HYRE – HIGHER YOU REACH EVERY TIME

A Project Report

Submitted by

Mr. Prajwal M S	**20171CSE0519**
Mr. Prajwal P	**20171CSE0520**
Mr. Pramukh N S	**20171CSE0524**
Ms. Priyanka A	**20171CSE0536**
Ms. Rachna Shah	**20171CSE0551**

Under the guidance of

Mr. James Mathew P

in partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

Department of Computer Science and Engineering

School of Engineering

PRESIDENCY UNIVERSITY

BANGALORE

MAY 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING

PRESIDENCY UNIVERSITY

CERTIFICATE

This is to certified that the Project report **"HYRE – HIGHER YOU REACH EVERY TIME"** being submitted by "**Prajwal MS, Prajwal P, Pramukh NS, Priyanka A, Rachna Shah"** bearing roll number(s): **20171CSE0519, 20171CSE0520, 20171CSE0524, 20171CSE0536, 20171CSE0551** in partial fulfillment of requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide work carried out under my supervision.

Dr. C. Kalaiarasan
UP-II (In charge)
Associate Dean-Admin
Department of CSE
Presidency University

Mr. James Mathews P
Guide
Assistant Professor
Department of CSE
Presidency University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING

PRESIDENCY UNIVERSITY

DECLARATION

I hereby declare that the work, which is being presented in the project report entitled **HYRE – HIGHER YOU REACH EVERY TIME** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Mr. James Mathew P, Department of Computer Science and Engineering, School of Engineering, Presidency University, Bangalore.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Mr.Prajwal M S	20171CSE0519	Prajwal MS
Mr.Prajwal P	20171CSE0520	Prajwal-P
Mr.Pramukh NS	20171CSE0524	Pramukh. NS
Ms.Priyanka A	20171CSE0536	Priyanka.A
Ms.Rachna Shah	20171CSE0551	Rachna Shah

ABSTRACT

The Information Age has affected the workforce in several ways. Employees are also being forced to compete in a global job market. Technology has increased job opportunities in developing countries and has promoted the globalization of the workforce. HYRE is an application which acts as a bridge to connect Freelancers and Company. It is a one stop destination for both Freelancers and Company. Freelancers access this application to advertise themselves in a talent marketplace to gain temporary employment via a customized profile, which typically includes resumes, proficiencies and interests. Companies also access this application to browse profiles of freelancer candidates based on skills, experience and hire the freelancers for specific projects.

HYRE provides a portal for companies to post projects and for freelancers to create profiles and describe their skills and experience. This application enables freelancers and companies to collaborate directly and manage the workflows and digital assets involved with the role or project.

ACKNOWLEDGEMENT

First of all, we indebted to the GOD ALMIGHTY for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Abdul Sharief**, Dean, School of Engineering, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved professor **Dr. C. Kalaiarasan,** University Project-II In charge, Associate Dean-Admin, Department of Computer Science and Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide Mr**. James Mathews P, Assistant Professor** Department of Computer Science and Engineering, Presidency University for his/her inspirational guidance, valuable suggestions and providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We thank our friends for the strong support and inspiration they have provided us in bringing out this project.

Mr. Prajwal M S	20171CSE0519
Mr. Prajwal P	20171CSE0520
Mr. Pramukh N S	20171CSE0524
Ms. Priyanka A	20171CSE0536
Ms. Rachna Shah	20171CSE0551

List of Figures

Sl.No	Figure Name	Caption	Page No
1	Fig 1	Unemployment Rate	1
2	Fig 2.1.1	Android Studio	2
3	Fig 2.1.2	Android OS	3
4	Fig 2.1.3	Android Emulator	4
5	Fig 2.1.5	Firebase	5
6	Fig 3.1	Reference Paper	7
7	Fig 3.2	Reference Paper	8
8	Fig 4.1	Toptal	9
9	Fig 4.2	Upwork	10
10	Fig 4.3	Freelancer	10
11	Fig 6.1	System Architecture	12
12	Fig 6.2	Modules	13
13	Fig 6.4	Level 0 DFD	14
14	Fig 6.5	Level 1 Freelancer	15
15	Fig 6.5.1	Level 1 Employer	15
16	Fig 6.6	Level 2 Freelancer	16
17	Fig 6.6.1	Level 2 Employer	17
18	Fig 6.7	HYRE ER Diagram	17
19	Fig 6.8.1	Login Table Schema	18
20	Fig 6.8.2	ProfileTableSchema	18
21	Fig 6.8.3	Project Table Schema	19
22	Fig 7.1	Login Page	20
23	Fig 7.2	Signup Page	21
24	Fig 7.3	Database	21
25	Fig 7.4	Freelancer homepage	22
26	Fig 7.4.1	Company homepage	22
27	Fig 7.5	Freelancer Profile	23
28	Fig 7.5.1	Company Profile	23
29	Fig 7.6	New Project	24
30	Fig 7.7	FreelancerNotification	25
31	Fig 7.7.1	Company Notification	25
32	Fig 7.8	Hire Module	26
33	Fig 7.8.1	Connection Module	26
34	Fig 7.9	Ratings	27

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	DECLARATION	**i**
	ABSTRACT	**ii**
	ACKNOWLEDGEMENT	**iii**
CHAPTER 1	**INTRODUCTION**	1
CHAPTER 2	**REQUIREMENT ANALYSIS**	2-6
	2.1 SOFTWARE REQUIREMENTS	2-6
	2.1.1 ANDROID STUDIO 4.1	2
	2.1.2 ANDROID OS	2-3
	2.1.3 ANDROID EMULATOR	3-4
	2.1.4 HAXM 6.2.1	4
	2.1.5 FIREBASE	5
	2.1.5.1 FIREBASE AUTHENTCATION	5
	2.1.5.2 FIRESTORE DATABASE	6
	2.1.5.3 REALTIME DATABASE	6
	2.2 HARDWARE REQUIREMENTS	6
	2.2.1 4GB RAM OR HIGHER AND PROCESSOR	6
CHAPTER 3	**LITERATURE REVIEW**	7-8
CHAPTER 4	**EXISTING SYSTEM**	9-10
	4.1 TOPTAL	9
	4.2 UPWORK	10
	4.3 FREELANCER	10
CHAPTER 5	**PROPOSED SYSTEM**	11
CHAPTER 6	**SYSTEM DESIGN**	12-19
	6.1 HYRE SYSTEM ARCHITECTURE	12-13
	6.2 MODULES	13
	6.3 DATA FLOW DIAGRAM	13
	6.3.1 DFD COMPONENTS	13-14
	6.4 LEVEL 0 DFD	14

	6.5 LEVEL 1 DFD	15
	6.6 LEVEL 2 DFD	16-17
	6.7 HYRE ER DIAGRAM	17
	6.8 HYRE TABLE SCHEMA	18-19
CHAPTER 7	**IMPLEMENTATION**	20-27
CHAPTER 8	**TESTING**	28-31
	8.1 UNIT TESTING	28-30
	8.2 INTEGRATION TESTING	30-31
	8.3 SYSTEM TESTING	31
	CONCLUSION	32
	APPENDIX	33-60
	REFERENCES	61

1. INTRODUCTION

Applications are computer programs which have simplified human life to a great extent over a decade. In consideration for boon in application development, Our team has brainstormed an idea of helping the freelancer and company bridge their requirements through our application so called HYRE. Primarily, HYRE provides a platform for the organization find their particularly skilled employee, On the other hand the freelancer with their dream project which can help them enhance their profiles. This mutual benefit relaxes the stress of unemployment and recruitment significantly on both the ends which is our key motive.

Unemployment is caused by various reasons that come from both the demand side, or employer, and the supply side, or the worker.

Demand-side reductions may be caused by high interest rates, global recession, and financial crisis. From the supply side, frictional unemployment and structural employment play a great role India's unemployment rate (UER) for the month of November 2020 stood at 6.51%, the lowest since September 2018 when it was 6.47%.According to the Centre for Monitoring Indian Economy (CMIE) data. While the urban unemployment rate stood at 7.07%, rural UER was 6.26%.

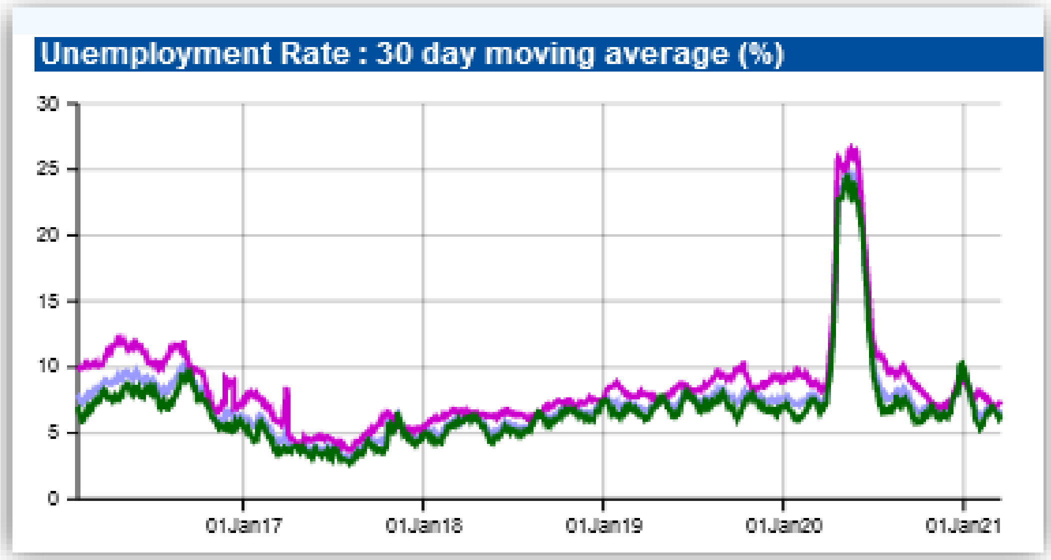


Fig 1: Unemployment Rate

2. REQUIREMENT ANALYSIS

The requirement specification for the application "HYRE – HIGHER YOU REACH EVERY TIME" states a basic document that constitutes the foundation of the development process.

SOFTWARE REQUIREMENTS

ANDROID STUDIO 4.1

Android Studio is the official integrated development environment (IDE) for Google's Android operating system. It is purpose-built for Android to accelerate your development and helps to build the highest-quality apps for every Android device.
It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service.

Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager. You can start with a code template or even right-click an API in the editor and select Find Sample Code to search for examples. Moreover, you can import fully functional apps from GitHub, right from the Create Project screen.

