations that were identified during development and testing:

1. **Limited Recommendation Intelligence**
   The current recommendation system is rule-based and lacks advanced machine learning capabilities. As a result, suggested job seekers or job listings may not always be optimally matched based on user behavior or preferences.
2. **Scalability Constraints**
   The use of Firebase on a free-tier plan restricts the number of users, storage, and real-time

database reads/writes. This may lead to performance issues when the user base grows significantly without migrating to a premium plan.

3. **Basic Coin Economy**
   The coin system is functional but lacks advanced financial controls like refunds, discounts, or wallet history logs, which could enhance transparency and user engagement.
4. **Limited Admin Analytics**
   The Admin Panel currently provides basic service management features but lacks in-depth analytics, reporting tools, or visualization dashboards for monitoring platform performance.
5. **Chat Functionality Scope**
   The real-time chat system does not support multimedia messages (images, documents) and is limited to text-based interactions.
6. **Dependence on Internet Connectivity**
   Since the app relies heavily on real-time services (Firebase, Maps API, etc.), it requires a stable internet connection to function properly, which may limit usability in rural or low-connectivity areas.
7. **Cross-Platform UI Inconsistencies**
   Although Flutter offers cross-platform capabilities, minor inconsistencies in UI behavior or performance between Android and iOS may still occur.

6.3 Future Work

Although the Job Portal Application's present version offers a solid basis for job searching and recruitment, there are a few areas that could be enhanced or added in later versions. The goals of these improvements are to increase user experience, intelligence, and scalability.

1. Recommendation System Driven by AI:
Smarter recommendations and more user satisfaction may result from using machine learning algorithms to match job searchers with job postings based on their profile, activity, abilities, and previous interactions.
2. The Coin Wallet System with Advanced Features:
A fully functional digital wallet system with transaction history, coin return options, promotional offers, and interaction with additional payment gateways for a broader reach could be included in future iterations.
3. Skill analysis and resume parsing:
To make the hiring process quicker and more data-driven, include resume parsing software that automatically extracts experience, education, and skills from uploaded CVs.
4. Integration of Voice and Video Calls:
The interview process would be enhanced and the platform would become more dynamic and professional if recruiters and job seekers could communicate through voice or video calls within the app.
5. Verification System for Job Posts
To guard against fraud and guarantee the legitimacy of recruiters, include a function that allows job postings to be validated by an administrator or third party.
6. Dashboard for Admin Analytics
More control and insight into platform performance would be possible with an enhanced analytics panel for the administrator that includes charts and information on job activity, user engagement, currency usage, and growth trends.
7. Offline Mode with Restricted Functionalities
Accessibility in places with poor connectivity might be improved by adding an offline option that allows users to view previously loaded data and create job applications without live internet.
8. Multilingual Support Adding support for multiple languages will make the program more inclusive and help it reach a wider user base, particularly in varied places.

References

[[1] Google, “Flutter Documentation,” [Online]. Available: https://flutter.dev/docs.

[2] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall.

[3] Ostad, “Ostad Online Learning Platform,” https://ostad.app/dashboard/my-courses/64f48092690da38bc2fe78e5. [19]

[4] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner’s Approach*, 8th ed. New York, NY: McGraw-Hill, 2014.

[5] W. Zhang and X. Zhang, “A Study on Online Job Portal System and Applications,” *Int. J. Comput. Appl.*, vol. 162, no. 6, pp.

[6] Firebase, “Firebase for Flutter,” [Online]. Available: https://firebase.flutter.dev.

[7]Bkash, “Integrate Payments in Flutter,” [Online]. Available: https://bkash.com/docs/payments.

[8] M. Reso Coder, “Flutter State Management with Riverpod,” YouTube Video, Jul. 15, 2024. https://www.flutter.dev/flutter/apps.

[9] Pub.dev, “google_nav_bar,” package v5.0.6, 2024. [Online]. Available: https://pub.dev/packages/google_nav_bar. [Accessed: Apr. 20, 2025].

[10] World Bank, “Digital Jobs for Youth: A Manual for Job Creation,” [Online]. Available: https://www.worldbank.org. [Accessed: May 1, 2025].

[11] GitHub, “Open Source Job Portal Projects,” [Online]. https://github.com. [May 1, 2025].

[12] S. Talasila, S. Jayaraj, and K. Dey, “Building a Coin-Based Job Application Platform Using Flutter and Firebase,” *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 6, pp. 2349–2355, Jun. 2020.

[13] *Systems and software engineering – Software life cycle processes*, Int. Organization for Standardization, Geneva, Switzerland, 2017.

[14] Dart Team, “Dart language tour,” Dart.dev, 2025. [Online].https://dart.dev/guides/language/language-tour.

[15] OpenAI, “ChatGPT Technical Report,” [Online]. Available: https://openai.com/research.

[16] Flutter Team, “Flutter documentation,” Flutter.dev, 2025. [Online]. Available: https://flutter.dev/docs.

[17] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA: Pearson Education

[18] YouTube, “Job Portal App Development in Flutter,” https://www.youtube.com/watch?v=FP737UMx7ss.

[19]] S. McConnell, *Code Complete*, 2nd ed. Redmond, WA: Microsoft Press, 2004.

201-15-3741

ORIGINALITY REPORT

12%	8%	2%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	5%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
3	Submitted to United International University Student Paper	<1%
4	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
5	Submitted to University of Southampton Student Paper	<1%
6	Submitted to Siksha 'O' Anusandhan University Student Paper	<1%
7	families-share.eu Internet Source	<1%
8	Elena V. Tikhonova, Galina I. Efremova, Marina A. Kosycheva. "Computer Simulation as a Tool of Psychological Readiness for Employment Process for Migrant Students", 2021 5th International Conference on Education and Multimedia Technology (ICEMT), 2021 Publication	<1%
9	dit.jnec.edu.bt Internet Source	<1%