

# TEAM UP – AN ANDROID APPLICATION FOR JOB AND TEAM FINDING

Senior Project



Primary Advisor: **Samia Asloob Qureshi**  
Secondary Advisor: **Ali Faheem**

Presented by:

21-11102  
21-10730  
21-10365

Ahad Tariq  
Muhammad Bilal  
Razi Ul Hassan

Department of Computer Science

**Forman Christian College (A Chartered University)**

# TEAM UP – AN ANDROID APPLICATION FOR JOB AND TEAM FINDING

By

**Ahad Tariq , Muhammad Bilal , and Razi Ul Hasan**

Project submitted to

Department of Computer Science,

Forman Christian College (A Chartered University),

Lahore, Pakistan.

in partial fulfillment of the requirements for the degree of

**BACHELOR OF SCIENCE  
IN  
COMPUTER SCIENCE (Honors)**

Samia Asloob Qureshi

---

Primary Project Advisor

Ali Faheem

---

Secondary Project Advisor

---

Senior Project Management  
Committee Representative

## Abstract

In the modern day , there is a large gap between academia and industry. Many fresh graduates from university are unable to secure a full-time job for themselves in any industrial domain. This is because after graduating , individuals lack a technical and mental skill set to sustain themselves in an industry. Due to this problem, they are not equipped to handle the responsibilities associated with a job role. The product we have built is an Android Application which gives fresh graduates and unemployed individuals to create teams for project building which would enhance their skill set and therefore enhance their resume. This enhancement of skill set, and confidence would make them more equipped to handle the responsibilities associated with a job role in an industry.

The product we are building is not only solving this problem. Entrepreneurs in the modern day today also face trouble in finding and recruiting individuals for their project ideas. This application would give them a platform to post their project ideas and recruit individuals for their teams.

Lastly , employers and HR associates of companies today receive a lot of resumes daily for their job openings. This results in them being overloaded with resumes and they have no option to go through the thousands of resumes they receive for a particular job. Our application would be able to solve this problem as well by shortlisting only deserving candidates for a job. This feature of our application would help employers and HR associates in order to recruit quickly and not be hassled.

## **Acknowledgement**

We would like to acknowledge the Computer Science Department for giving us an opportunity to create an industrial level project in our final year. This opportunity has provided us with the time and space to learn and develop a skill set associated with modern day technologies that will help us in our future careers.

## List of Figures

Figure 1	Use-case diagram	15
Figure 2	Context diagram	18
Figure 3	Activity diagram	19
Figure 4	Class diagram	20
Figure 5	System design block diagram	21
Figure 6	DFD diagram	22
Figure 7	Sequence diagram	23
Figure 8	System Architecture Diagram	26
Figure 9	Swim Lane Diagram	27

# List of Tables

Provide a list of all tables in following format.

Table 1	Sign Up	4
Table 2	Login	5
Table 3	Create Job	5
Table 4	View Posted Jobs	6
Table 5	View Job	6
Table 6	Create Team	7
Table 7	View Posted Teams	8
Table 8	View Team	8
Table 9	Edit Job	9
Table 10	Edit Team	10
Table 11	Edit Team Role	10
Table 12	Add more Team Roles	11
Table 13	Create Quiz	12
Table 14	Take Quiz	12
Table 15	Find Jobs	13
Table 16	Find Teams	14

Table 6.1	TC-1	34
Table 6.2	TC-2	34
Table 6.3	TC-3	35
Table 6.4	TC-4	35
Table 6.5	TC-5	35
Table 6.6	TC-6	36
Table 6.7	TC-7	36
Table 6.8	TC-8	36
Table 6.9	TC-9	37
Table 6.10	TC-10	37
Table 6.11	TC-11	38
Table 6.12	TC-12	38
Table 6.13	TC-13	39
Table 6.14	TC-14	39
Table 6.15	TC-15	40
Table 6.16	TC-16	40
Table 6.2	Summary of Test Results	41



# TABLE OF CONTENTS

<b>ABSTRACT.....</b>	<b>I</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>II</b>
<b>LIST OF FIGURES.....</b>	<b>III</b>
<b>LIST OF TABLES.....</b>	<b>IV</b>
<b>CHAPTER 1.INTRODUCTION.....</b>	<b>1</b>
1.1INTRODUCTION.....	1
1.2OBJECTIVES.....	2
1.3PROBLEM STATEMENT.....	2
1.4SCOPE.....	2
<b>CHAPTER 2.REQUIREMENTS ANALYSIS.....</b>	<b>2</b>
2.1LITERATURE REVIEW.....	3
2.2USER CLASSES AND CHARACTERISTICS.....	3
2.3DESIGN AND IMPLEMENTATION CONSTRAINTS.....	3
2.4ASSUMPTIONS AND DEPENDENCIES.....	4
2.5FUNCTIONAL REQUIREMENTS.....	4
2.6USE CASE DIAGRAM.....	15
2.7NONFUNCTIONAL REQUIREMENTS.....	16
2.8OTHER REQUIREMENTS.....	17
<b>CHAPTER 3.SYSTEM DESIGN.....</b>	<b>5</b>
3.1APPLICATION AND DATA ARCHITECTURE.....	18
3.2COMPONENT INTERACTIONS AND COLLABORATIONS.....	22
3.3SYSTEM ARCHITECTURE.....	26
3.4ARCHITECTURE EVALUATION.....	27
3.6SCREENSHOTS/PROTOTYPE.....	27
<b>CHAPTER 4.TEST SPECIFICATION AND RESULTS.....</b>	<b>7</b>
4.1TEST CASE SPECIFICATION.....	34
4.2SUMMARY OF TEST RESULTS.....	40
<b>CHAPTER 5.CONCLUSION AND FUTURE WORK.....</b>	<b>8</b>
5.1PROJECT SUMMARY.....	41
5.2PROBLEMS FACED AND LESSONS LEARNED.....	41
<b>REFERENCES.....</b>	<b>42</b>
<b>APPENDIX A GLOSSARY.....</b>	<b>43</b>
<b>APPENDIX B DEPLOYMENT/INSTALLATION GUIDE.....</b>	<b>44</b>
<b>APPENDIX C USER MANUAL.....</b>	<b>44</b>
<b>APPENDIX D STUDENT INFORMATION SHEET.....</b>	<b>45</b>
<b>APPENDIX E PLAGIARISM FREE CERTIFICATE.....</b>	<b>46</b>
<b>APPENDIX F PLAGIARISM REPORT.....</b>	<b>46</b>



## Revision History

Name	Date	Reason For Changes	Version

# 1. Introduction and Background

## 1.1 Introduction

In today's market individuals who are being recruited for jobs have a complete and polished skill set, experience, and confidence to fulfill the role and responsibilities that are being obligated to them along with their respective job. Many students at universities and fresh graduates from many disciplines in today's age have a lot of theoretical knowledge and creativity but lack this combination and therefore go through a struggle during the employment process. The product that we have built is a Team and Job finding Android application that would help fresh graduates in posting their project ideas and find other unemployed individuals, fresh graduates, or students whom they would be able to collaborate and form a team with in-order to polish and complete their skill set, gain experience, build confidence and enhance their resumes which would ultimately, as a result, increase their chances for a full time job. Furthermore, this product would be solving another problem which is that many entrepreneurs and freelance creative artists in the modern day have wonderful ideas but struggle to find the necessary people in no time with who they can form a team and turn that idea into a reality. This product can help them form a team and quicken the process of their project work. Last, but not the least this product would also solve an HR oriented problem which is that many HR representatives receive a large number of resumes daily, but this application would short list deserving candidates which would eliminate the need for the employer and HR representatives to go through all of the received resumes.

For this application we have used three tools:

- 1- Python with flask.
- 2- Android Studio
- 3- Firebase

With python we have designed an Artificial Intelligence based recommendation system using Natural language processing methodologies. Flask is a web application framework. It will support the back-end architecture of our application by receiving and sending API requests and responses. Android Studio is used as the coding environment platform to create this mobile application. Firebase Real Time Database is used to store the data generated from the application and Firebase Authentication Service is also used to authenticate registered users of the application. The mobile application will start with a sign-up page where the user will register himself/herself along with details on the application. The user along with finding teams and jobs will also be able to create jobs and teams through registering himself as an Employer and Team Owner. The jobs would be recommended to the users based on the user skill set through the Artificial Intelligence based recommendation system. After recommendation, in order to apply for the job, the user will have to take the Job Quiz and pass the quiz to be shortlisted. Once shortlisted an interview date for the job seeker would be set by the employer. Teams would be recommended to the users through traditional programming based on the user skill set and required experience. Users would be able to apply for the team and be a part of team projects which would enhance their resume and skill set.

## **1.2 Objectives**

The objective to be achieved after the project completion is to provide an application to fresh graduates and unemployed individuals which would be a platform for them to form teams in order to work on projects of their disciplines together to enhance their resume. Entrepreneurs would also be given a platform where they can post their projects and through which they can recruit individuals and build teams for their personal projects after looking through their profiles. Another objective which this application would be achieving is that to eliminate undeserving candidates for job selection automatically. The application would shortlist deserving candidates based on a Job Quiz posted by the employer for the particular job.

## **1.3 Problem Statement**

The product is aiming to solve a modern-day problem of fresh graduates and university students who lack a technical and mental skill set along with experience which is required to get into a job market and sustain themselves there. It is a significant problem as many students after graduating from university do not know how to build and create industrial level projects by themselves and do not have the required experience on their resumes which is mandatory to enter the job market. This application would provide opportunities to fresh graduates, university students, and unemployed individuals to form teams with one another and work on a project which would be helpful for them in enhancing their resume. On the application any fresh graduate would be able to build a team for his personal project. Other fresh graduates and students would be able to find that team on the application and apply for a particular role on the team. Through this they would be able to work on the project, gain experience, and build a skill set for themselves for any industrial job. Entrepreneurs, the same way, can create a team for their project idea on the application and recruit individuals for their respective project. They can view all the different team applicants' profiles and find the appropriate fit for their team. Another significant problem being solved in this application is that many employers and companies today receive thousands of resumes for their posted jobs. The application takes care of this problem by giving the employer a feature to post a Job Quiz for a job. The application automatically shortlists the deserving candidates for the job by selecting only the candidates that have passed the Job Quiz.

## **1.4 Scope**

From a public perspective, the scope of this product comprises of fresh graduates, students, unemployed individuals, employers, team builders, and entrepreneurs.

From an application perspective our scope consists of two modules "Find a Team" and "Find a Job". The "Find a Job" module would have an Artificial Intelligence based Recommendation Engine implemented through Natural Language processing and Cosine Similarity. The algorithm would be trained using a data set of Naukri.com obtained from Kaggle. The data set comprises of different jobs associated with their respective skill set. Our recommendation engine generates different jobs based on a user skill set. The recommendation engine produces a similarity score of the user skill set with the skill sets of all the jobs in the data set. The 20 jobs which produce the highest similarity score between the user skill set and the job requirement skill set are the recommended jobs. For this process, we have used Natural Language Processing, specifically Count Vectorizer. In order to obtain the similarity scores of the skill sets of all the jobs with the user skill set the skill set data which is in textual format needs to be converted to machine readable form. Count Vectorizer extracts features

from the textual data and creates a specific numerical feature vector for each skill set using the count of a particular skill in a skill set. It does that for each job's skill set in the data set along with the user skill set and creates a matrix of feature vectors. Cosine Similarity is then used to find the distance between these vectors which determines the similarity and similarity score. CountVectorizer also , while creating the matrix of feature vectors , applies the process of tokenization for all the skills in a skill set. Word Level Tokenization is applied in this context for a skill in a skill set. Once a job would be recommended to the candidate, he would also be bound to take a quiz related to the job to be considered by the employer. Such a quiz would be taken by the candidate in order to verify his skills to the employer. The "Find a Team" module would be based on traditional programming where we would be using the Firebase Real Time Database to generate teams which comprise of team roles matching the user skill set.

## **2. Requirements Analysis**

### **2.1 Literature Review**

There are many websites and applications which provide similar features to our application but there are also plenty of important differences.

- Linked In facilitates individuals with job listings based on their profile but it is not a platform where individuals can build and find teams for their own personal projects. Another difference is that employers who wish to post a job on Linked In are not able to post a quiz associated with their job to verify an applicant's skill set and shortlist deserving applicants.
- Rozee.pk gives a platform to users to post jobs and find jobs but not build and find teams , similar to Linked In. It also does not provide an opportunity for employers to post a job quiz.
- Indeed is identical to Linked In and Rozee.pk.
- Freelancing websites such as Fiver, Up work, and Freelancer are platforms which have similarities but many differences with our applications . On these websites people post their projects related to content writing , software development , photography , and digital marketing. But such projects are only given to one individual. Teams are not built for a project and the individual to whom the project is given is always an experienced individual with a polished skill set. This results in no chance for fresh graduates or students landing an opportunity for themselves to enhance their own skill set. These websites , also , do not facilitate individuals with job listings.

### **2.2 User Classes and Characteristics**

The application will have seven user classes candidate, employer, team owner, team, team member, job, question, quiz. The characteristics of the candidate class would be education, experience, skills, contact information, gender, name, email, and password, and username. The characteristics of the employer class would be employer name, company, company role , and number of jobs posted. The characteristics of job class will be job name, organization name, roles and responsibilities of the job, skills required for the job, experience required for the job, salary, and location city. The job class would have a composition relationship with employer. The characteristics of team owner would be team owner name and posted teams. The team class would have a composition relationship with team

owner and the team member class would have a composition relationship with team. The characteristics of the team class would be team name, project description, project discipline, and team members. The team member would have the characteristics of team role, required experience, quantity, required skills.

The remaining two classes are Question and Quiz would have a composition relationship as a quiz cannot exist without a question. Every job of the application would have a quiz so the job and quiz class would have an association relationship.

## 2.3 Design and Implementation Constraints

No implementation constraints were found.