Fig 2.1.1: Android Studio

ANDROID OS

This OS is a Linux-based operating system. It is designed primarily for Android mobile devices such as smart phone and tablet computers.

Android OS is a powerful operating system and it supports a large number of applications on Smart phones. These applications are more comfortable and advanced for users. The hardware that supports android software is based on the ARM architecture platform.

The android is an open-source operating system. Android has got millions of apps available that can help you manage your life one or another way and it is available at low cost in the market for that reason android is very popular.
Jelly bean and above OS versions will work in Android Studio.

Fig 2.1.2: Android OS

ANDROID EMULATOR

The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

Testing your app on the emulator is in some ways faster and easier than doing so on a physical device.

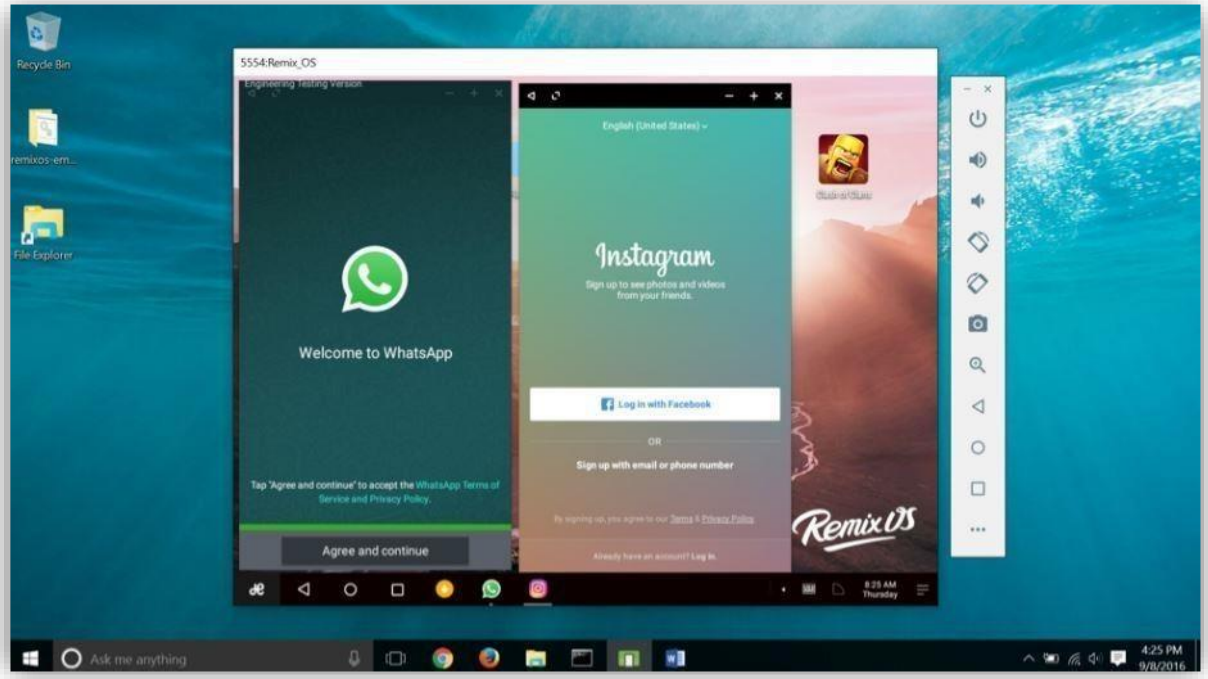


Fig 2.1.3: Android Emulator

2.1.4 HAXM 6.2.1

HAXM is a cross-platform hardware-assisted virtualization engine (hypervisor), widely used as an accelerator for Android Emulator and QEMU.

It has always supported running on Windows and macOS, and has been ported to other host operating systems as well, such as Linux and NetBSD.

HAXM runs as a kernel-mode driver on the host operating system, and provides a KVM-like interface to user space, thereby enabling applications like QEMU to utilize the hardware virtualization capabilities built into modern Intel CPUs, namely Intel Virtualization Technology

FIREBASE

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.

The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server.

For our project we have used Firebase authentication, Firestore database, Realtime database and Storage.

We have used the above mentioned features by implementing their respective dependencies.

Fig 2.1.5: Firebase

FIREBASE AUTHENTICATION

Firebase authentication supports authentication using a password, phone numbers, Google sign in and Facebook login. It also provides some user interface libraries which enable screens for us when we are logging it. Firebase Authentication provides all the server-side stuff for authenticating the user.

FIRESTORE DATABASE

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.

REALTIME DATABASE

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receive that update within milliseconds.

HARDWARE REQUIREMENTS

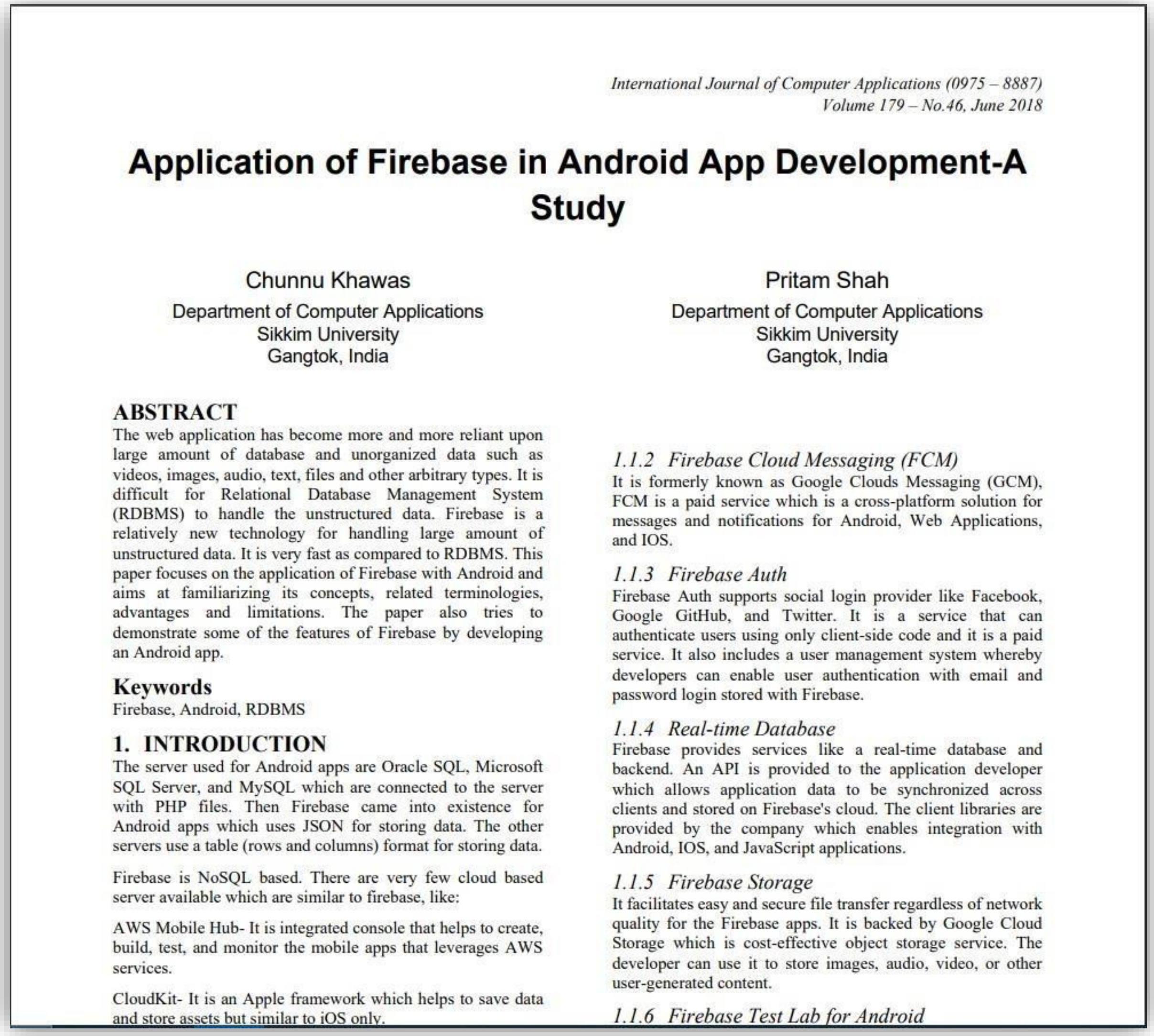
4GB RAM OR HIGHER AND PROCESSOR

Android studio is a very large IDE which needs lot of memory at a time to make it run smooth. So a minimum of 4GB of RAM is required for it to run. But 8GB Ram and higher processor is recommended to make Android Studio run faster and smooth.

3. LITERATURE REVIEW

An objective oriented subject study and existing system analysis plays a significant role for rooting foundation of any research and system optimization schemes. Thus taking into consideration of app development literature review we have conducted a review that discusses various challenges and best practices for app development process.

“Application of Firebase in Android App Development – A Study” by Chunnu Khawas and Pritam Shah is an international journal paper about firebase with android and aims at concepts, related terminologies advantages and limitations. It also specifies some features of firebase by developing an Android App



International Journal of Computer Applications (0975 – 8887)
Volume 179 – No.46, June 2018

Application of Firebase in Android App Development-A Study

Chunnu Khawas
Department of Computer Applications
Sikkim University
Gangtok, India

Pritam Shah
Department of Computer Applications
Sikkim University
Gangtok, India

ABSTRACT

The web application has become more and more reliant upon large amount of database and unorganized data such as videos, images, audio, text, files and other arbitrary types. It is difficult for Relational Database Management System (RDBMS) to handle the unstructured data. Firebase is a relatively new technology for handling large amount of unstructured data. It is very fast as compared to RDBMS. This paper focuses on the application of Firebase with Android and aims at familiarizing its concepts, related terminologies, advantages and limitations. The paper also tries to demonstrate some of the features of Firebase by developing an Android app.

Keywords

Firebase, Android, RDBMS

1. INTRODUCTION

The server used for Android apps are Oracle SQL, Microsoft SQL Server, and MySQL which are connected to the server with PHP files. Then Firebase came into existence for Android apps which uses JSON for storing data. The other servers use a table (rows and columns) format for storing data.

Firebase is NoSQL based. There are very few cloud based server available which are similar to firebase, like:

AWS Mobile Hub- It is integrated console that helps to create, build, test, and monitor the mobile apps that leverages AWS services.

CloudKit- It is an Apple framework which helps to save data and store assets but similar to iOS only.

1.1.2 Firebase Cloud Messaging (FCM)

It is formerly known as Google Clouds Messaging (GCM), FCM is a paid service which is a cross-platform solution for messages and notifications for Android, Web Applications, and IOS.

1.1.3 Firebase Auth

Firebase Auth supports social login provider like Facebook, Google GitHub, and Twitter. It is a service that can authenticate users using only client-side code and it is a paid service. It also includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

1.1.4 Real-time Database

Firebase provides services like a real-time database and backend. An API is provided to the application developer which allows application data to be synchronized across clients and stored on Firebase's cloud. The client libraries are provided by the company which enables integration with Android, IOS, and JavaScript applications.

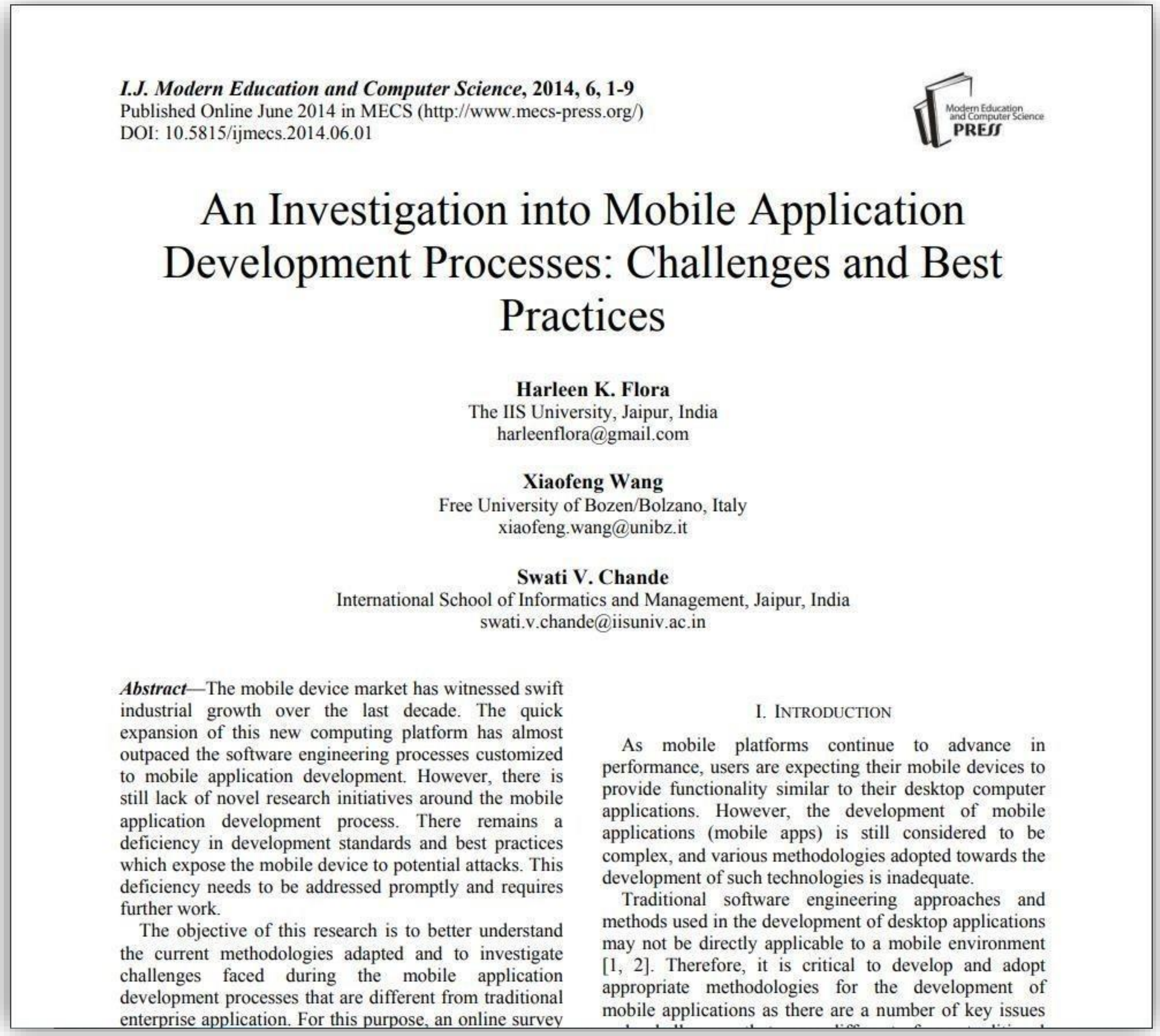
1.1.5 Firebase Storage

It facilitates easy and secure file transfer regardless of network quality for the Firebase apps. It is backed by Google Cloud Storage which is cost-effective object storage service. The developer can use it to store images, audio, video, or other user-generated content.

1.1.6 Firebase Test Lab for Android

Fig 3.1: References Papers

"An Investigation into Mobile Application Development Processes: Challenges and Best Practices" by Harleen K. Flora and Xiaofeng Wang is an journal paper about understanding the current methodologies adapted and to investigate challenges faced during mobile application development process.



I.J. Modern Education and Computer Science, **2014, 6, 1-9**
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.06.01

An Investigation into Mobile Application Development Processes: Challenges and Best Practices

Harleen K. Flora
The IIS University, Jaipur, India
harleenflora@gmail.com

Xiaofeng Wang
Free University of Bozen/Bolzano, Italy
xiaofeng.wang@unibz.it

Swati V. Chande
International School of Informatics and Management, Jaipur, India
swati.v.chande@iisuniv.ac.in

Abstract—The mobile device market has witnessed swift industrial growth over the last decade. The quick expansion of this new computing platform has almost outpaced the software engineering processes customized to mobile application development. However, there is still lack of novel research initiatives around the mobile application development process. There remains a deficiency in development standards and best practices which expose the mobile device to potential attacks. This deficiency needs to be addressed promptly and requires further work.

The objective of this research is to better understand the current methodologies adapted and to investigate challenges faced during the mobile application development processes that are different from traditional enterprise application. For this purpose, an online survey

I. INTRODUCTION

As mobile platforms continue to advance in performance, users are expecting their mobile devices to provide functionality similar to their desktop computer applications. However, the development of mobile applications (mobile apps) is still considered to be complex, and various methodologies adopted towards the development of such technologies is inadequate.

Traditional software engineering approaches and methods used in the development of desktop applications may not be directly applicable to a mobile environment [1, 2]. Therefore, it is critical to develop and adopt appropriate methodologies for the development of mobile applications as there are a number of key issues

Fig 3.2: – Reference Paper

We also referred to many papers and articles.

4. EXISTING SYSTEM

Freelancing has become a very popular practice among our modern workforce.
Freelance work has grown immensely over the past decade and is expected to further increase in the coming years. With the increase of freelance platforms, the competition is tougher than ever.

Freelancing platforms help companies find and hire independent professionals for temporary job roles or special projects. They offer a marketplace for businesses and freelancer workers from all over the world. These platforms allow business to browse profiles of freelancer candidates based on skills, experience, location, or other criteria. Alternatively, companies can also post a project description and solicit proposals from freelancers. These capabilities help companies of all sizes outsource work on projects that require specialized skills or additional manpower, freeing up full-time employees to focus on other business activities.
There are few existing freelancer applications

TOPTAL

Toptal is a global remote company that provides a freelancing platform connecting businesses with software engineers, designers, and business consultants.
This is an application for an experienced, talented software engineer looking to work with top clients.

Fig 4.1: Toptal

UPWORK

Upwork is an employment website that was founded in 2015. It was developed in response to the rising need for remote work opportunities. There are millions of remote freelance jobs posted on Upwork each year.

This platform has been used by 9million freelancers.

Fig 4.2: Upwork

FREELANCER

Freelancer.com is an Australian freelance marketplace website, which allows provincial employees to post jobs that freelancers can bid to complete.

This platform is widely used in India and it allows freelancers to compete with other freelancer to win notoriety and engagements.

Fig 4.3: Freelancer.com

5. PROPOSED SYSTEM

The existing freelancers applications witnessed above illustrates varies features and uniqueness. So is our application HYRE , this platform has its newness not only bound to freelancers but also the company who will be seeing their upcoming employees in the freelancer signed up to HYRE.

Discussing How HYRE is Atypical:

- **Bidding is not an option**

  Freelancers are known for bidding per piece of their work which is holds good sometimes. However, any client will tend to handover the project to the freelancer who quotes less. This leads less work for the good skilled freelancers. Bidding is not an option.

- **Appreciative**

  Secondly while our team was taking up a small survey from the freelancer they were been asked what are the features they would like to enhance from the existing app, we obtained the answer to be: Profile and work must be recognized of the freelancer who is working as an employee. Furthermore must get a better reach. We are trying to get it better in our app.

- **Experience paves the way**

  Based on the work of the freelancers they are being awarded tags rankings so that the company can choose among the best freelancer for their work.

- **Skills are Pivotal**

  Last but not the least our application is skill oriented and not money. Recognizing the skills and appreciating the work is considered to be of the highest priority.