## 2.4 Assumptions and Dependencies

In order to ensure rapid and smooth development in the process of implementation and design a lot of debugging is required. To ensure the part of debugging in the development phase is rapid, an operating system which would be ideal for development is important. For this purpose, we would be using Ubuntu core 20.04 which is an ideal operating system for development because of its mind-blowing community and resources available, providing not just rapid development but also for it being open source is a plus point. Another assumption is that the user will have an android phone and wifi connection.

## 2.5 Functional Requirements

### 2.5.1 Sign Up

<b>Identifier</b>		UC-1
<b>Purpose</b>		Signing up on the application
<b>Priority</b>		High
<b>Pre-conditions</b>		The person should have a name, email address, mobile number, and resume.
<b>Post-conditions</b>		User Dashboard opens
<b>Typical Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	User enters all the information required to register on the application.	System signs the user up on the application and displays the user dashboard.
<b>Alternate Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	User does not enter all, some, or one of the required pieces of information.	System does not sign the user up on the application and displays a message to the user to enter all of the required information.

**Table 1: UC-1**

### 2.5.2 Log In

<b>Identifier</b>		UC-2
<b>Purpose</b>		Logging on the application
<b>Priority</b>		High
<b>Pre-conditions</b>		The person should have a signed up email address and password combination.
<b>Post-conditions</b>		User Dashboard opens
<b>Typical Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	User enters an email address and password to log in the application.	System logs the user in the application and displays the user dashboard.
<b>Alternate Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	User enters an invalid email address and password combination.	System does not log the user in the application and displays a message to the user that invalid information has been entered.

Table 2: UC-2

### 2.5.3 Create Job

<b>Identifier</b>		UC-3
<b>Purpose</b>		Posting a job on the application
<b>Priority</b>		High
<b>Pre-conditions</b>		The employer posting a job must belong to a real world organization which has a job opening.
<b>Post-conditions</b>		Job details page opens
<b>Typical Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	The employer clicks on the navigation drawer icon on the employer dashboard.	System opens the navigation drawer on the screen.
2	The employer clicks on 'Create Job'	System opens the 'Create Job' page.
3	The employer enters the required job information.	System posts the job on the application and displays the job details page.
<b>Alternate Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	The employer does not enter all, some, or one of the required pieces of information.	System does not post the job on the application and displays a message to the employer to enter all of the required information.

Table 3: UC-3

#### 2.5.4 View Posted Jobs

Identifier	UC-4	
Purpose	Viewing the posted jobs.	
Priority	Medium	
Pre-conditions	The employer should have posted a job or jobs on the application.	
Post-conditions	The posted jobs are displayed.	
Typical Course of Action		
S#	Actor Action	System Response
1	The employer clicks on the navigation drawer icon on the dashboard.	System opens the navigation drawer on screen.
2	The employer clicks on ‘My Jobs’	System opens the posted jobs page.

Table 4: UC-4

#### 2.5.5 View Job

Identifier	UC-5	
Purpose	Viewing a posted job.	
Priority	Medium	
Pre-conditions	The job should have been posted.	
Post-conditions	The job details page of the selected job opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The employer clicks on the navigation drawer icon on the employer dashboard.	System opens the navigation drawer on the screen.
2	The employer clicks on ‘My Jobs’	System opens the posted jobs page.
3	The employer clicks on a specific job on the page.	System opens the job details page of that specific job.
Alternate Course of Action		
S#	Actor Action	System Response
1	The user clicks on the navigation drawer icon on the user dashboard.	System opens the navigation drawer on the screen.
2	The user clicks on ‘Find Jobs’	System displays the recommended jobs to the users.
3	The user clicks on a specific job from the recommended jobs.	System opens the job details of that recommended job.

Table 5: UC-5

#### 2.5.6 Create Team

Identifier	UC-6	
Purpose	Creating a Team on the application	
Priority	High	
Pre-conditions	The team owner should have a project for which he wants to build a team.	
Post-conditions	Team details page opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the navigation drawer icon on the team owner dashboard.	System opens the navigation drawer on the screen.
2	The team owner clicks on ‘Create Team’	System opens the ‘Create Team’ page.
3	The team owner enters the required team information.	System posts the team on the application and displays the team details page.
Alternate Course of Action		
S#	Actor Action	System Response
1	The team owner does not enter all, some, or one of the required pieces of information.	System does not post the team on the application and displays a message to the team owner to enter the required information.

Table 6: UC-6

### 2.5.7 View Posted Teams

Identifier	UC-7	
Purpose	Viewing the posted teams.	
Priority	Medium	
Pre-conditions	The team owner should have posted a team or teams on the application.	
Post-conditions	The posted teams are displayed.	
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the navigation drawer icon on the dashboard.	System opens the navigation drawer on screen.
2	The employer clicks on ‘My Teams’	System opens the ‘Posted teams’ page.

Table 7: UC-7

### 2.5.8 View Team

<b>Identifier</b>	UC-8	
<b>Purpose</b>	Viewing a posted team.	
<b>Priority</b>	Medium	



Pre-conditions		The team should have been posted.
Post-conditions		The team details page of the selected team opens.
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the navigation drawer icon on the dashboard.	System opens the navigation drawer on the screen.
2	The team owner clicks on ‘My Teams’	System opens the ‘Posted teams’ page.
3	The team owner clicks on a specific team on the page.	System opens the team details page of that specific team.
Alternate Course of Action		
S#	Actor Action	System Response
1	The user clicks on the navigation drawer icon on the user dashboard.	System opens the navigation drawer on the screen.
2	The user clicks on ‘Find Teams’	System displays the recommended teams to the users.
3	The user clicks on a specific team from the recommended teams.	System opens the team details of that recommended team.

Table 8: UC-8

### 2.5.9 Edit Job

Identifier	UC-9	
Purpose	Editing job details on the application.	
Priority	Medium	
Pre-conditions	The job should have been posted on the application.	
Post-conditions	Job page with modified details opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The employer clicks on the specific information of the job to be edited.	System opens the particular job information page.
2	The employer enters the new information and clicks on ‘Update’.	System displays a message that job details have been updated.
3		System displays the job page with edited details.
Alternate Course of Action		
S#	Actor Action	System Response
1	The employer clicks on the specific information of the job to be edited.	System opens the particular job information page.
2	The employer does not enter any new information and clicks on ‘Update’.	System displays a message that that no details have been updated.

Table 9: UC-9

### 2.5.10 Edit Team

Identifier	UC-10	
Purpose	Editing team details on the application.	
Priority	Medium	
Pre-conditions	The team should have been posted on the application.	
Post-conditions	Team page with modified details opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the team information to be edited.	System opens the team information page.
2	The team owner enters the new information and clicks on ‘Update’.	System displays a message that team details have been updated.
3		System displays the team page with edited details.
Alternate Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the team information to be edited.	System opens the team information page.
2	The team owner does not enter any new information and clicks on ‘Update’.	System displays a message that that no details have been updated.

Table 10: UC-10

### 2.5.11 Edit Team Role

Identifier	UC-11	
Purpose	Editing team role details on the application.	
Priority	Medium	
Pre-conditions	The team role to be edited should be part of the team posted on the application.	
Post-conditions	Team page with modified team role details opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the team role to be edited on the team page.	System opens the particular team role page.
2	The team owner enters the new information and clicks on ‘Finish’.	System displays the team page with edited team role details.
Alternate Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the team role to be edited on the team page.	System opens the particular team role page.
2	The team owner leaves a piece of information empty and clicks on ‘Finish’.	System displays a message to enter complete information.

Table 11: UC-11

## 2.5.12 Add More Team Roles

Identifier	UC-12	
Purpose	Add more team roles to a team posted on the application.	
Priority	Medium	
Pre-conditions	The team should have been posted on the application.	
Post-conditions	Team page with added team role opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the add button on the team page.	System opens the ‘Add Team Roles’ page.
2	The team owners enters the team role information and clicks on ‘Finish’.	System displays the team page with added team role.
Alternate Course of Action		
S#	Actor Action	System Response
1	The team owner clicks on the add button on the team page.	System opens the ‘Add Team Roles’ page.
2	The team owner leaves a piece of information empty and clicks on ‘Finish’.	System displays a message to enter complete information.

Table 12: UC-12

## 2.5.13 Create Quiz

Identifier	UC-13	
Purpose	Creating a quiz for a particular job on the application.	
Priority	High	
Pre-conditions	The job should have been posted on the application.	
Post-conditions	The employer dashboard opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The employer clicks on the ‘Create Job Quiz’ button on the job page.	System opens the ‘Create Question’ page.
2	The employer enters the question and clicks on ‘Add Question’.	System displays the Quiz Page with the added question.
3	The employer clicks on the ‘Add More Questions’ button.	System opens the ‘Create Question’ page.
4	The employer enters the question and clicks on ‘Add Question’.	System displays the Quiz Page with the added question.
5	The employer clicks on the ‘Finish Quiz’ button.	System displays the ‘Complete Quiz Creation’ page.
6	The employer enters the passing marks and	System displays the employer dashboard.

	time duration of the quiz.	
<b>Alternate Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
1	The employer clicks on the 'Create Job Quiz' button on the job page.	System opens the 'Create Question' page.
2	The employer enters the question and clicks on 'Add Question'.	System displays the Quiz Page with the added question.
3	The employer clicks on the 'Finish Quiz' button.	System displays the 'Complete Quiz Creation' page.
4	The employer enters the passing marks and time duration of the quiz.	System displays the employer dashboard.

Table 13: UC-13

#### 2.5.14 Take Quiz

Identifier	UC-14	
Purpose	Taking a quiz for a particular job on the application.	
Priority	High	
Pre-conditions	The job should have been posted on the application.	
Post-conditions	The quiz score screen opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The job seeker clicks on the ‘Take Job Quiz’ button on the job page.	System opens the Quiz page with the first question.
2	The job seeker answers the first question and moves to the next question.	System displays the page with the next question.
3	The job seeker answers the question.	System displays the quiz score screen with the score and message whether the user has passed the quiz.

Table 14: UC-14

#### 2.5.15 Find Jobs

Identifier	UC-15	
Purpose	Finding a job on the application.	
Priority	High	
Pre-conditions	The job seeker should be registered on the application.	
Post-conditions	A list of jobs opens.	
Typical Course of Action		
S#	Actor Action	System Response
1	The job seeker clicks on the ‘Find Jobs’ button on the navigation drawer	System displays the recommended jobs.

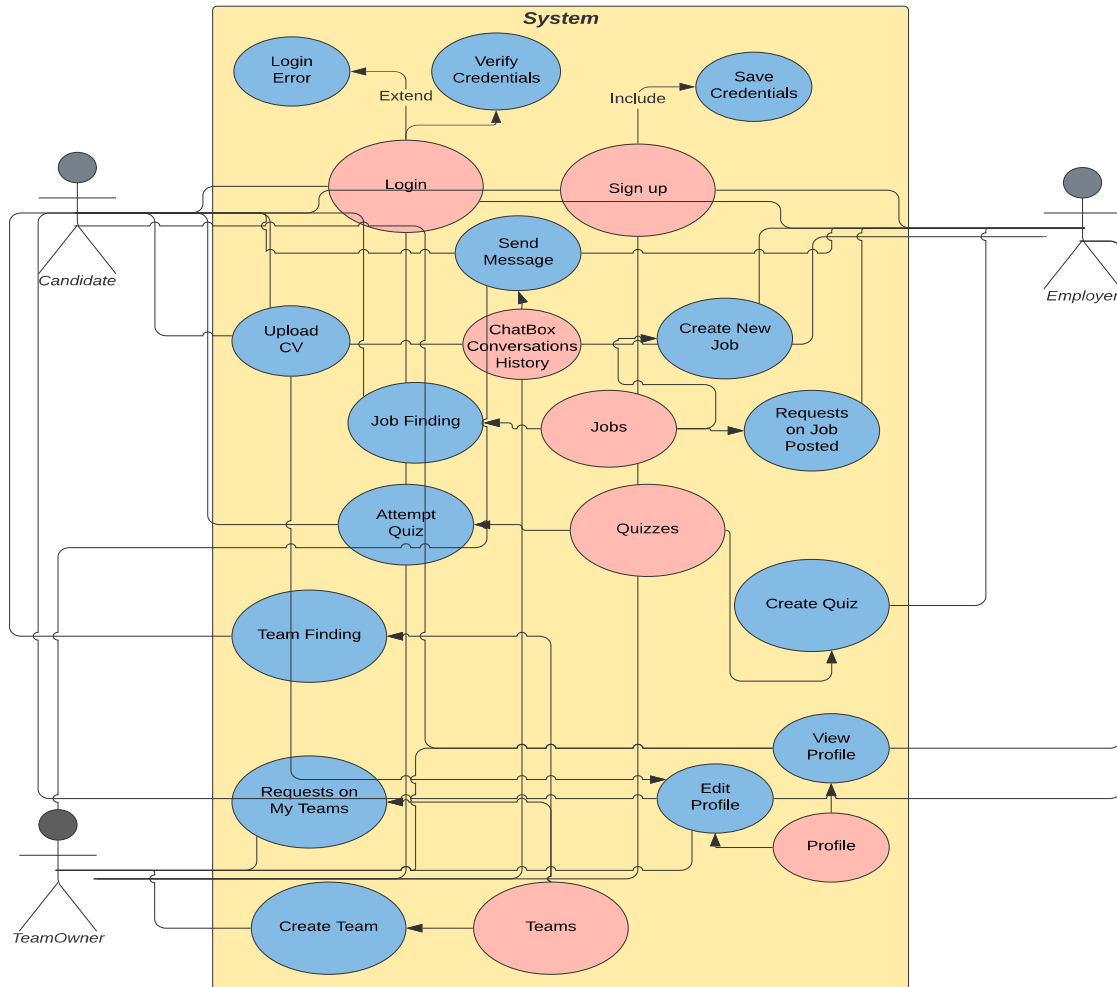
Table 15: UC-15

## 2.5.16 Find Teams

<b>Identifier</b>		UC-16
<b>Purpose</b>		Finding a team on the application.
<b>Priority</b>		High
<b>Pre-conditions</b>		The team seeker should be registered on the application.
<b>Post-conditions</b>		A list of teams opens.
<b>Typical Course of Action</b>		
<b>S#</b>	<b>Actor Action</b>	<b>System Response</b>
<b>1</b>	The team seeker clicks on the 'Find Teams' button on the navigation drawer	System displays the recommended teams.