6. SYSTEM DESIGN

HYRE SYSTEM ARCHITECTURE

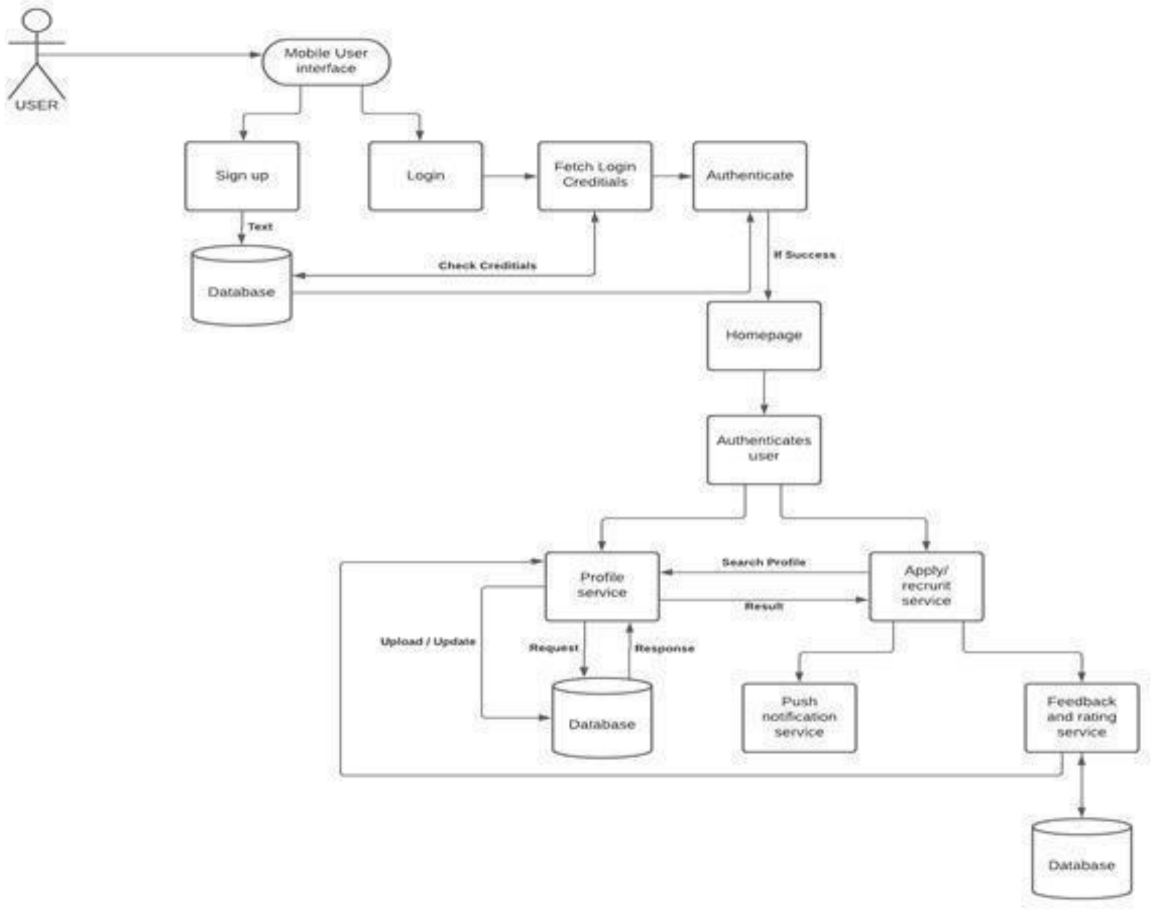


Fig 6.1: System Architecture

System architecture explains the overview and the work flow of the system.

The users of our application are the freelancer and the company. This is a generic architecture which explains the work flow for both freelancer and the company.

The user sees is the mobile interface which is responsible for interaction between the user and the application. The new user has to sign up to the application. The details will be stored in the database. This database has a table with different columns like name contact info username password type of user, that is whether he is a freelancer or the company. If the user is an existing user, he can login directly and the credentials will be fetched from the database and then authenticated. If success, the user can now see the homepage. Different type of services is available for the user, like profile services, apply / recruit services. A user can create a new profile or he can update his existing profile. This will be reflected in the database. If the user is a freelancer, he can search for a company and apply based on his skills and if the user is a company he can recruit the freelancers based on the skills they require. When the user applies or recruits he will be offered services like the push notification services and the feedback and rating service, which will the reflected on the user profile and will be saved in the database.

MODULES

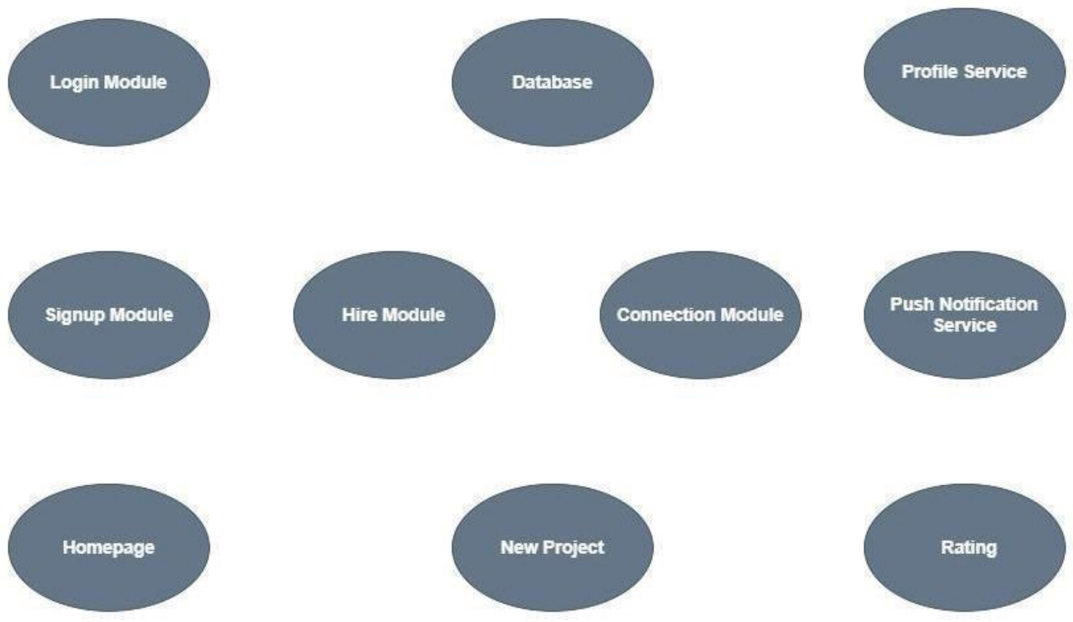


Fig 6.2: Modules

DATA FLOW DIAGRAMS

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

DFD COMPONENTS

- **Process**

  Input to output transformation in a system takes place because of process function.

- **Dataflow**

  Data flow describes the information transferring between different paths of the systems. The arrow symbol is the symbol of the data flow.

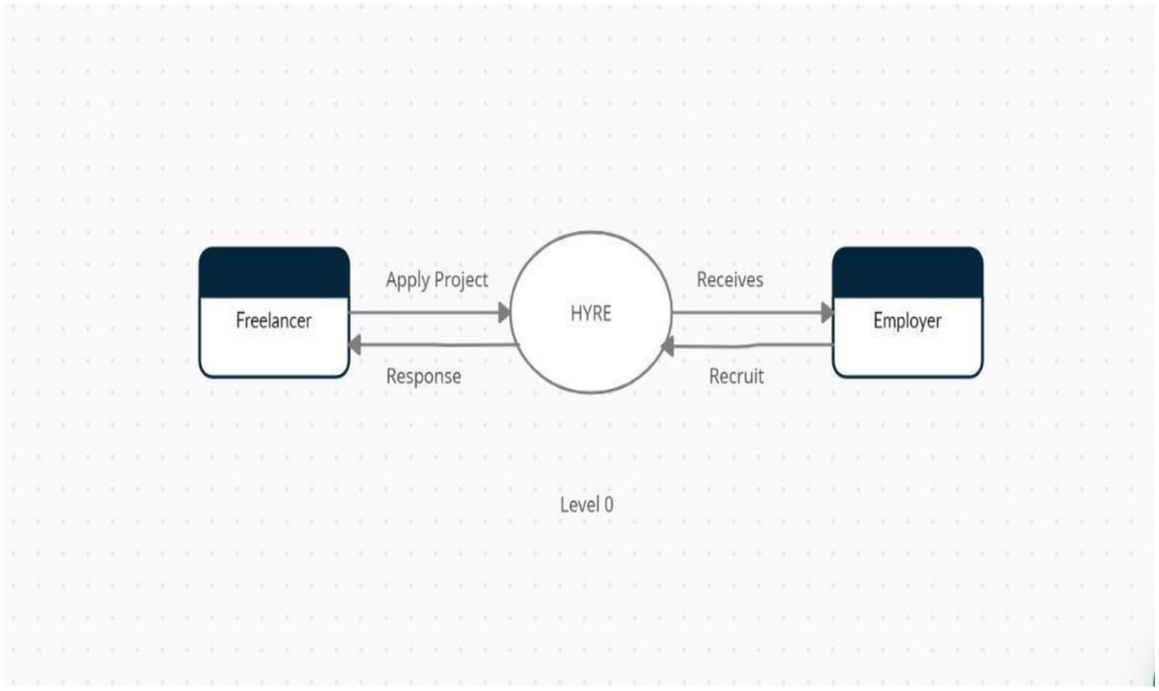
- **Warehouse**

  The data stored in the warehouse for later use. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data update.

- Terminator

  The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of peoples like customers.

LEVEL 0 DFD



. Fig 6.4: Level 0

LEVEL 1 DFD

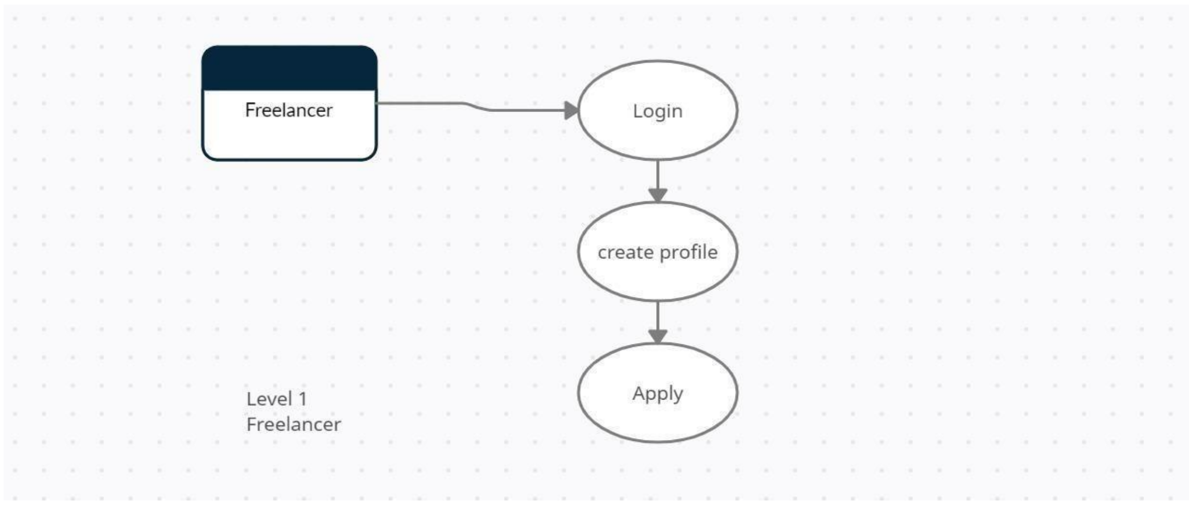


Fig 6.5: Level 1 Freelancer

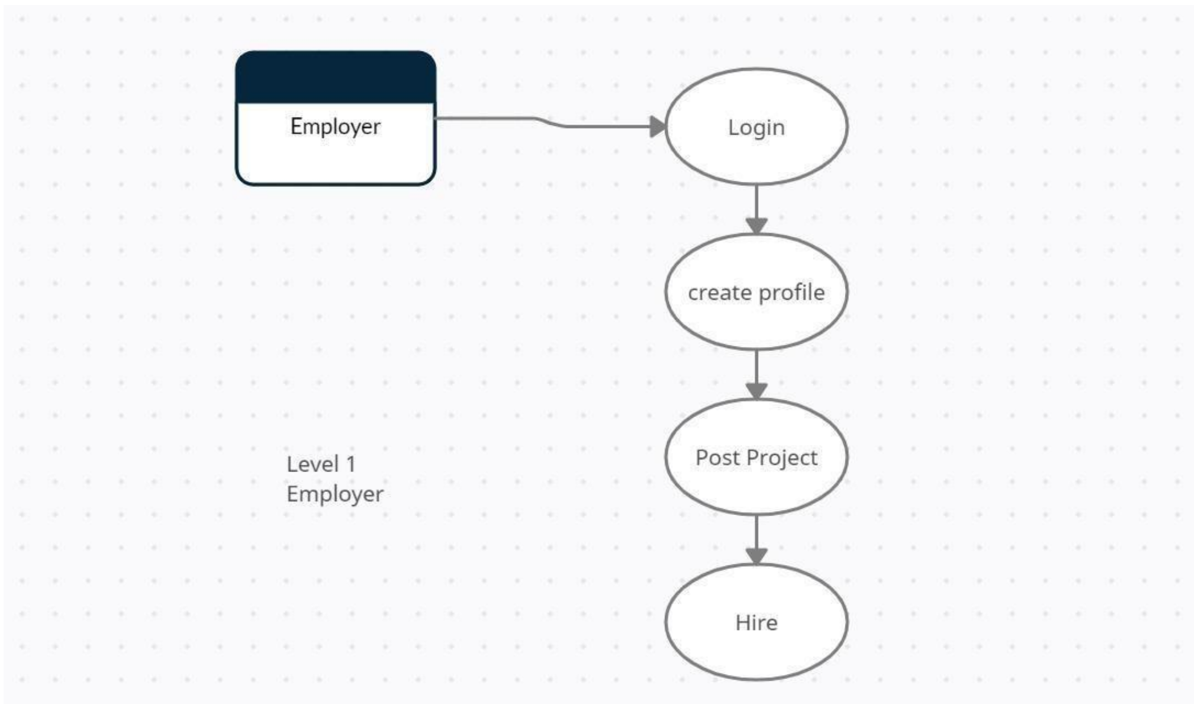


Fig 6.5.1: Level 1 Employer

LEVEL 2 DFD

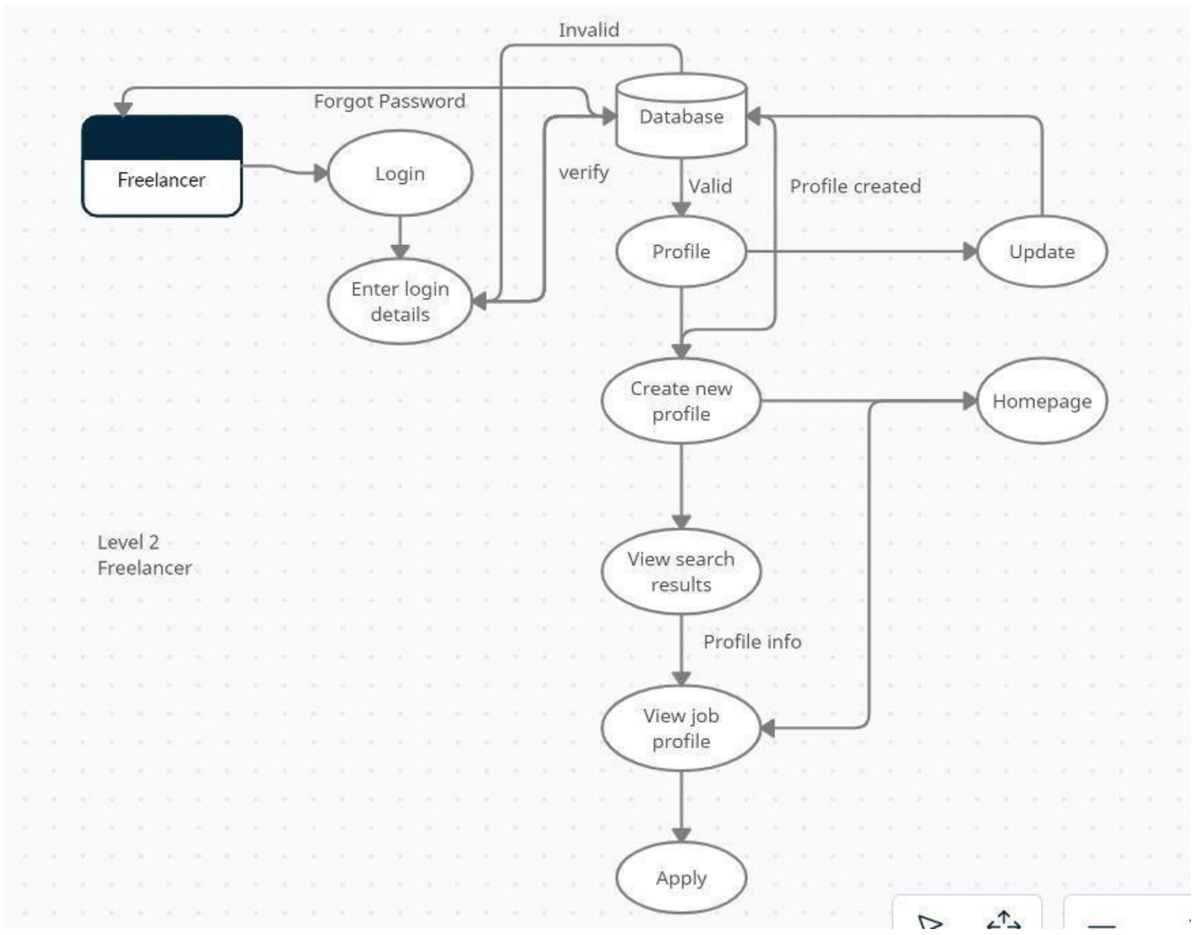


Fig 6.6: Level 2 Freelancer

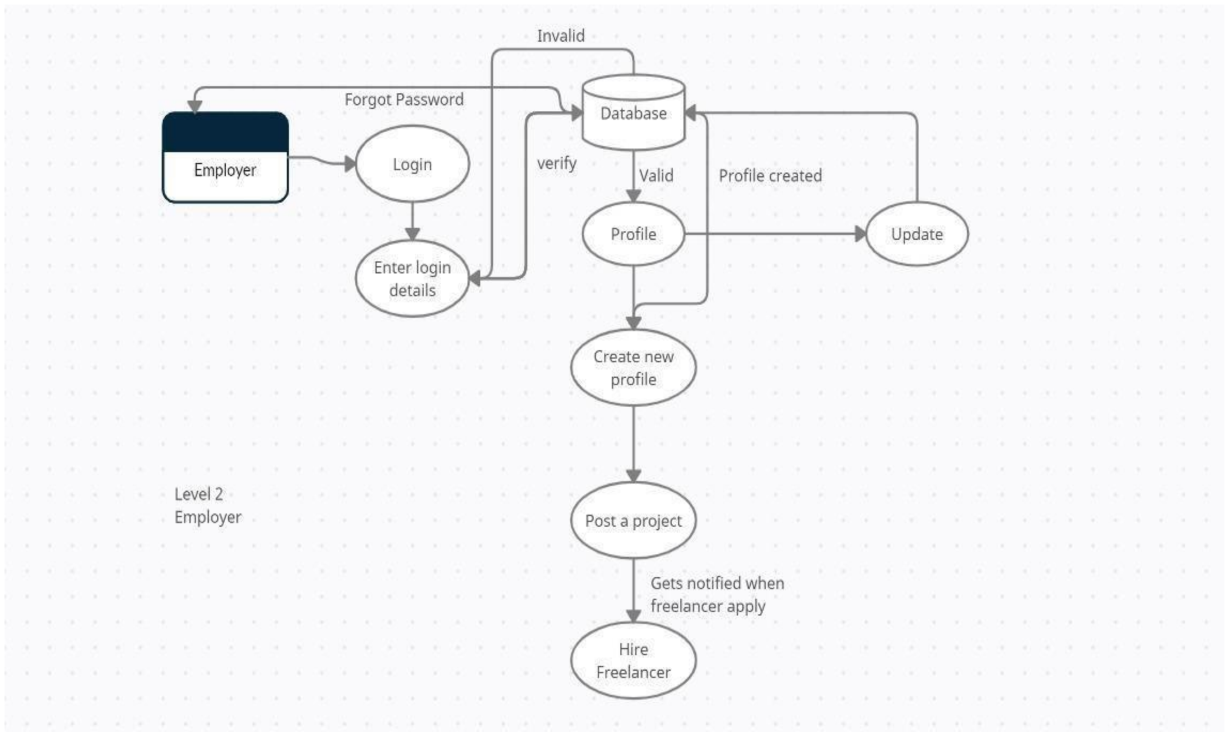


Fig 6.6.1: Level 2 Employer

HYRE DATABASE ER DIAGRAM

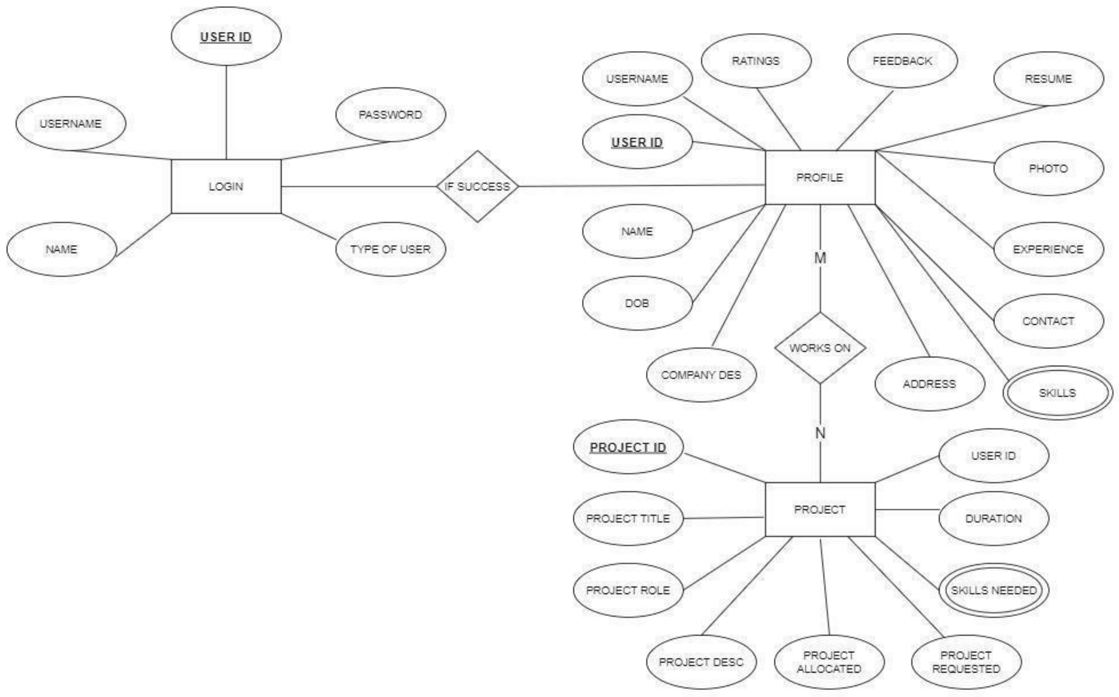
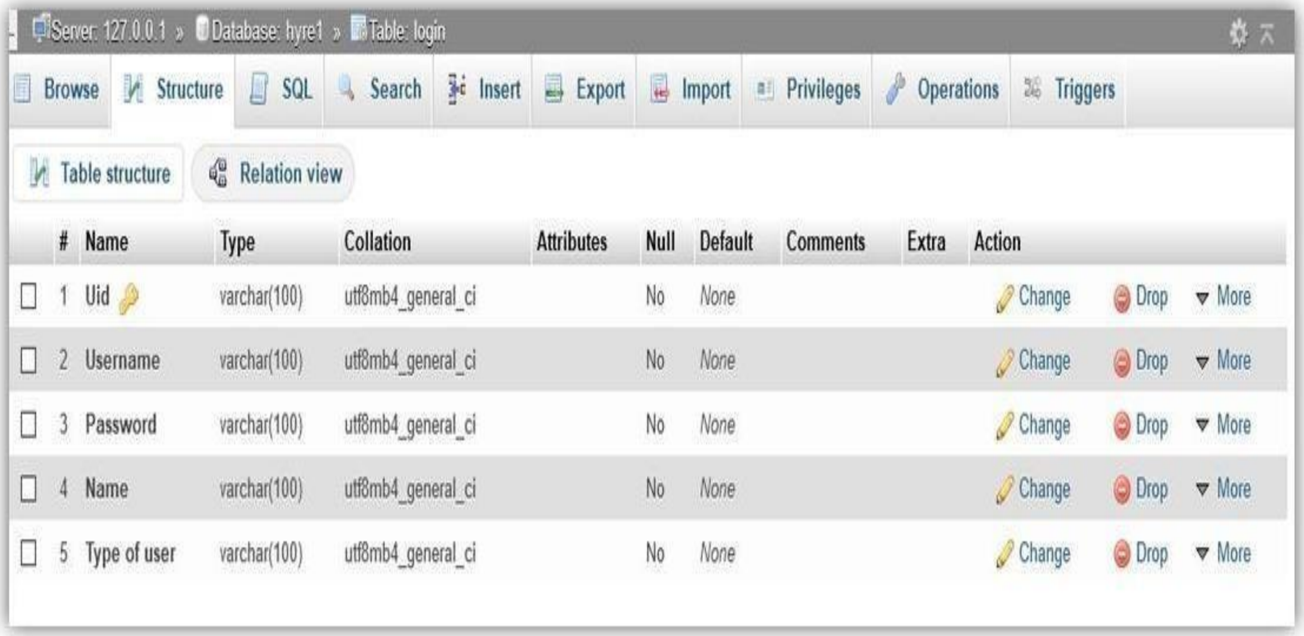


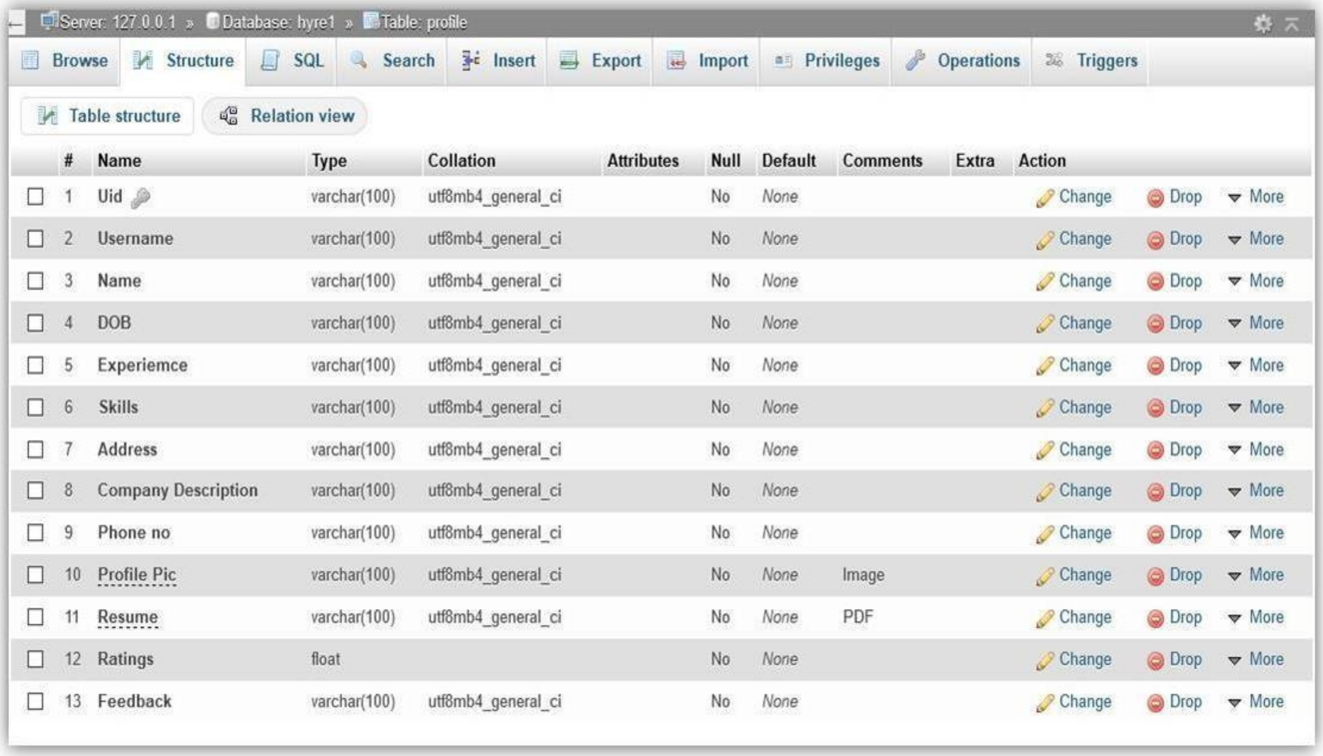
Fig 6.7: ER Diagram

HYRE TABLE SCHEMA


Server: 127.0.0.1 » Database: hyre1 » Table: login
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Table structure Relation view

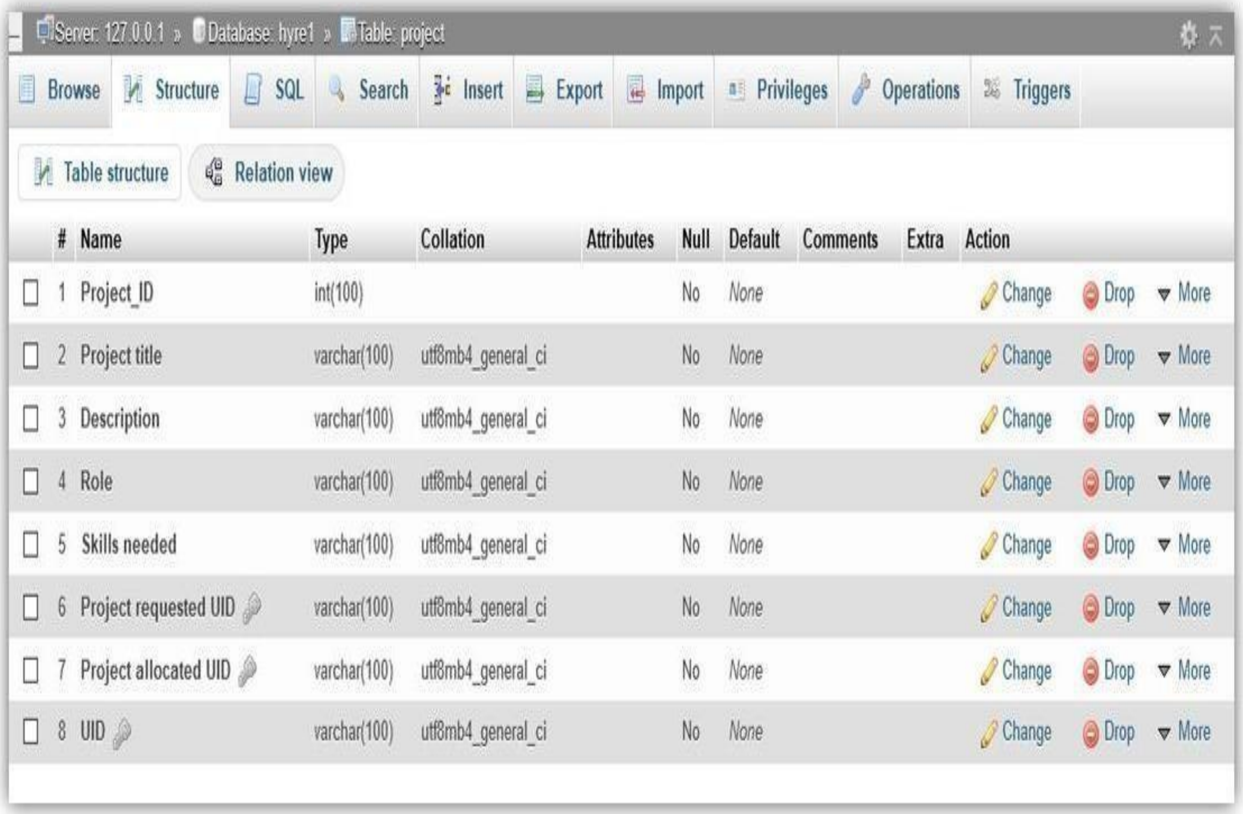
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Uid	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
2	Username	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	Password	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	Name	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	Type of user	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More

Fig 6.8.1: Login Table Schema


Server: 127.0.0.1 » Database: hyre1 » Table: profile
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Uid	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
2	Username	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	Name	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	DOB	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	Experiemce	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
6	Skills	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
7	Address	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
8	Company Description	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
9	Phone no	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
10	Profile Pic	varchar(100)	utf8mb4_general_ci		No	None	Image		Change Drop More
11	Resume	varchar(100)	utf8mb4_general_ci		No	None	PDF		Change Drop More
12	Ratings	float			No	None			Change Drop More
13	Feedback	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More

Fig 6.8.2: Profile Table Schema



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Project_ID	int(100)			No	None			Change Drop More
2	Project title	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	Description	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	Role	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	Skills needed	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
6	Project requested UID	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
7	Project allocated UID	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
8	UID	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More

Fig 6.8.3: Project Table Schema

7. IMPLEMENTATION

The implementation phase is the where visions and plans become reality. We have implemented our HYRE application using a platform called Android Studio and coded in Java language.