Table 16: UC-16

## 2.6 Use Case Diagram



**Figure1: Use-case diagram**

Figure 1 provides an illustration for the use-cases of our application and how/which each use case is connected to the user classes in our application. Candidate , Team Owner , and Employer are the classes of our application. The block in the center comprises of all the features/ use-cases of our application.

## 2.7 Nonfunctional Requirements

### 2.7.1 Performance Requirements

The application is required to be connected to the same internet Wifi of the flask server in order to make API calls to fetch data from the AI recommendation system placed on the flask server. The performance shall depend upon hardware components of the user and server should be enough capable that there should be no delay because of the server. The system shall have maximum of 4 clicks to reach any content. The system shall have high availability. The system shall have lower failure rate. The system must not lag. All the queries and results shall be shown within the maximum time of 10 seconds, returning the accurate results. The system shall be interactive and user friendly as it should be easier for the new user to learn it. The database shall be designed to perform faster queries. Opening, sorting and computing from the database should be carried out as fast as possible.

### 2.7.2 Safety Requirements

The application is not intended to cause any loss, damage, or harm in any way directly or indirectly. And every internal and external communication channel and calls must be secure and ensure safety of user's privacy.

For the accurate and secured system, all users must be logged in to the application before they can access any functionality except registration. The system should be capable to display information with as much clarity as the hardware infrastructure is capable of providing.

### 2.7.3 Security Requirements

The application would be featuring the user authentication service provided by "Firebase". and optionally, there could be integration with Google Auth and LinkedIn, to cater users with one click sign ups.

### 2.7.4 Additional Software Quality Attributes

The software would be adaptable and flexible when it comes to differently formatted resumes. The software would be adaptable to Android users since it is an Android application. The Recommendation Model containing the data would be maintained through the flask server. Security: the main concern is manager's account who is the sole user of the system. Proper login mechanism should be used to avoid hacking. Furthermore, below are the quality attributes system should consider.

**Correctness:** the data should be correct in terms of relevance as well as refinement in order to process the accurate results from the recommendation model.

**Efficiency:** the system should be able to bear the load of data as well as performance tests and work efficiently.

**Service availability:** the system is internet based, it is important to ensure that internet service is always provided in order to interact with the system.

**Response time:** the system should be fast so that users have good experience while using the application

**Reliable:** the system should be reliable as this factor can affect the audience as business model of the application.

**Scalability:** In terms of our model, the performance algorithms are also at best interest since we

hope to exercise the use of big data (In the context of our machines).

**Robustness:** the system should be robust as we aspire to design a model that is validated in terms of all core functionalities of our system.

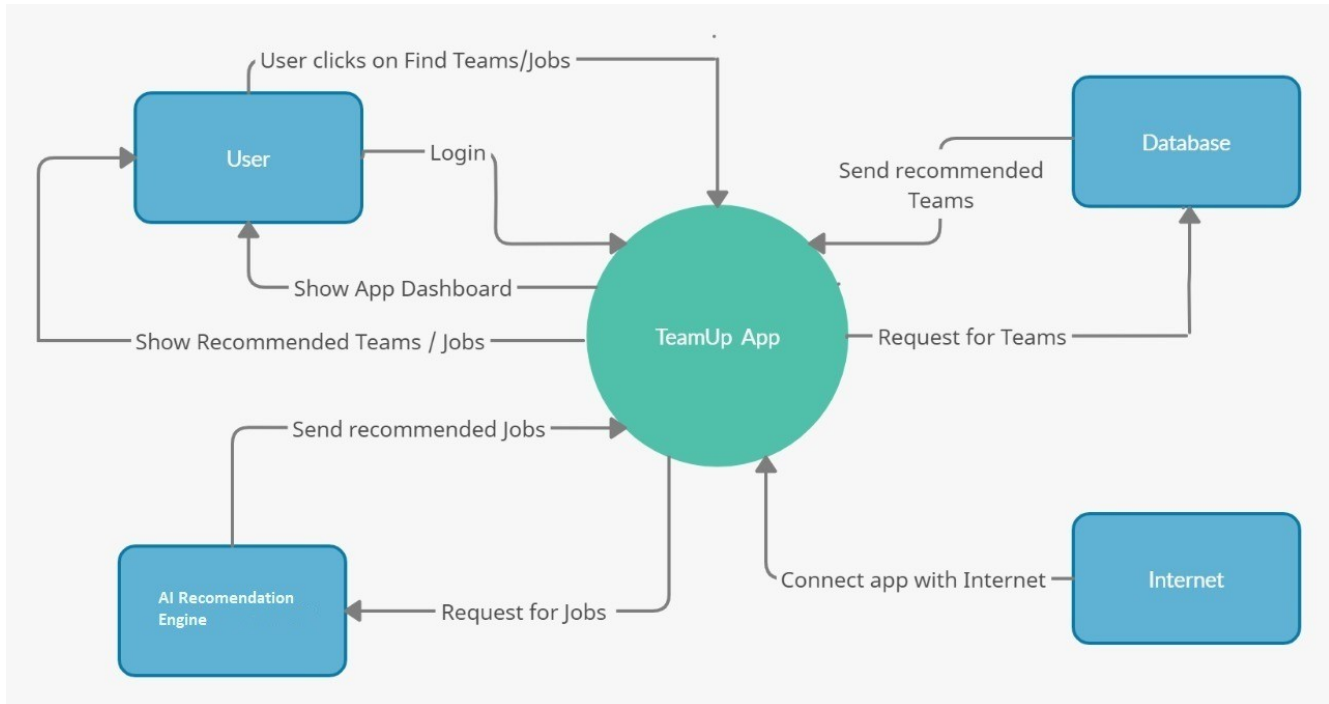
## 2.8 Other Requirements

The hardware requirements would be a 8GB Ram or more. An Ubuntu core 20.0 or higher Operating System. Firebase is the database we would be using.



### 3. System Design

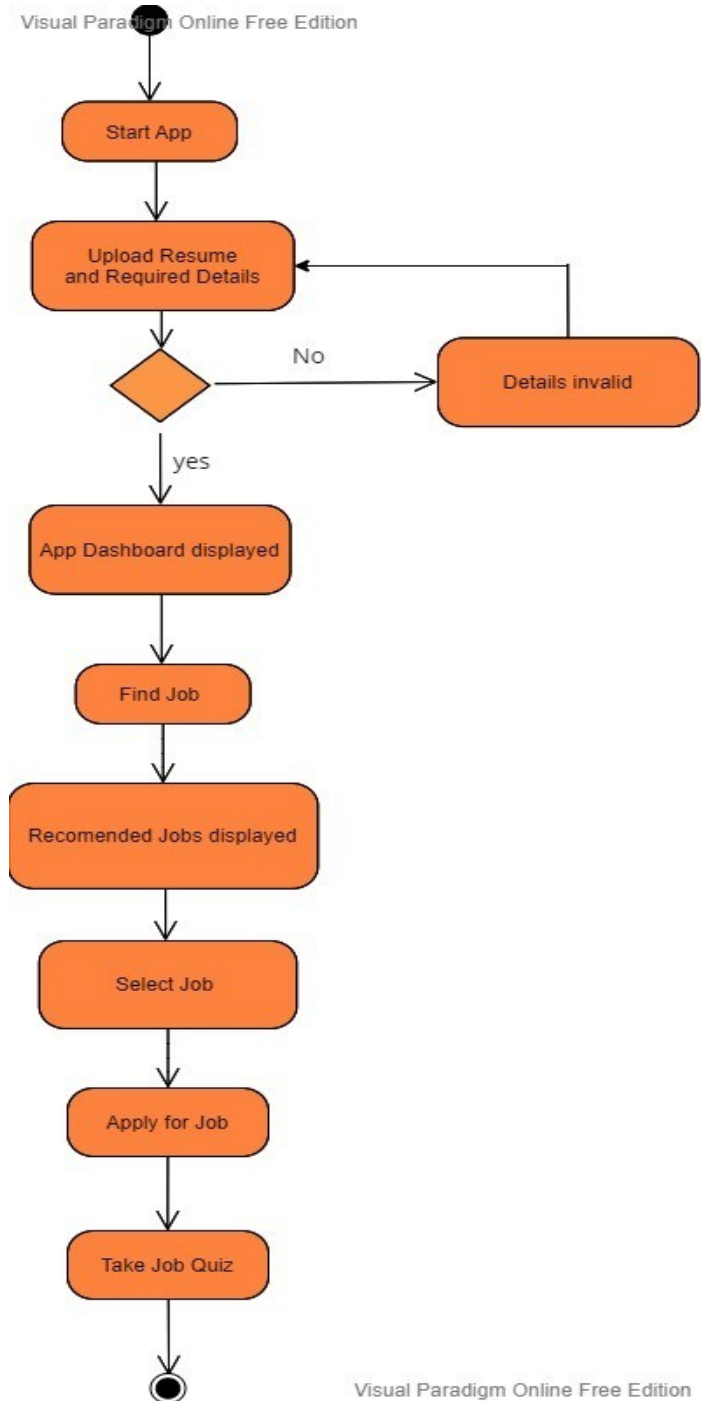
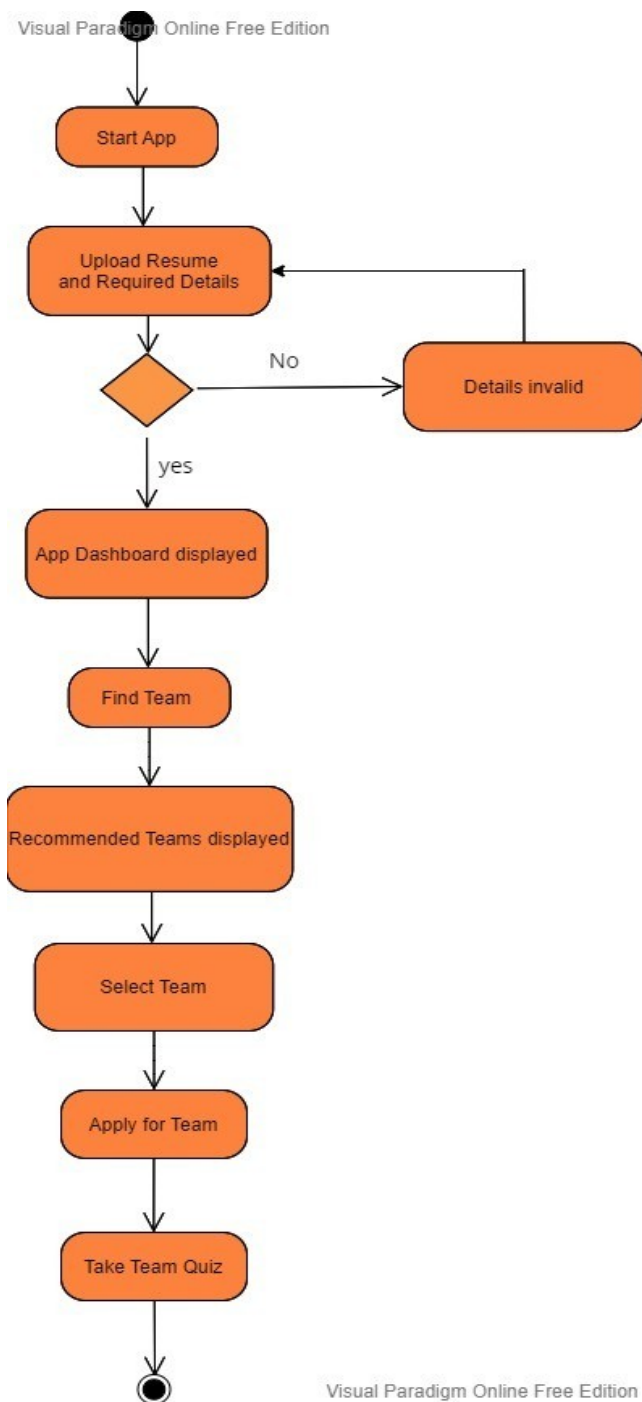
#### 3.1 Application and Data Architecture