Application: HYRE – Higher You Reach Every time is an android application where a company can hire freelancers on project basis from our application which serve the purpose for both the company and freelancer. Once the project is completed the company also no longer has to pay the freelancer as they pay for their permanent employee and if they are satisfied with the work of freelancer, they can further hire them as an employee for their company.

Our application consists of certain modules with their functionalities and they are listed below

Login: This is a module that allows users to gain access to an application by entering their credentials. If authentication is successful, then user is directed to the homepage. If authentication fails, user remains on login page and an error message is displayed about the failure.

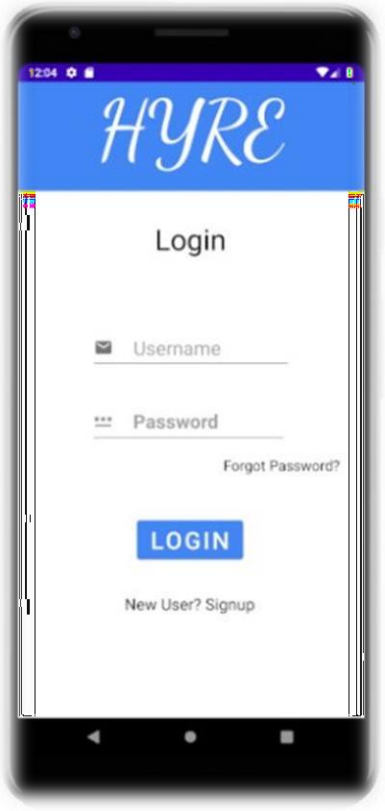


Fig 7.1: Login Page

Signup: This is a module that enables users to independently register, select their role and gain access to our application based on their role.

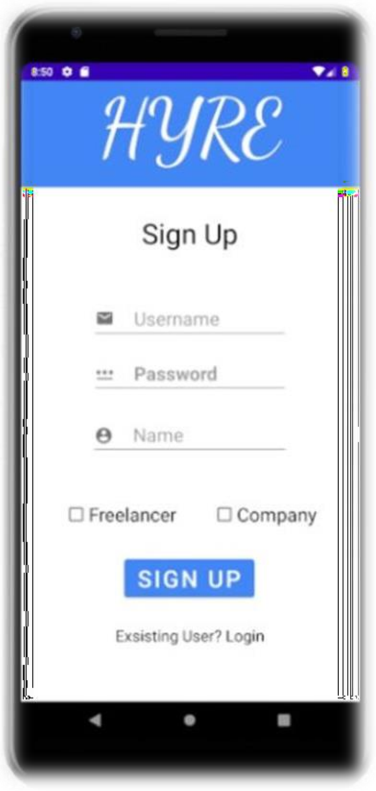
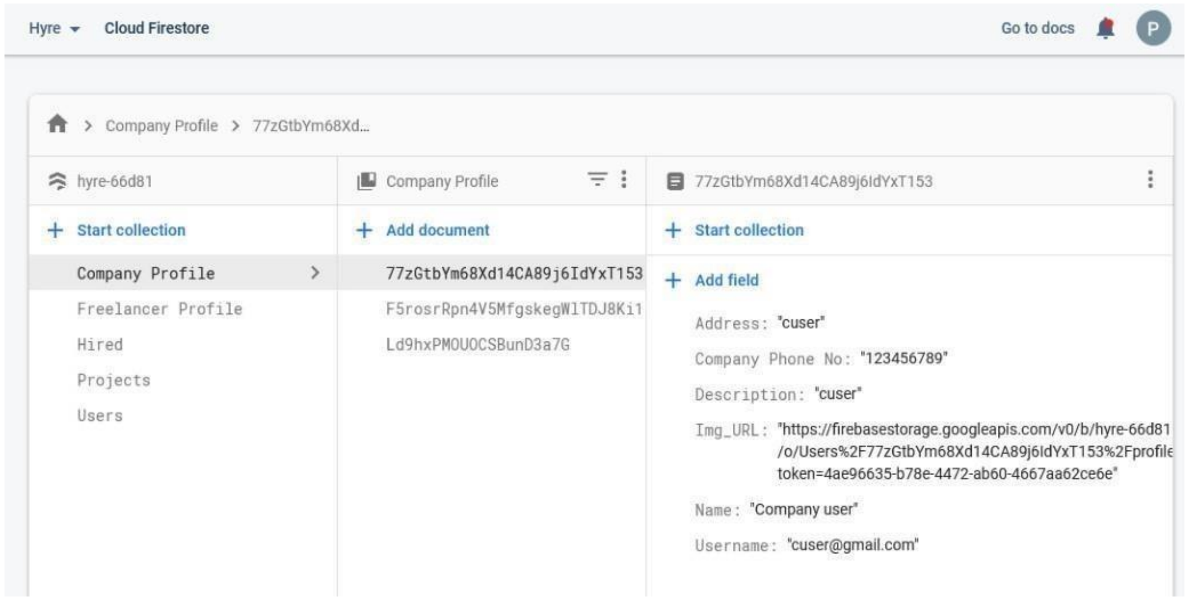


Fig 7.2: Signup Page

Database: It is a collection of user's data and its primary purpose to retrieve the requested information to the user.



```
Address: "cuser"
Company Phone No: "123456789"
Description: "cuser"
Img_URL: "https://firebasestorage.googleapis.com/v0/b/hyre-66d81
         /o/Users%2F77zGtbYm68Xd14CA89j6IdYxT153%2Fprofile
         token=4ae96635-b78e-4472-ab60-4667aa62ce6e"
Name: "Company user"
Username: "cuser@gmail.com"
```

Fig 7.3: Database

Homepage: This is a module that serves as the main page of our application and user's can find their respective information.

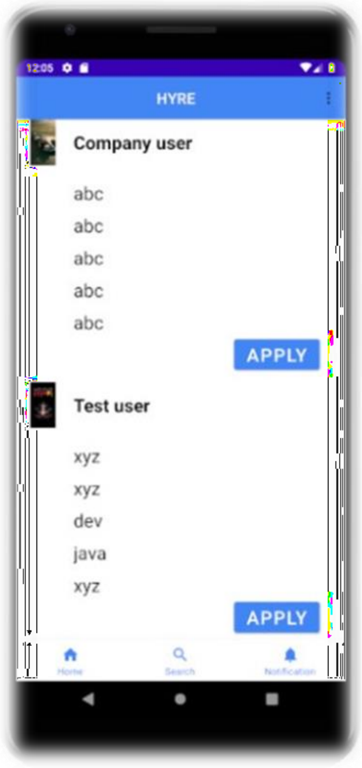


Fig 7.4: Freelancer Homepage

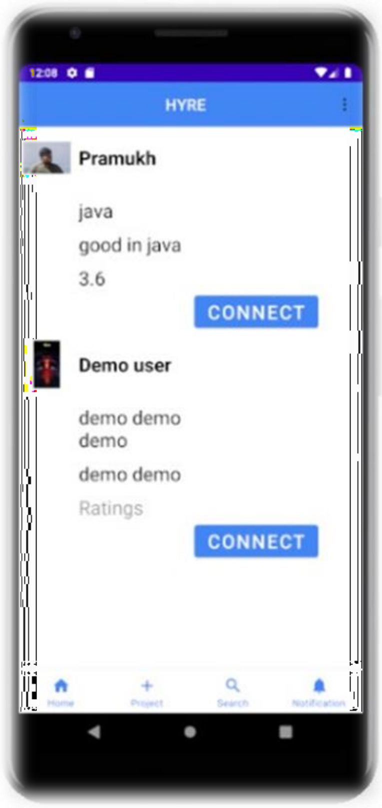
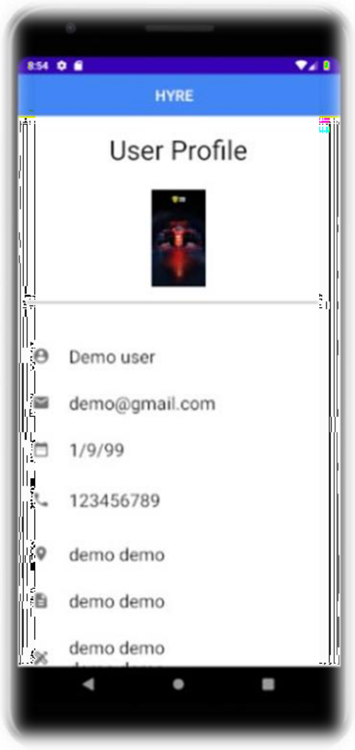


Fig 7.4.1: Company Homepage

User Profile Module: It is a collection of information which is used to identify an individual, such as their name, username, DOB, address, portrait photograph and individual characteristics such as skills and experience.



8:54

HYRE

User Profile

Demo user

demo@gmail.com

1/9/99

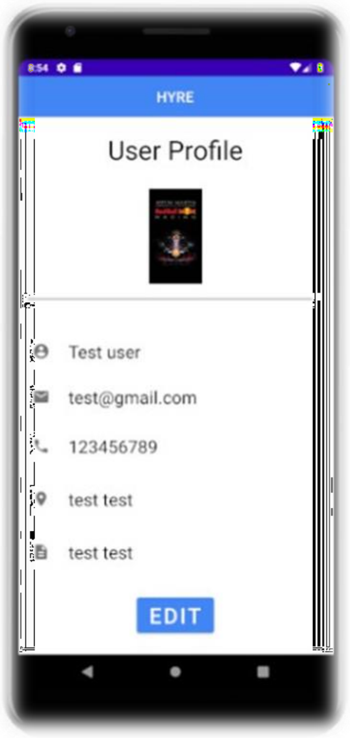
123456789

demo demo

demo demo

demo demo

Fig 7.5: Freelancer Profile



8:54

HYRE

User Profile

Test user

test@gmail.com

123456789

test test

test test

Fig 7.5.1: Company Profile

New Project Module: This module allows the company to add new projects into our application, so freelancer can apply for a project.