**Figure2: Context Diagram**

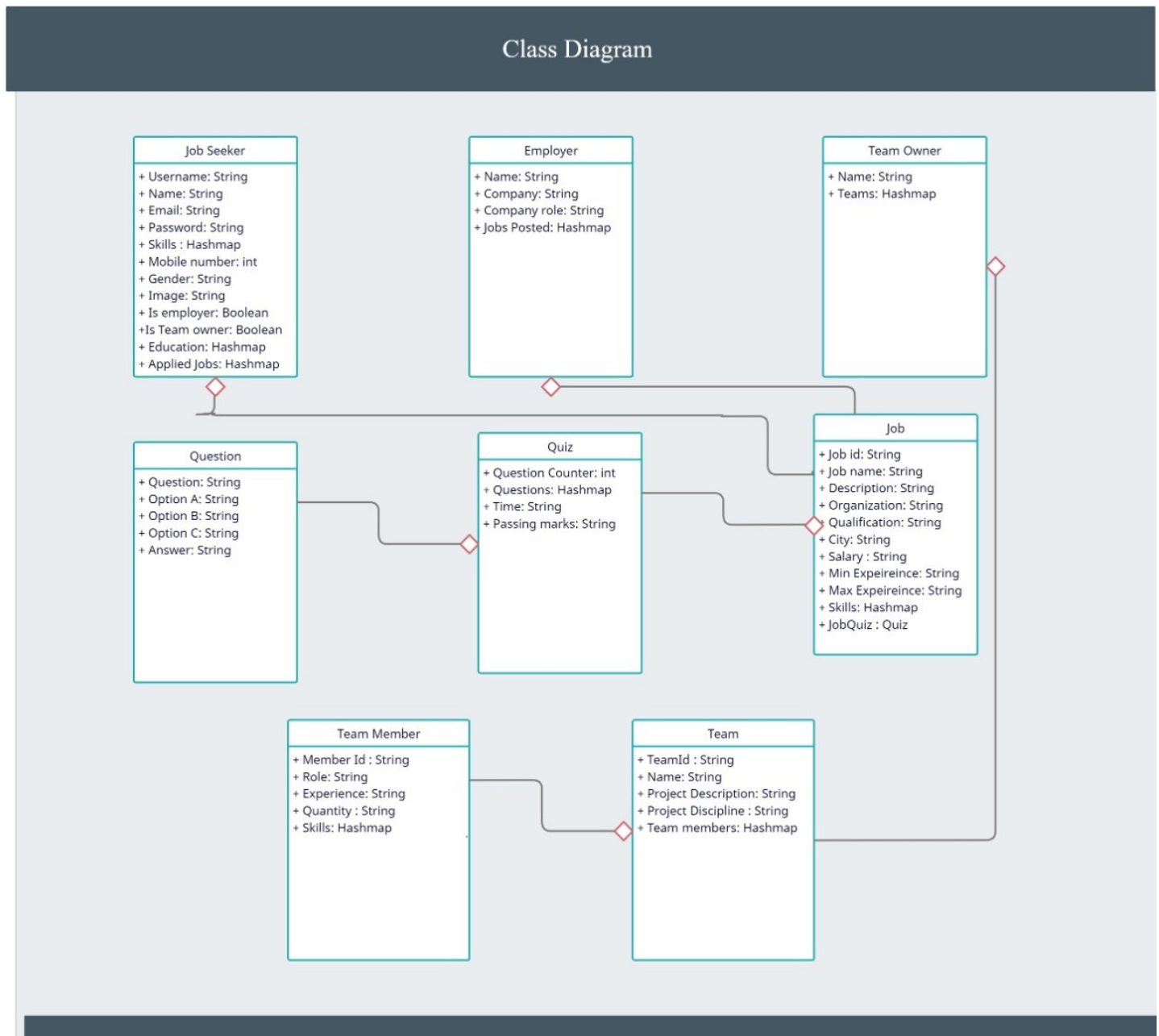
Figure 2 it defines and clarifies the boundaries of our system. It shows the flow of information and data between our system and external entities. Our complete software system is shown in a single diagram. In our system a user will login on the application and click on “Find Jobs” or “Find Teams” on the application. The application will then send a get request to the flask server on which the recommendation engine is placed. The recommendation engine will generate different jobs and they will be sent as a response to the user by the flask server. The teams would be sent to the user by the database.

The mobile application is the central part of our system which acts as the front end. It integrates the user with the back end of our system. The application connects the local server and a user interface for the users which makes a very easy and uncomplicated system for the user in terms of usability.

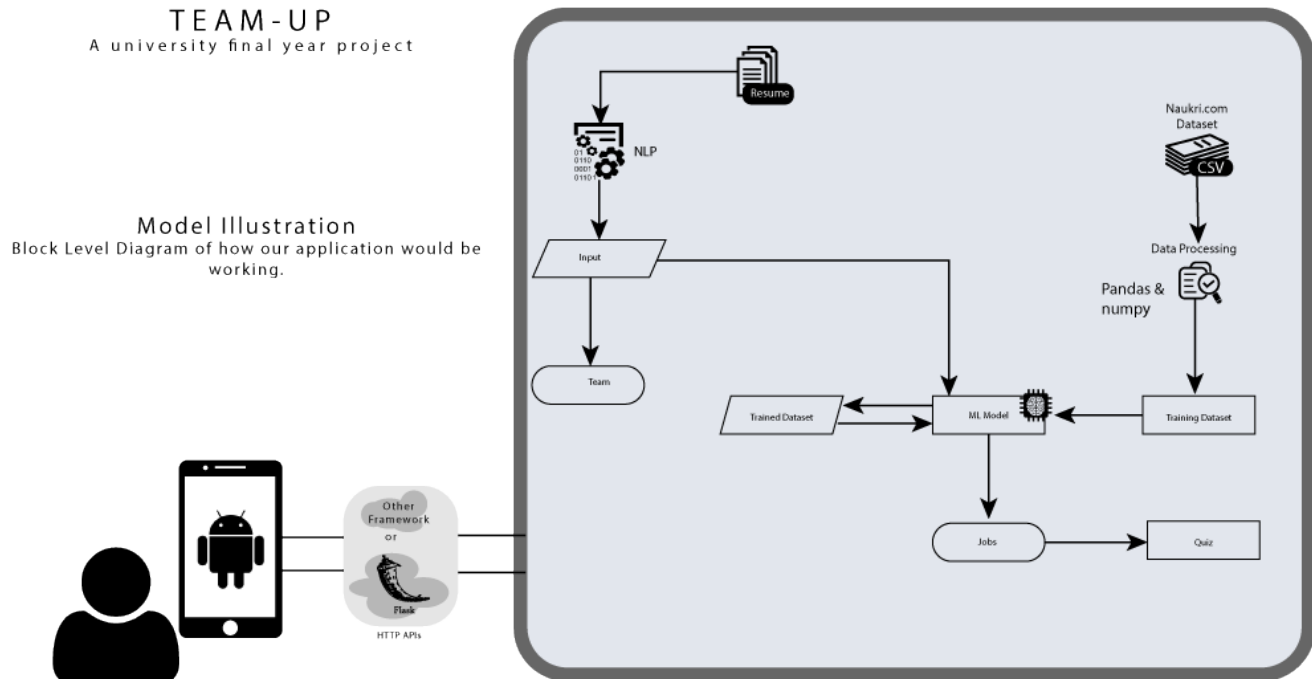


**Figure 3: Activity Diagram**

Both modules ‘Find a Team’ and ‘Find a Job’ are displayed in Figure 3. The diagram shows the flow from one activity to another.



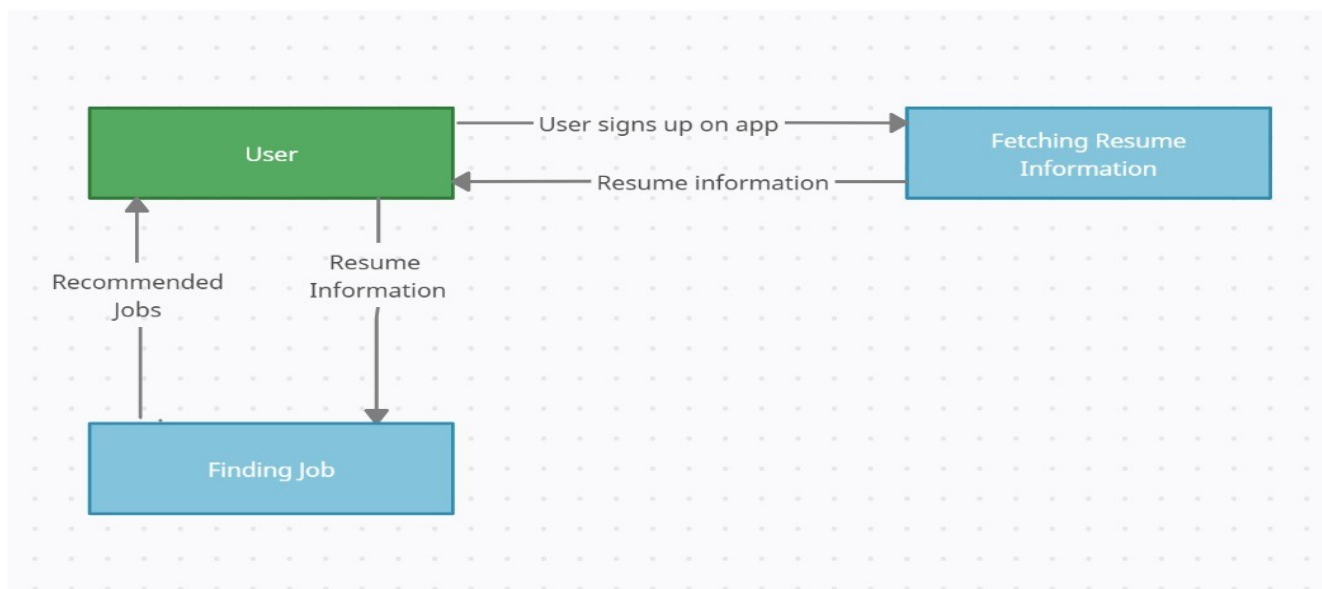
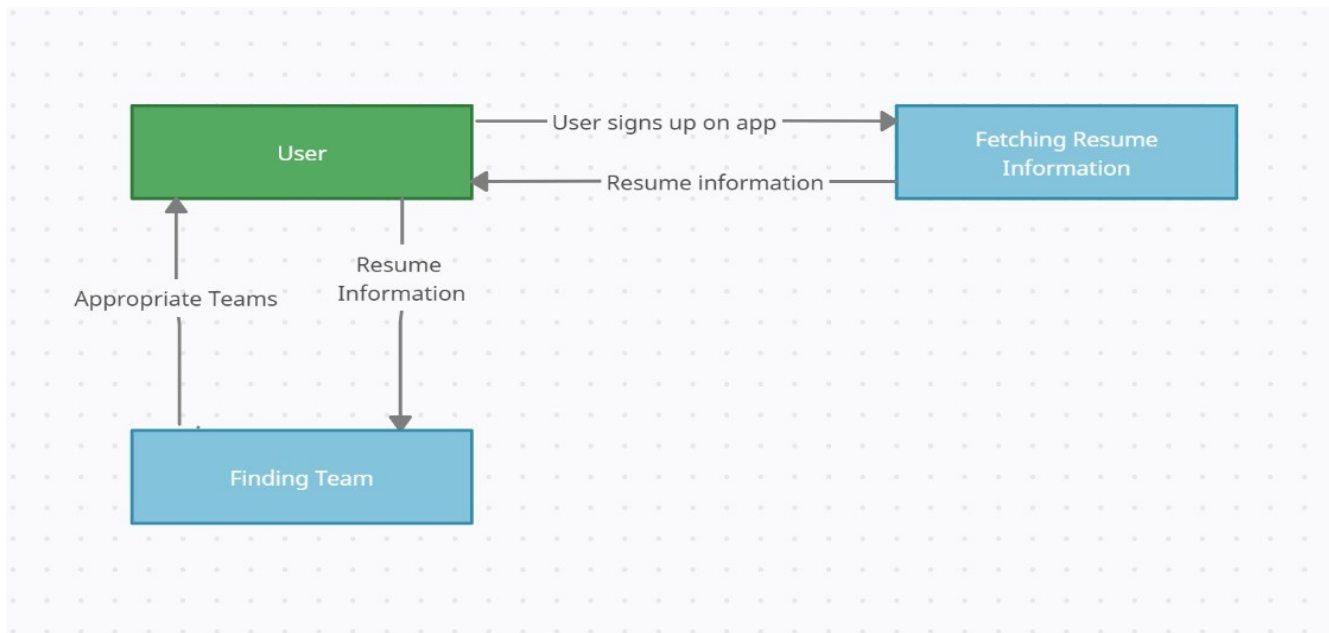
**Figure 4: Class Diagram**



**Figure 5: System Design Block Diagram**

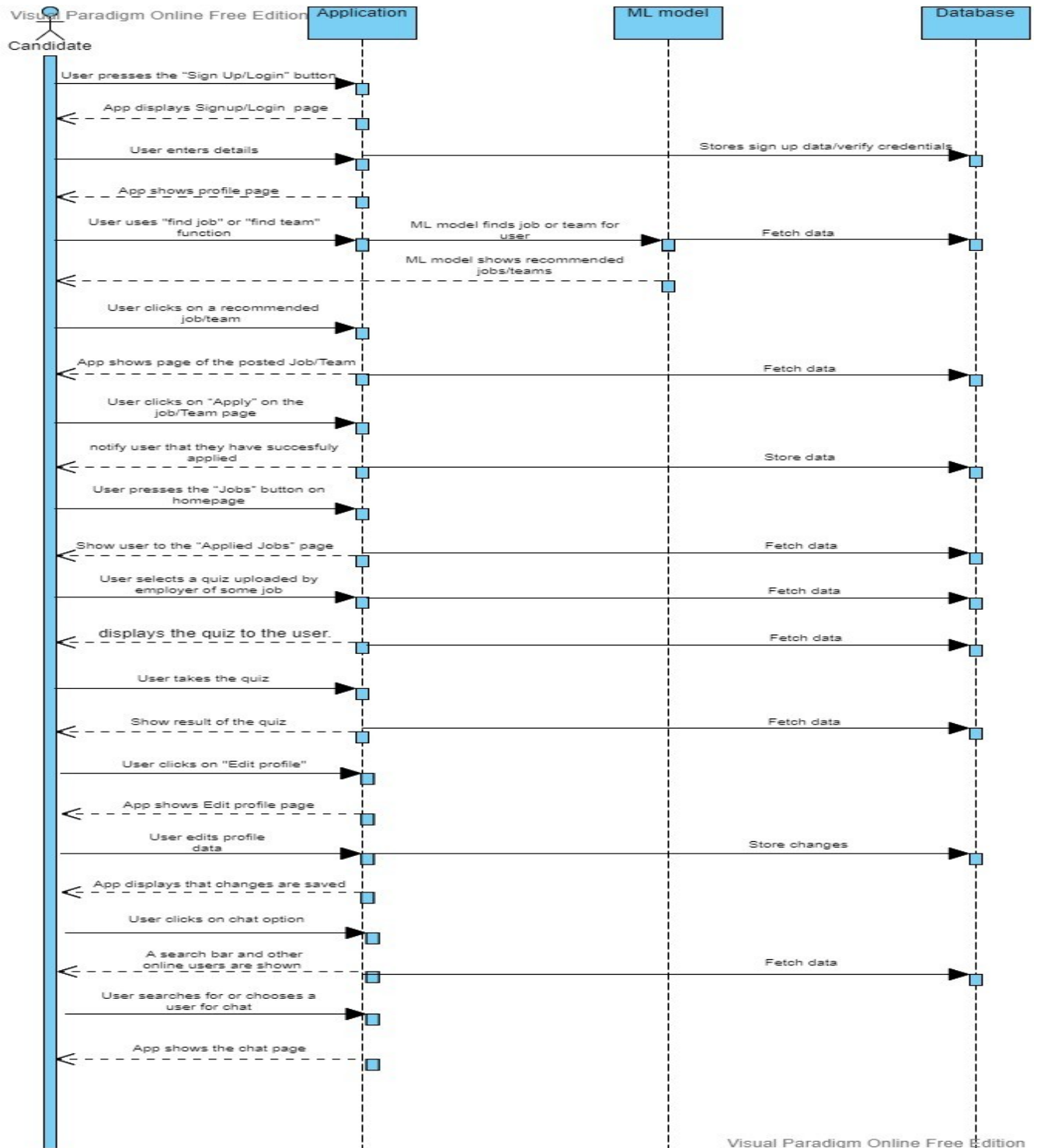
Figure 5 we can see the block level diagram showing how the system would be working. A more detailed diagram is provided in the 'System Architecture' portion.

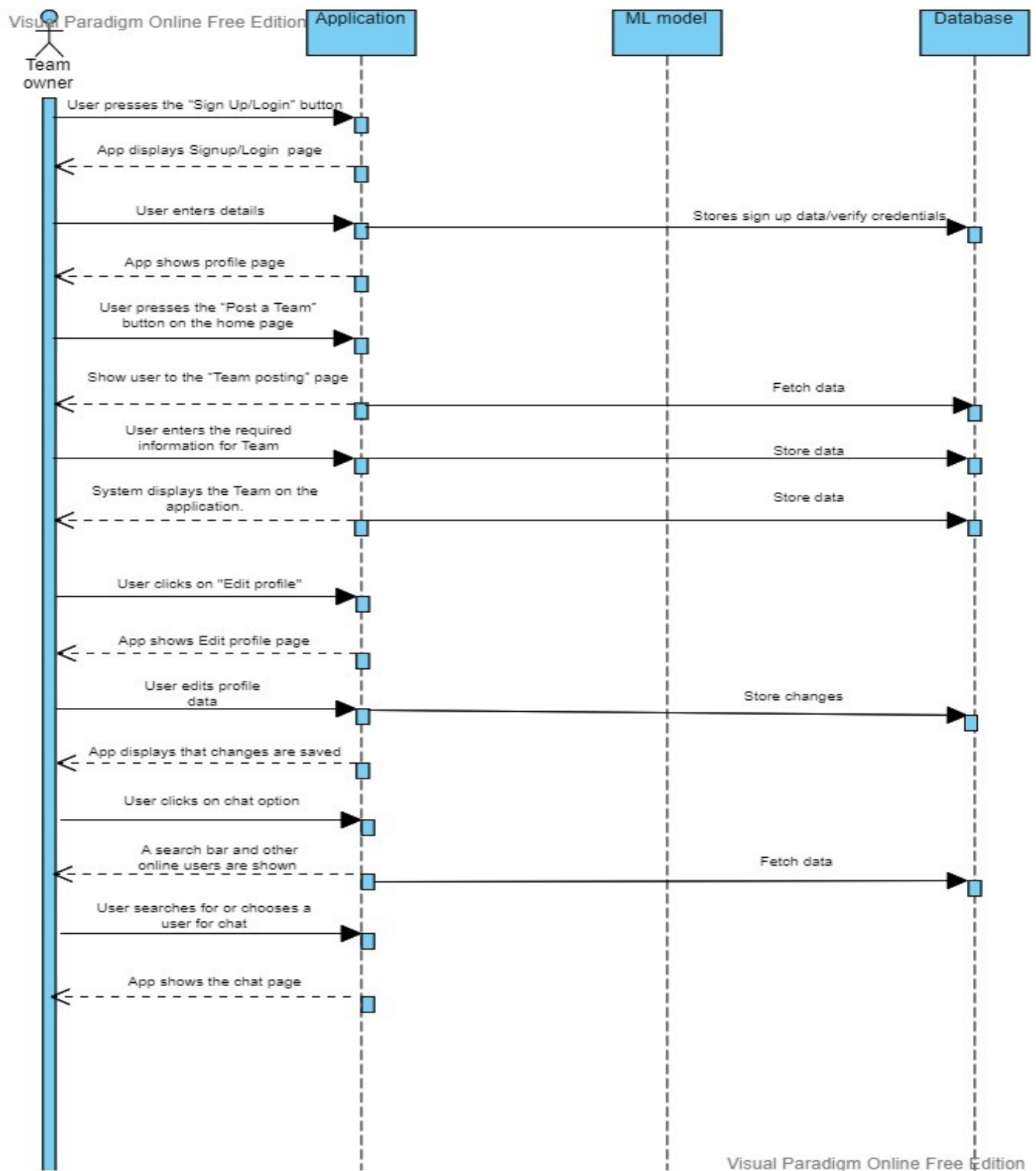
### 3.2 Component Interactions and Collaborations

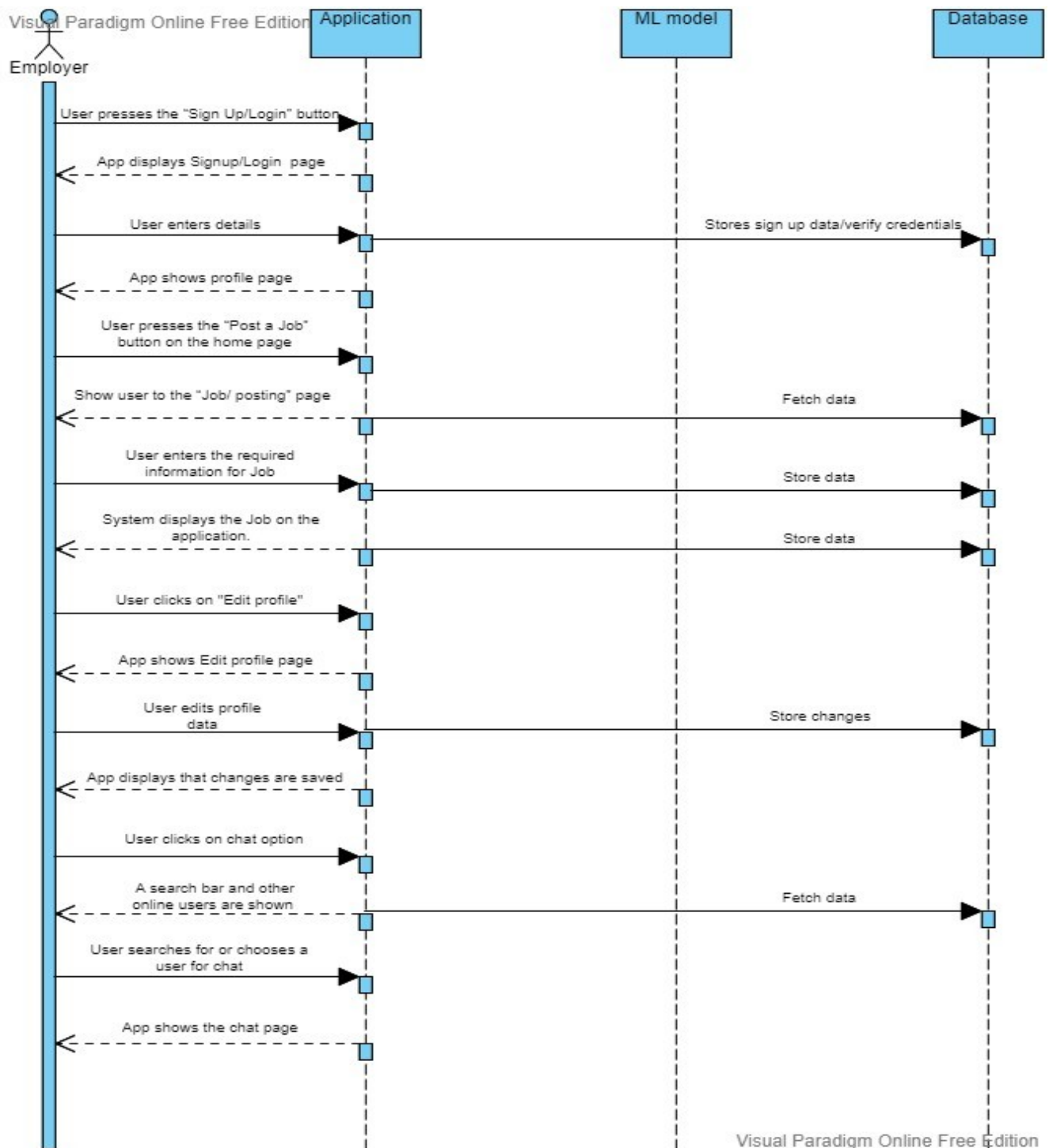


**Figure 6: DFD Diagram**

Figure 6 are the DFD Diagrams of the two main modules 'Find a Job' and 'Find a Team'. The user registers himself on the application with his or her resume. Based on the skill set provided in the resume the user is recommended jobs and teams on the application.





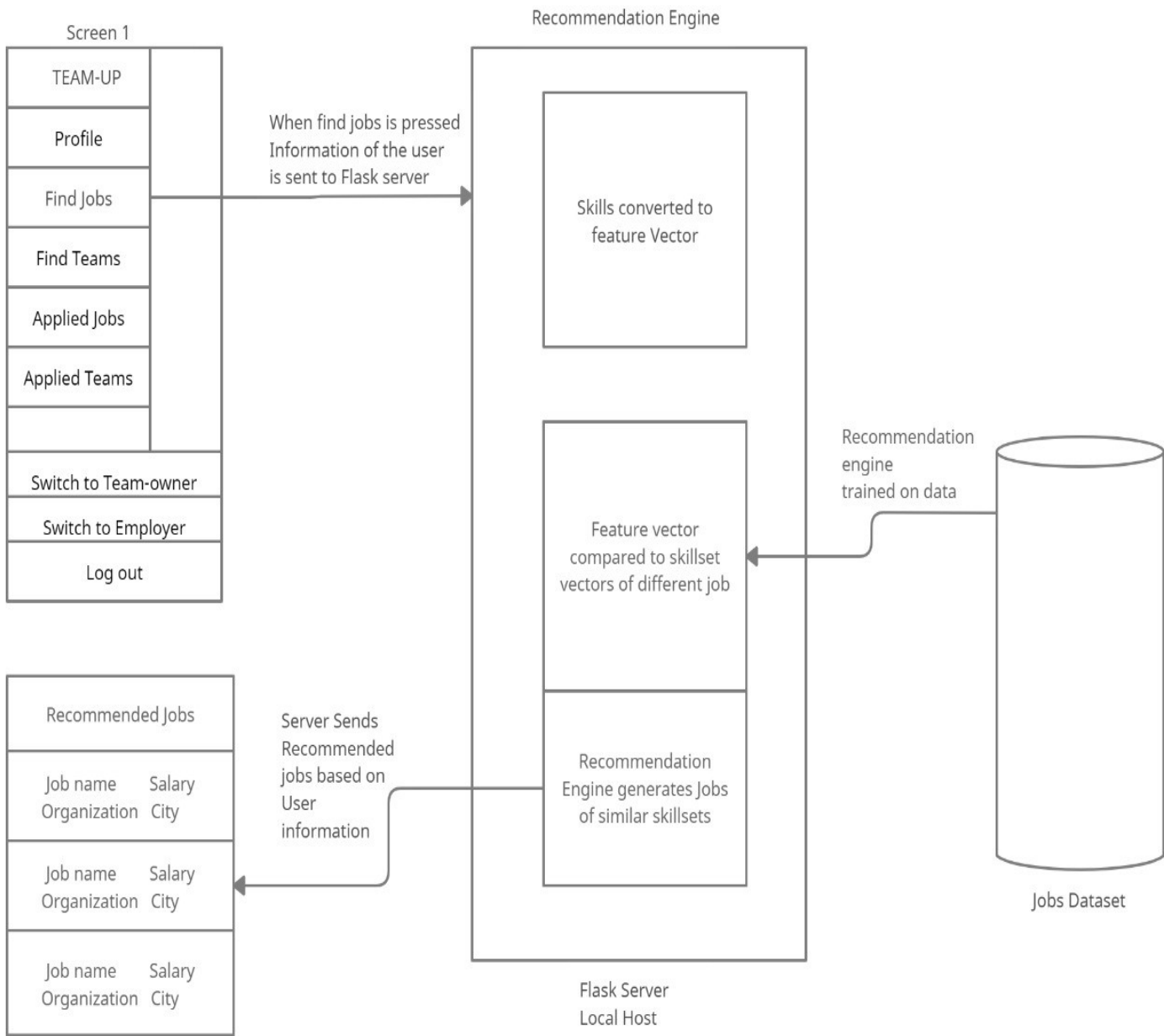


**Figure 7: Sequence Diagram**



Figure 7 shows the sequence diagrams, which show the flow of operations of a user from the perspective of three main user classes.

3.3 System Architecture



**Figure 8: System Architecture Diagram**

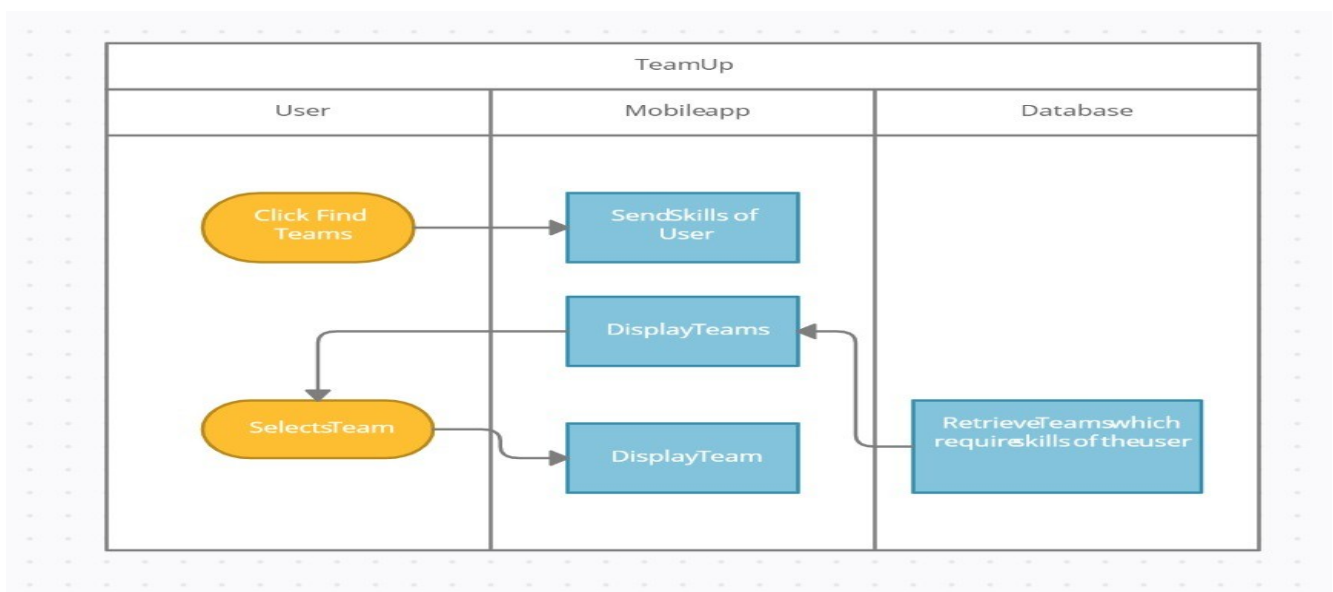
In Figure 8 the detailed architecture structure of our mobile application can be seen. The main functionality of our application is displayed above. The flow shown above is when the user presses the ‘Find Jobs’ button on the application the skillset of the user is sent to the flask server where the recommendation engine is placed. The recommendation engine is trained on the data set we have used which comprises of 28,000 jobs with their respective skillsets. The skillsets of those jobs are converted into feature vectors in numerical form using a Natural Language Processing method known as Count Vectorizer. The skillset of the user sent to the recommendation engine is converted into a numerical feature vector. The recommendation engine checks the similarity score of the user skillset with the skillsets of all the jobs present in the data set. Based on the similarity score the recommendation engine generates recommended jobs to the user. The flask server APIs play a large part in this process. The skillset of the user sent to the recommendation engine is through a “POST” request API call and the recommended jobs displayed are received by the mobile application through a “GET” request API call.

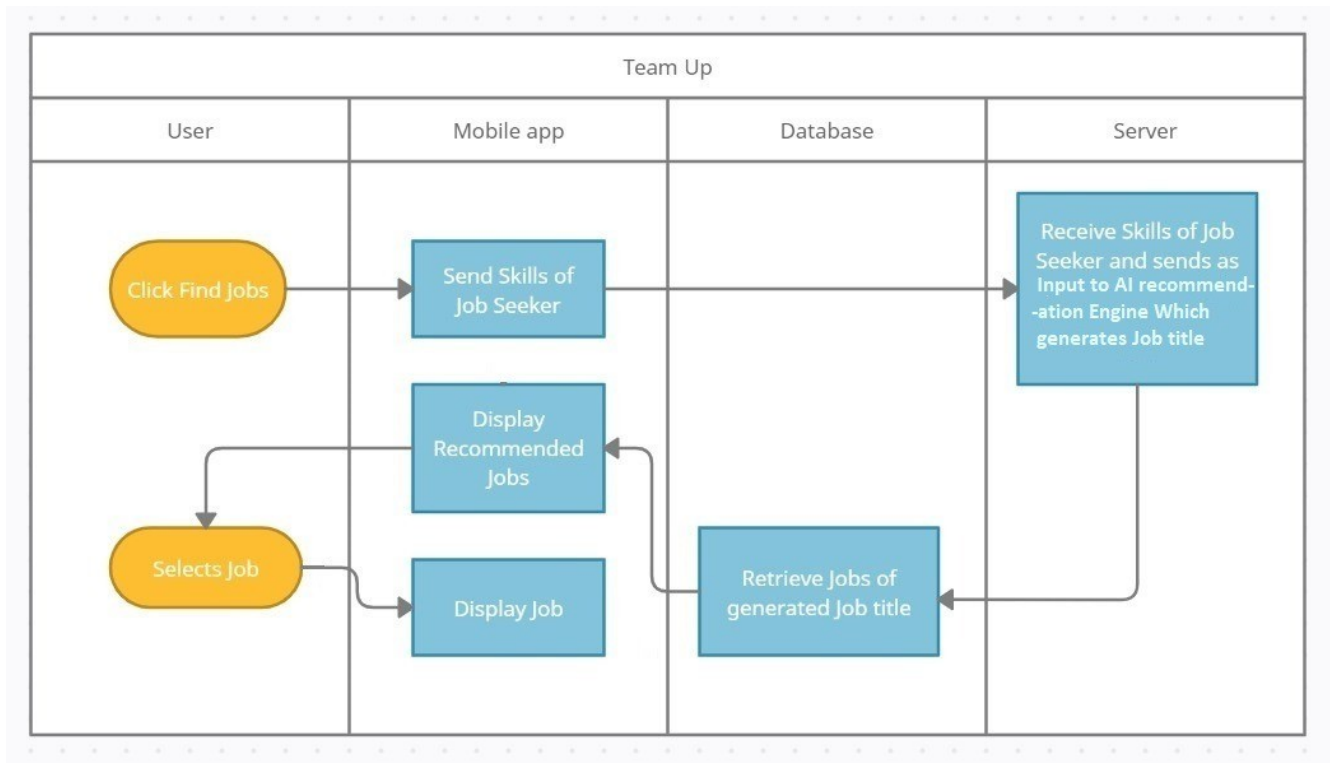
### 3.4 Architecture Evaluation

The four main technologies which we have used for this application are Android, Flask, Python, and Firebase. We have used Android for its popularity level. Android applications have the largest market in the world. Python is used to create the Recommendation engine because of its level of user friendly and power. Flask is used because of its ability to scale applications despite their complexity. Firebase is used because it prevents developers from writing a lot of base code and it can deal with scalability when it comes to dealing with large amounts of data.

### 3.5 Screenshots/Prototype

#### Workflow



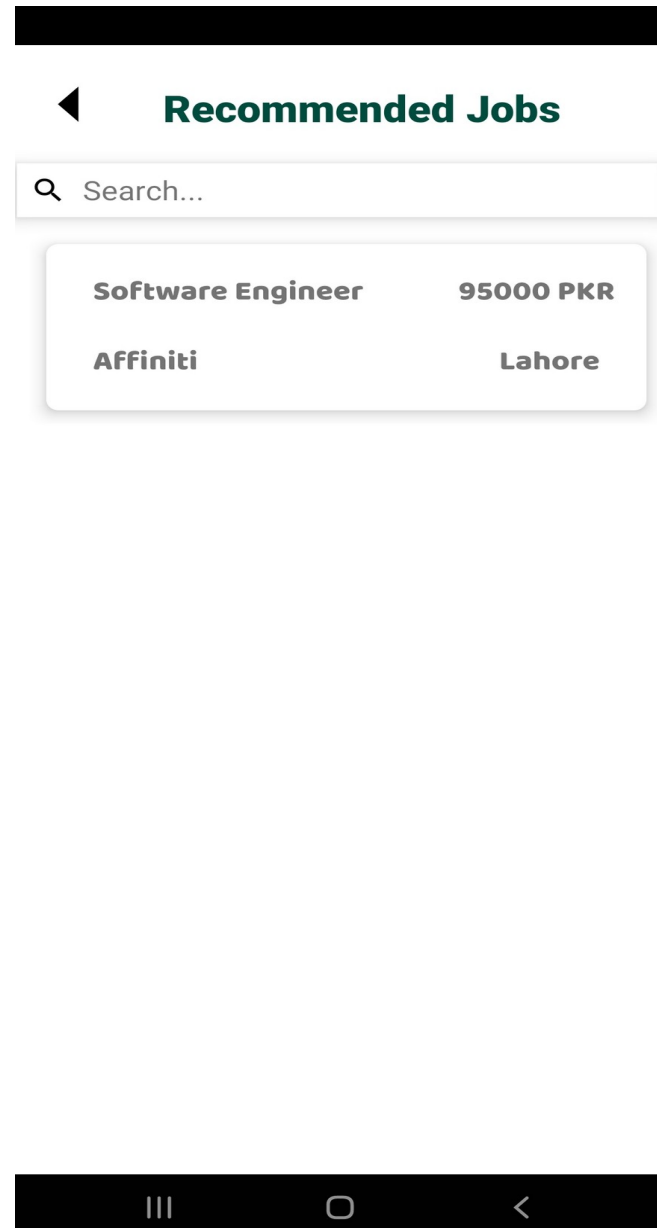
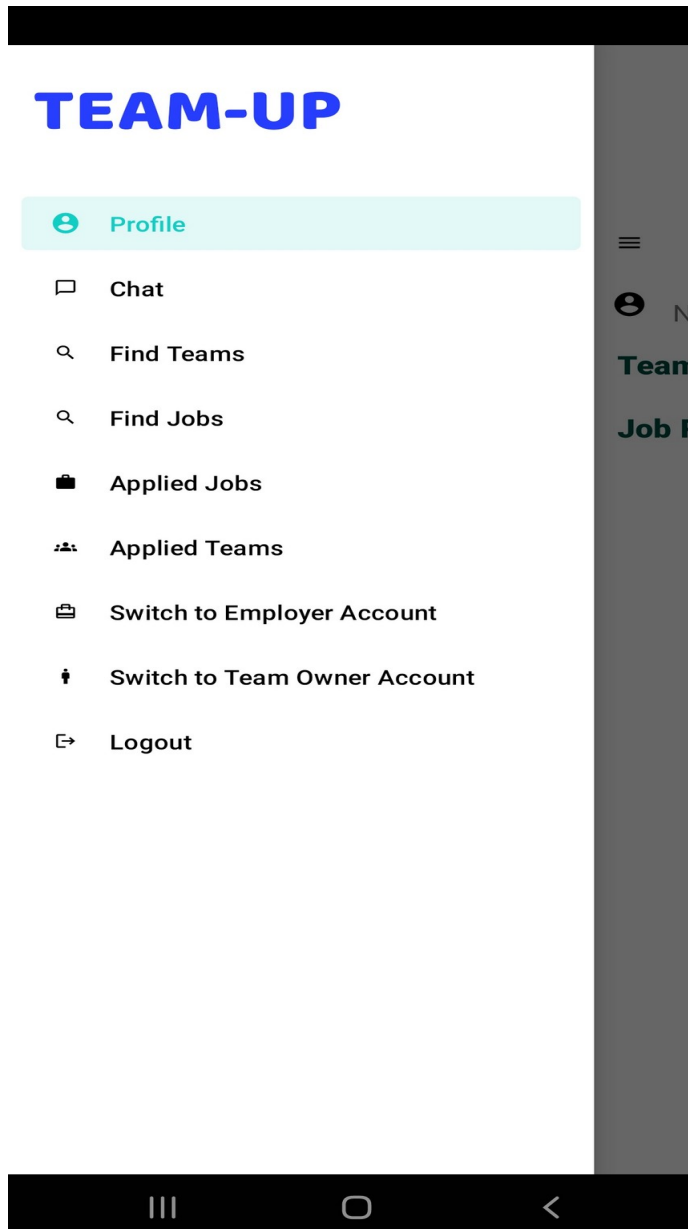


**Figure 9: Swim Lane Diagram**

### 3.6 Screens

The screens below show the work flow of two modules “Find Job” and “Recruit Job Seeker”.

### 3.6.1 Find Job



01:10

29%



## Job

### Job Information

Software Engineer      Salary: 95000

Affiniti      Min Exp: 2 years

Lahore      Max Exp: 4 years

### Role and Responsibilities

Build front ends of web and mobile applications. Collaborate with junior developers.

### Required Qualification

BS Computer Science

### Job Skills Required

CSS , Java Script , Html

**Take Job Quiz**



01:10

29%

## Quiz

question1

a

b

c

d



-14:00



01:10

29%



## Quiz Score

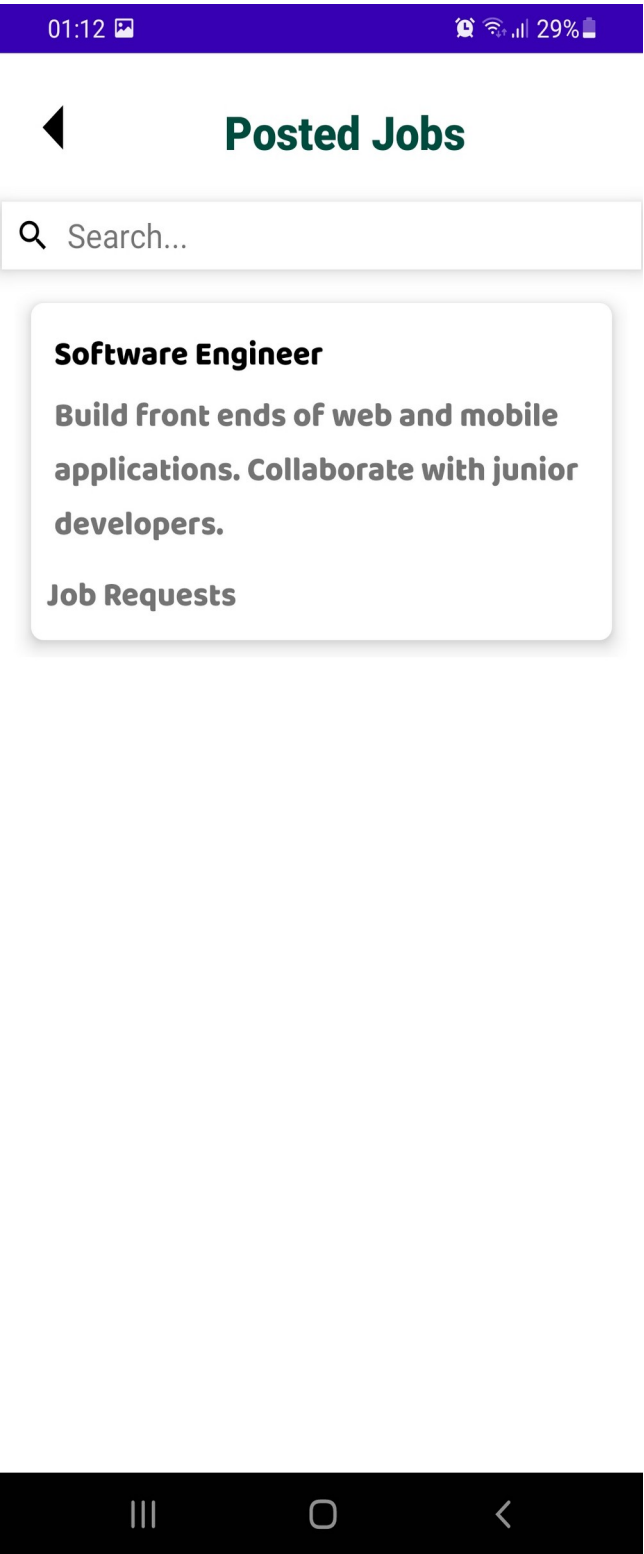
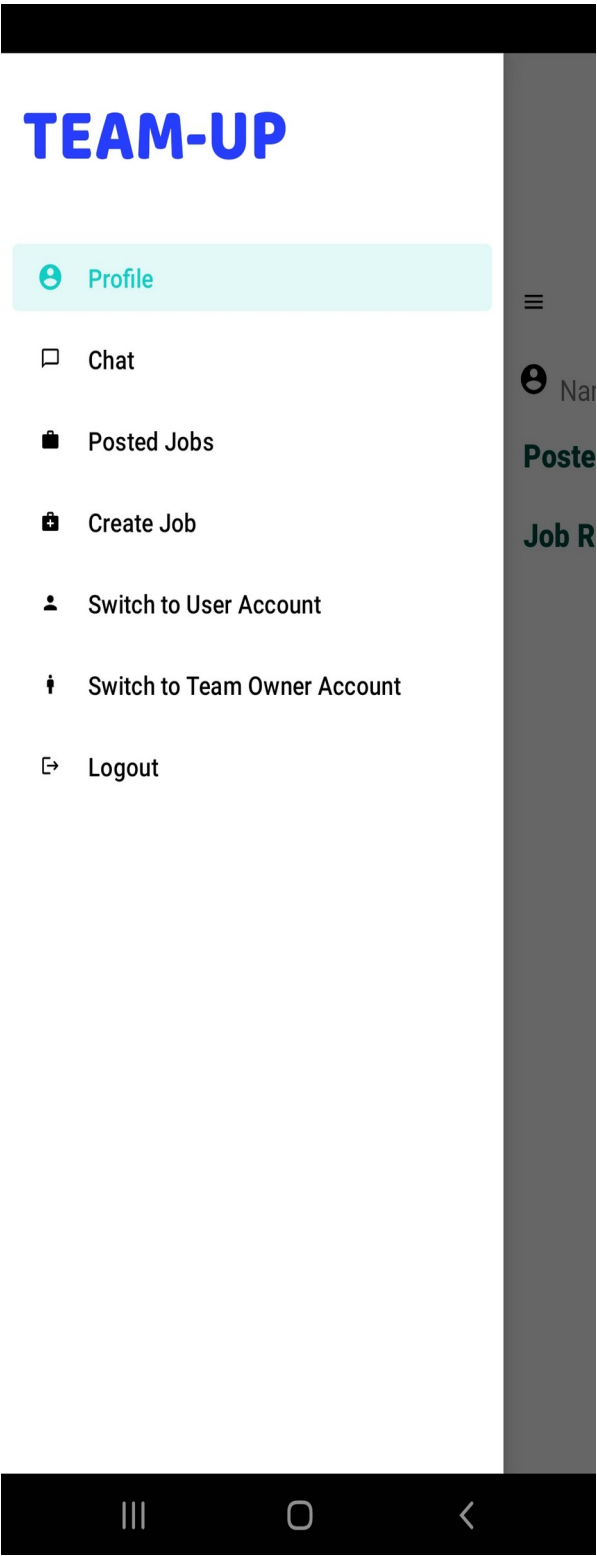
# Your Score

1/1

Congratulations , you have passed the Quiz! You will receive an interview date soon.



3.6.2 Recruit Job Seeker



## Job

**Job Information**

Software Engineer

Affiniti

Salary: 95000

Lahore

Min Exp: 2 years

Qualification

Max Exp: 4 years

**Role and Responsibilities**


Build front ends of web and mobile applications. Collaborate with junior developers.

**Job Skills Required**


CSS , Java Script , Html


Edit Job Quiz

## Passing Applicants





**Razi Ul Hasan**  
1/1 marks






**Mohammad Faizan**  
1/1 marks  
7 Feb 2022

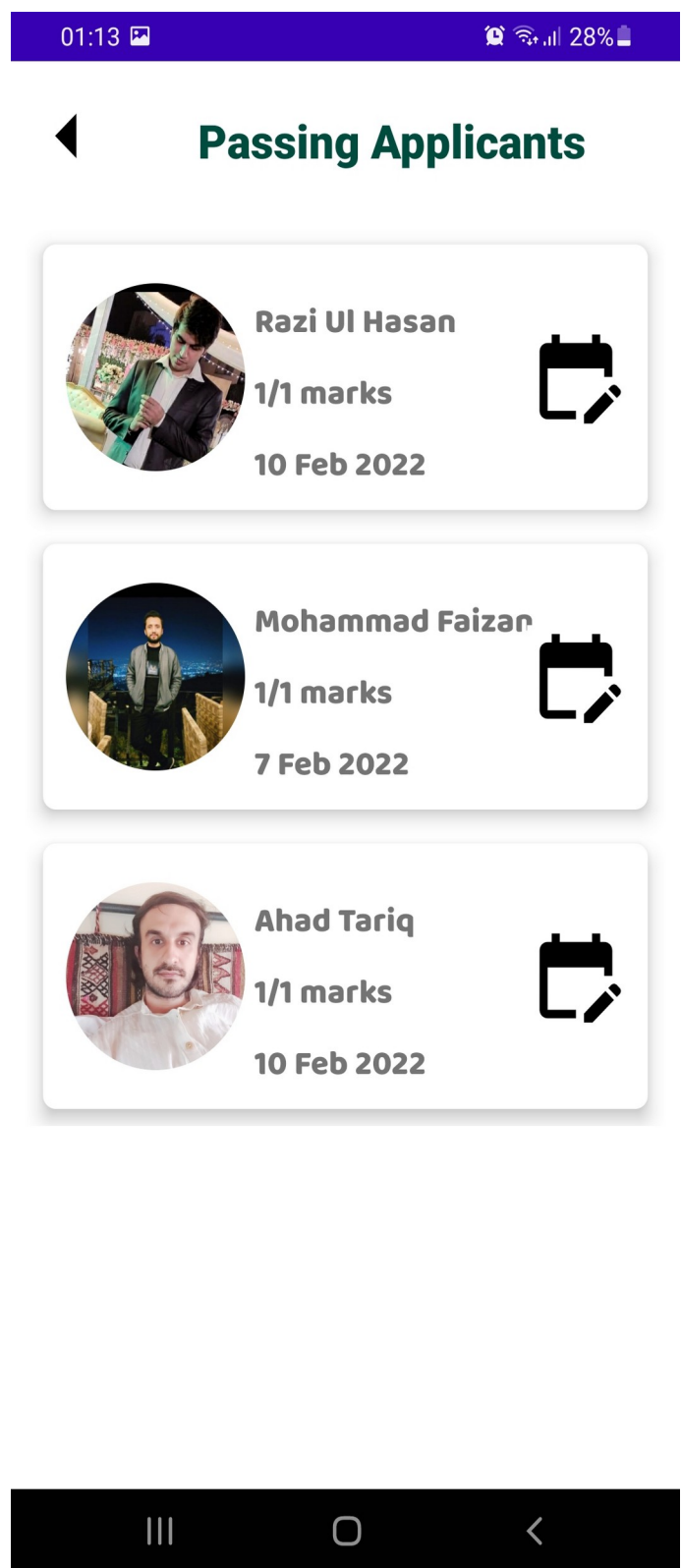
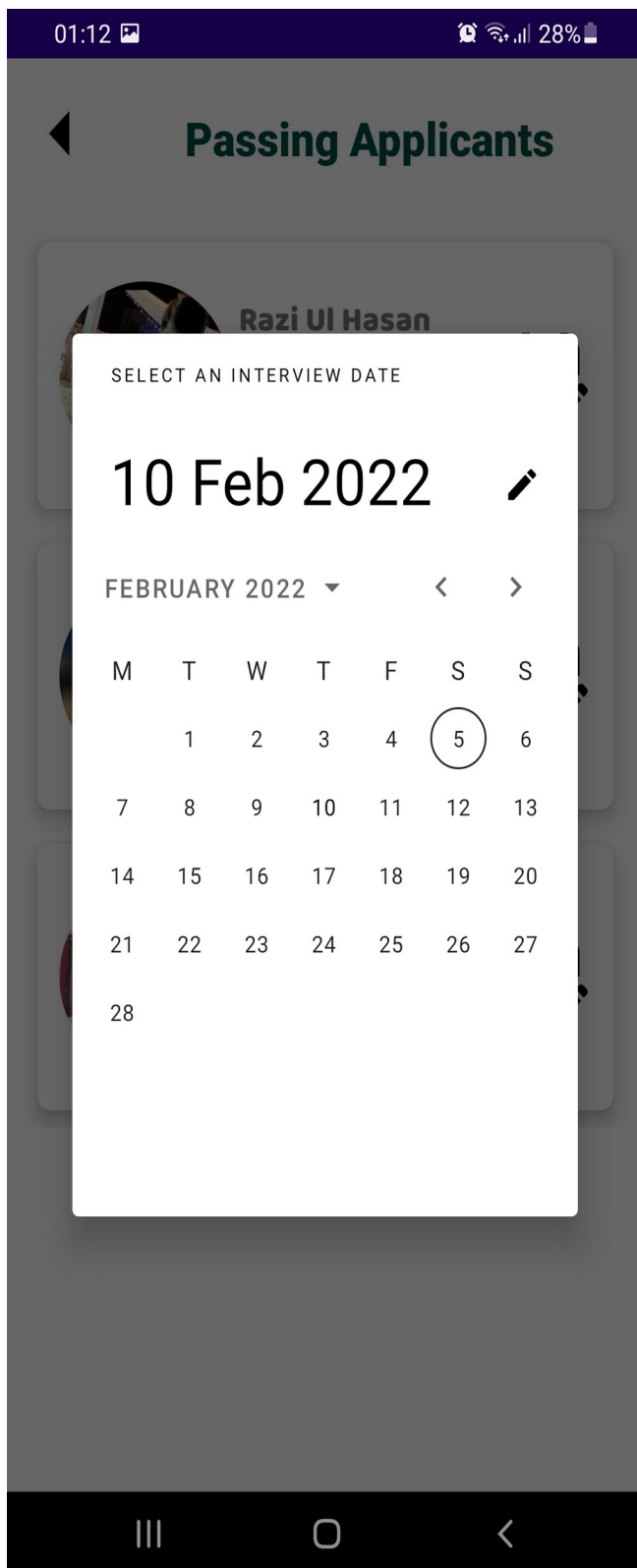




**Ahad Tariq**  
1/1 marks  
10 Feb 2022







## 4. Test Specification and Results

### Test Case Specification

<b>Identifier</b>	TC-1
<b>Related requirements(s)</b>	UC-1
<b>Short description</b>	Check if the user can sign up on an application.
<b>Pre-condition(s)</b>	The person should have a name, email address, mobile number, and resume.
<b>Input data</b>	Enter the required information.
<b>Detailed steps</b>	1) Enter required information 2) Press Sign Up Button
<b>Expected result(s)</b>	User signs up on the application.
<b>Post-condition(s)</b>	User Dashboard opens.
<b>Actual result(s)</b>	User signed up on the application.
<b>Test Case Result</b>	PASS

**Table 6.1: TC-1**

<b>Identifier</b>	TC-2
<b>Related requirements(s)</b>	UC-2
<b>Short description</b>	Check if the user can log in the application.
<b>Pre-condition(s)</b>	The person should have a valid email address and password combination.
<b>Input data</b>	Enter the email address and password combination.
<b>Detailed steps</b>	1) Enter email address and password 2) Press Log In Button
<b>Expected result(s)</b>	User logs in the application.
<b>Post-condition(s)</b>	User Dashboard opens.
<b>Actual result(s)</b>	User logged in the application.
<b>Test Case Result</b>	PASS

**Table 6.2: TC-2**

<b>Identifier</b>	TC-3
<b>Related requirements(s)</b>	UC-3
<b>Short description</b>	Check if an employer can post a job on the application.
<b>Pre-condition(s)</b>	The employer posting a job must belong to a real world organization which has a job opening.
<b>Input data</b>	The employer enters the required job information.
<b>Detailed steps</b>	1) Enter required job information 2) Press Finish Job Post Button
<b>Expected result(s)</b>	Job gets posted on the application.
<b>Post-condition(s)</b>	Job Details page opens.
<b>Actual result(s)</b>	Job Details page opens on the screen.
<b>Test Case Result</b>	PASS

**Table 6.3: TC-3**

<b>Identifier</b>	TC-4
<b>Related requirements(s)</b>	UC-4
<b>Short description</b>	Check if an employer can view his posted jobs.
<b>Pre-condition(s)</b>	The employer must have posted a job or jobs.
<b>Input data</b>	-
<b>Detailed steps</b>	1) Open the navigation drawer 2) Click on My Jobs
<b>Expected result(s)</b>	Posted jobs get displayed.
<b>Post-condition(s)</b>	Posted jobs get displayed.
<b>Actual result(s)</b>	Posted jobs are displayed.
<b>Test Case Result</b>	PASS

**Table 6.4: TC-4**

<b>Identifier</b>	TC-5
<b>Related requirements(s)</b>	UC-5
<b>Short description</b>	Check if an employer can view a specific posted job.
<b>Pre-condition(s)</b>	The employer must have posted the specific job.
<b>Input data</b>	-
<b>Detailed steps</b>	1) Open the navigation drawer 2) Click on My Jobs 3) Employer selects the job
<b>Expected result(s)</b>	Job is displayed.
<b>Post-condition(s)</b>	Job is displayed.

<b>Actual result(s)</b>	Job is displayed.
<b>Test Case Result</b>	PASS

**Table 6.5: TC-5**

<b>Identifier</b>	TC-6
<b>Related requirements(s)</b>	UC-6
<b>Short description</b>	Check if a team owner can create a team on the application.
<b>Pre-condition(s)</b>	The team owner should have a project for which he wants to build a team.
<b>Input data</b>	The team owner enters the required team information.
<b>Detailed steps</b>	1) Enter required team information. 2) Press Finish Team Post Button.
<b>Expected result(s)</b>	Team gets posted on the application.
<b>Post-condition(s)</b>	Team details page opens.
<b>Actual result(s)</b>	Team details page opens on the screen.
<b>Test Case Result</b>	PASS

**Table 6.6: TC-6**

<b>Identifier</b>	TC-7
<b>Related requirements(s)</b>	UC-7
<b>Short description</b>	Check if a team owner can view his posted teams.
<b>Pre-condition(s)</b>	The team owner must have posted a team or teams.
<b>Input data</b>	-
<b>Detailed steps</b>	1) Open the navigation drawer 2) Click on My Teams
<b>Expected result(s)</b>	Posted teams are displayed.
<b>Post-condition(s)</b>	Posted teams are displayed.
<b>Actual result(s)</b>	Posted teams are displayed.
<b>Test Case Result</b>	PASS

**Table 6.7: TC-7**

<b>Identifier</b>	TC-8
-------------------	------

<b>Related requirements(s)</b>	UC-8
<b>Short description</b>	Check if a team owner can view a specific posted team.
<b>Pre-condition(s)</b>	The team owner must have posted the specific team.
<b>Input data</b>	-
<b>Detailed steps</b>	1) Open the navigation drawer. 2) Click on My Teams. 3) Team Owner selects the team.
<b>Expected result(s)</b>	Team is displayed.
<b>Post-condition(s)</b>	Team is displayed.
<b>Actual result(s)</b>	Team is displayed.
<b>Test Case Result</b>	PASS

**Table 6.8: TC-8**

<b>Identifier</b>	TC-9
<b>Related requirements(s)</b>	UC-9
<b>Short description</b>	Check if an employer can edit information of a posted job.
<b>Pre-condition(s)</b>	The employer must have posted the job on the application.
<b>Input data</b>	The employer enters the new information.
<b>Detailed steps</b>	1) Click on the specific information of the job to be edited. 2) Enter new information.
<b>Expected result(s)</b>	Job with new information is updated.
<b>Post-condition(s)</b>	Job details page with new information is displayed.
<b>Actual result(s)</b>	Job with new information is updated.
<b>Test Case Result</b>	PASS

**Table 6.9: TC-9**

<b>Identifier</b>	TC-10
<b>Related requirements(s)</b>	UC-10
<b>Short description</b>	Check if a team owner can edit information of a specific team.
<b>Pre-condition(s)</b>	The team owner must have posted the team on the application.
<b>Input data</b>	The team owner enters the new information.
<b>Detailed steps</b>	1) Click on the specific information of the team to be edited. 2) Enter new information.
<b>Expected result(s)</b>	Team with new information is updated.
<b>Post-condition(s)</b>	Team details page with new information is displayed.
<b>Actual result(s)</b>	Team with the new information is updated.

<b>Test Case Result</b>	PASS
-------------------------	------

**Table 6.10: TC-10**

<b>Identifier</b>	TC-11
<b>Related requirements(s)</b>	UC-11
<b>Short description</b>	Check if a team owner can edit information of a specific team role.
<b>Pre-condition(s)</b>	The team owner must have created the team role for a posted team on the application.
<b>Input data</b>	The team owner enters the new information.
<b>Detailed steps</b>	1) Click on the specific team role on the team page to be edited. 2) Enter new information.
<b>Expected result(s)</b>	Team role with new information is updated.
<b>Post-condition(s)</b>	Team role details with new information is displayed.
<b>Actual result(s)</b>	Team role with new information is updated.
<b>Test Case Result</b>	PASS

**Table 6.11: TC-11**

<b>Identifier</b>	TC-12
<b>Related requirements(s)</b>	UC-12
<b>Short description</b>	Check if a team owner can add a new role on the application.
<b>Pre-condition(s)</b>	The team owner must have created the team on the application to which he wants to add a new role.
<b>Input data</b>	The team owner enters the new role information.
<b>Detailed steps</b>	1) The team owner enters the role information. 2) Press finish team role button.
<b>Expected result(s)</b>	New team role is added.
<b>Post-condition(s)</b>	Team page with new team role opens.
<b>Actual result(s)</b>	Team Role in the team is added.
<b>Test Case Result</b>	PASS

**Table 6.12: TC-12**

<b>Identifier</b>	TC-13
<b>Related requirements(s)</b>	UC-13
<b>Short description</b>	Check if an employer can create a quiz on the application.

<b>Pre-condition(s)</b>	The employer must have created the job on the application for which wants to create a quiz.
<b>Input data</b>	The employer adds questions to the application.
<b>Detailed steps</b>	1) The employer clicks on the create quiz button. 2) The employer adds questions to the quiz. 3) The employer inputs the passing marks and duration of the quiz.
<b>Expected result(s)</b>	Quiz is created.
<b>Post-condition(s)</b>	Job page opens.
<b>Actual result(s)</b>	Quiz is created.
<b>Test Case Result</b>	PASS

**Table 6.13: TC-13**

<b>Identifier</b>	TC-14
<b>Related requirements(s)</b>	UC-14
<b>Short description</b>	Check if a job seeker can take a quiz on the application.
<b>Pre-condition(s)</b>	The job seeker must be registered on the application.
<b>Input data</b>	-
<b>Detailed steps</b>	1) The job seeker clicks on the take quiz button. 2) The job seeker answers the quiz question and moves to the next questions.
<b>Expected result(s)</b>	The job seeker finishes the job quiz on the application and his or her score gets saved.
<b>Post-condition(s)</b>	The quiz score screen gets displayed.
<b>Actual result(s)</b>	The quiz score of the job seeker gets recorded.
<b>Test Case Result</b>	PASS

**Table 6.14: TC-14**

<b>Identifier</b>	TC-15
<b>Related requirements(s)</b>	UC-15
<b>Short description</b>	Check if a job seeker can find a job on the application.
<b>Pre-condition(s)</b>	The job seeker must be registered on the application.
<b>Input data</b>	-
<b>Detailed steps</b>	1) The job seeker clicks on the navigation drawer and presses the find jobs button.
<b>Expected result(s)</b>	Recommended jobs get displayed to the user.
<b>Post-condition(s)</b>	Recommended jobs get displayed to the user.
<b>Actual result(s)</b>	Recommended jobs get displayed to the user.
<b>Test Case Result</b>	PASS

**Table 6.15: TC-15**

<b>Identifier</b>	TC-16
<b>Related requirements(s)</b>	UC-16
<b>Short description</b>	Check if a team seeker can find a team on the application.
<b>Pre-condition(s)</b>	The team seeker must be registered on the application.
<b>Input data</b>	-
<b>Detailed steps</b>	1) The team seeker clicks on the navigation drawer and presses the find team's button.
<b>Expected result(s)</b>	Recommended teams get displayed to the user.
<b>Post-condition(s)</b>	Recommended teams get displayed to the user.
<b>Actual result(s)</b>	Recommended teams get displayed to the user.
<b>Test Case Result</b>	PASS

**Table 6.16: TC-16**

## 4.2 Summary of Test Results

Module Name	Test cases run	Number of defects found	Number of defects corrected so far	Number of defects still need to be corrected
<b>Module 1 to Module 14</b>	TC1, TC2, TC3, TC4, TC5, TC6, TC7, TC8, TC9, TC10, TC11, TC12, TC13, TC14	0	0	0
<b>Module 15 and Module 16</b>	TC 15, TC 16	2	2	2
	...			
<b>Complete System</b>	TC1 , TC2, TC3, TC4, TC5, TC6, TC7, TC8, TC9, TC10, TC11, TC12, TC13, TC14	2	2	2

**Table 6.2: Summary of All Test Results**



## 5. Conclusion and Future Work

### Project summary

The final product gives unemployed individuals and fresh graduates a platform to create teams in order to build projects. Building projects in teams would help these people develop a professional skill set to enhance their resume and equip them to pursue a full-time job.

### Problems faced and lessons learned

A particular problem which we faced was regarding preprocessing large amounts of data. After preprocessing our data was reduced from 30000 to 28000. Preprocessing 28000 records of data was putting too much load on our processor and leading to the system being hanged. For this reason, we had to reduce our data set. When we reduced our data to 22000 our processor was able to deal with the data effectively.

Another particular problem which we faced was in the context of parsing information from a resume which we had initially planned to include in our application.

In the case of resumes which exist as printable pdf files information from the pdf files are is being transformed into unreadable characters. Another issue which arose was that many resume parsing techniques , which we came across on the internet , involved recognizing headings on resumes and then extracting information with respect to the headings(Education , Qualification, Skills , etc). The headings whose information is desired needs to be specified in code. The problem was that many resumes involved different headings and standard headings written in different words. For example in some resumes the Skills heading is written as “Technical skills”. In some “Professional skills”. In some “Skills, Tools, and Technologies”. In some “Education” is written as “Education”. In some “Education” is written as “Qualification”. This was creating information in extracting resumes in the sense that we were being forced to specify a lot of headings in hard code and we did not know how many and which headings should be specified.

## References

Guo, Q. (2019, April 22). The AI Behind LinkedIn Recruiter search and recommendation systems. LinkedIn. <https://engineering.linkedin.com/blog/2019/04/ai-behind-linkedin-recruiter-search-and-recommendation-systems>

COnnecting REpositories UK (CORE). (2016). Resume Ranking using NLP and ML. Anjuman-I Islam's Kalsekar Technical Campus, Navi Mumbai.  
<https://core.ac.uk/download/pdf/55305289.pdf>

Kulkarni, Y. (2017, May 3). yogeshhk/MiningResume. GitHub.  
<https://github.com/yogeshhk/MiningResume>

Kulkarni, Y. H. (2017, June). Text Mining 101: Mining Information From A Resume. KDnuggets. <https://www.kdnuggets.com/2017/05/text-mining-information-resume.html>

PromptCloud. (n.d.). Jobs on Naukri.com. Kaggle. Retrieved March 28, 2021, from  
<https://www.kaggle.com/promptcloud/jobs-on-naukricom>

Resume Parser to extract technical skills - Dipendrapant. (2020, June 28). Medium.  
<https://dipendrapant778.medium.com/resume-parser-to-extract-technical-skills-6a42b04cd037>

## A Glossary

### Abbreviation

UC

TC

API

NLP

AI

CV

### Terminologies

Use-case

Test-case

Application Programming Interface

Natural Language Processing

Artificial Intelligence

Curriculum Vitae

## **B    Deployment/Installation Guide**

The users can install this application on their Android device after its release.

## **C    User Manual**

The user should follow the following steps:

Job Seeker/ Team Seeker:

1. Install the application on your android phones.
2. Open the application.
3. Sign up on the application.
4. Click on Find Jobs/Find Teams.
5. Click on Job/Team.
6. Apply for Job/Team.

Employer:

1. Create Employer Account.
2. Fill in details for Employer profile.
3. Create Job.

Team Owner:

1. Create Team owner Account.
2. Fill in details for Team Owner profile.
3. Create Team.

## D Student Information Sheet

Roll No	Name	Email Address (FC College)	Frequently Checked Email Address	Personal Cell Phone Number
21-11102	Ahad Tariq	21-11102@formanite.fccollege.edu.pk	21-11102@formanite.fccollege.edu.pk	0324-8448344
21-10730	Muhammad Bilal	21-10730@formanite.fccollege.edu.pk	21-10730@formanite.fccollege.edu.pk	0332-4522255
21-10365	Razi Ul Hasan	21-10365@formanite.fccollege.edu.pk	21-10365@formanite.fccollege.edu.pk	0313-7827667

## E Plagiarism Free Certificate

This is to certify that, I am \_\_\_\_\_ S/D/o \_\_\_\_\_, group leader of FYP under registration no \_\_\_\_\_ at Computer Science Department, Forman Christian College (A Chartered University), Lahore. I declare that my final year project report is checked by my supervisor and the similarity index is \_\_\_\_\_% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix F. To the best of my knowledge and belief, the report contains no material previously published or written by another person except where due reference is made in the report itself.

Date: \_\_\_\_\_ Name of Group Leader: \_\_\_\_\_ Signature: \_\_\_\_\_

Name of Supervisor: \_\_\_\_\_ Co-Supervisor (if any): \_\_\_\_\_

Designation: \_\_\_\_\_ Designation: \_\_\_\_\_

Signature: \_\_\_\_\_ Signature: \_\_\_\_\_

Senior Project Management Committee Representative: \_\_\_\_\_

Signature: \_\_\_\_\_

## F Plagiarism Report