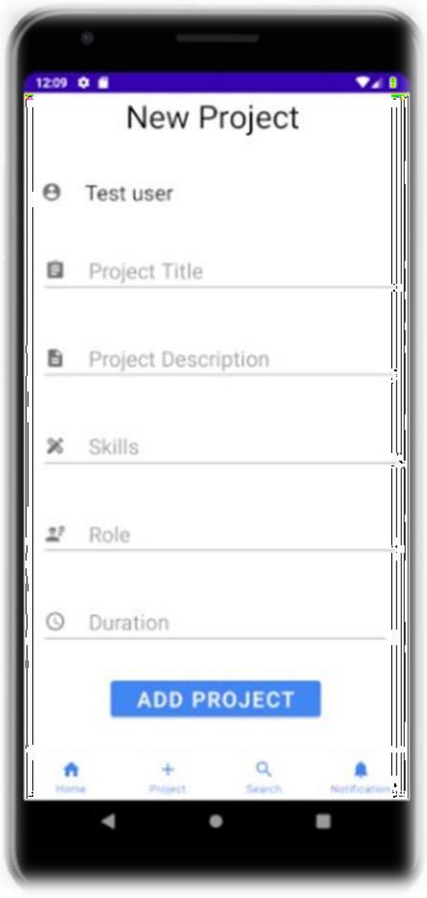
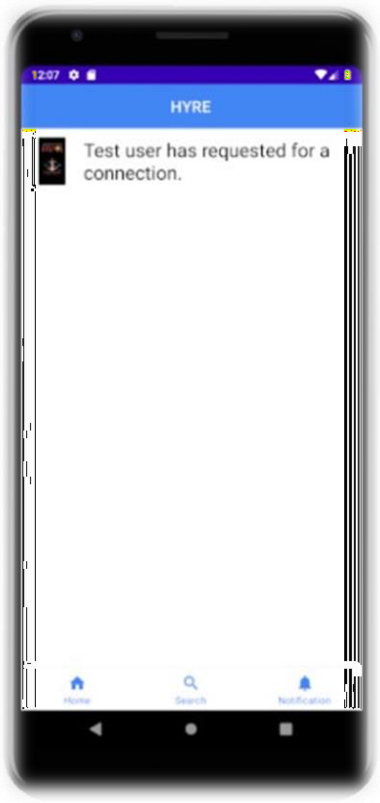


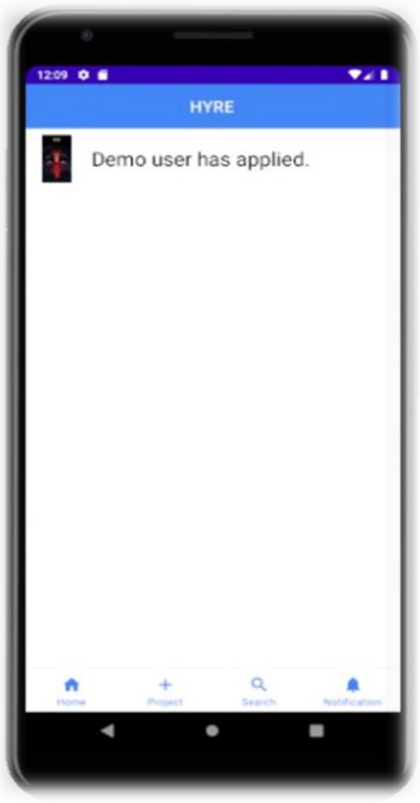
Fig 7.6: New Project

Push Notification: This is a module which allows the user to get notified when particular users apply/recruit for a project.



Test user has requested for a connection.

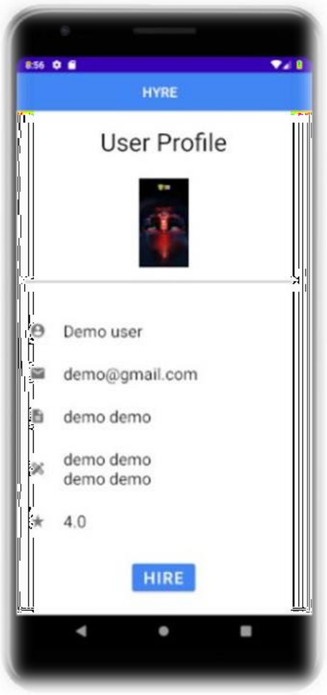
Fig 7.7: Freelancer Notification



Demo user has applied.

Fig 7.7.1: Company Notification

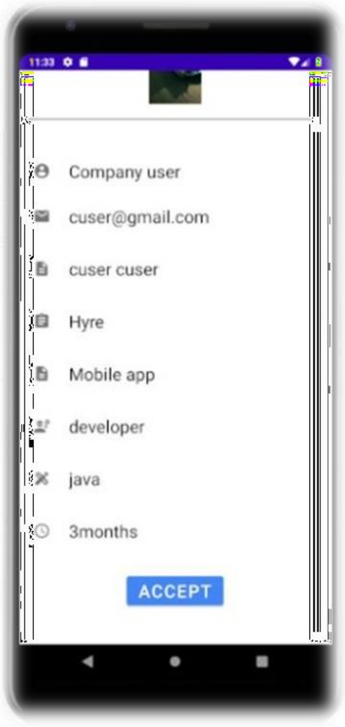
Hire: If a freelancer applies for a particular project, company gets a notification and hire freelancer for a project



Demo user
demo@gmail.com
demo demo
demo demo
demo demo
4.0

Fig 7.8: Hire module

Connection module: This module enables freelancer to accept connection from company



Company user
cuser@gmail.com
cuser cuser
Hyre
Mobile app
developer
java
3months

Fig 7.8.1: Connection Module

Ratings: This is a module which allows the company to provide ratings to the freelancer based on his work.

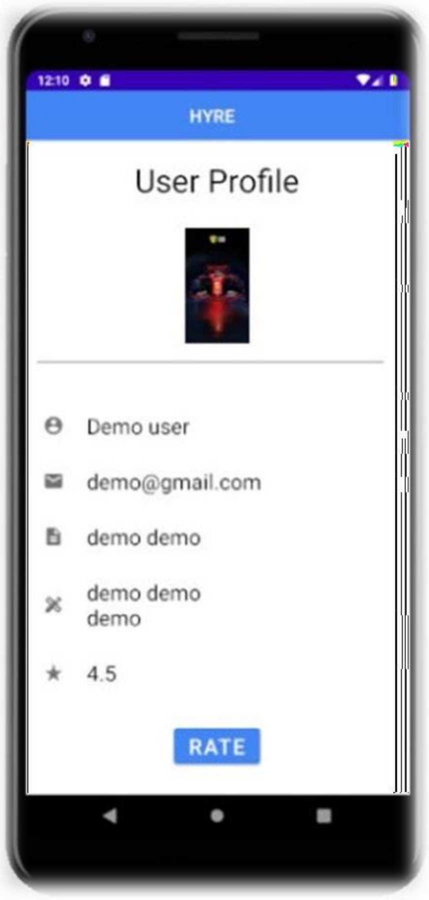


Fig 7.9: Ratings

8. TESTING

Testing is an important phase in the development life cycle. Testing provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding failures and verifying that the software product is fit for use.

During the testing, the program to be tested was performed with certain defined test scenarios and the output of the program for the test cases was evaluated to determine whether the program is performing as expected.
Therefore, testing exhibits a very significant responsibility for quality assurance and ensuring the reliability of the software.
Thus a series of testing was performed on the system before it was ready for implementation.

UNIT TESTING

Unit testing is a type of software testing where individual components of application are tested. A unit may be an individual function, method, procedure, module, or object.
Unit testing is important because entire system will only be able to work well if the individual parts are working well.
The purpose is to validate the developed interfaces of individual module were tested to ensure the code performs as expected.

Unit testing has been applied to each module in the application to ensure the module performs as expected.

Test Scenario for Login module

Verified if user will be able to login with a valid username and password and directed to homepage, functionality of "Forgot Password" and if user can login with invalid username and password.

Test Scenario for Signup module

Verified if user can enter details in input fields, user can signup into application successfully with all details, user credentials are reflecting in database, and “Type of User” functionality.

Test Scenario for Homepage module

Verified if homepage shows the respective information based on type of user after logged in to the application, appearance of Home, Search, Notification, Project icons and navigation from homepage to profile page

Test Scenario for New Project module

Verified if company can enter details in input fields, functionality of “ADD PROJECT” and Project details being able to get stored in database

Test Scenario for Database

Verified if user details like personal details, project details, login and signup details are saved in the database and connectivity between application and database.

Test Scenario for Profile Service

Verified if user can enter details in input fields, update his details and details are saved in the database.

Test Scenario for Push Notification

Verified if user gets notified when user apply/recruit for a project, displaying respective messages in this module like Test user has applied and Test user has requested for a connection.

Test Scenario for Rating

Verified if company able to give ratings to freelancer, appearance of freelancer profile with details and functionality of “RATE”.

Test Scenario for Hire module

Verified if company can hire the freelancer, appearance of freelancer profile with details and functionality of “HIRE”.

Test Scenario for Connection module

Verified if user can accept the connection received from company, appearance of company profile and functionality of "ACCEPT".

In this testing all modules performed well and produced expected results.

INTEGRATION TESTING

Integration testing is a type of software testing where individual components are combined and tested as a group.

The purpose of this testing is to assess the behaviour and functionality of both the modules after integration.

Integration testing has been applied to integrated modules in the application to ensure the both modules perform as expected.

Test Scenario for Login/Signup and Database

Verified if user able to successfully login/ signup in the application and user credentials are saved in the database.

Test Scenario for Login and Homepage

Verified if user able to successfully logged into application, navigation to homepage and appearance of respective information in homepage.

Test Scenario for Profile and Database

Verified if user profile details are saved and updated in the database.

Test Scenario for New Project and Database

Verified if project details are saved in the database.

Test Scenario for Apply and Push Notification

Verified if company notified through notification and appearance of text message in notification module when freelancer applies for a project.

Test Scenario for Hire and Push Notification

Verified if freelancer notified through notification and appearance of text message in notification module when company hires freelancer for a particular project.

Test Scenario for Connection and Push Notification

Verified if freelancer gets notified through notification when company request connection and appearance of company profile.

Test Scenario for Rating and Freelancer Profile

Verified if rating field in freelancer profile page is updated when company give ratings to freelancer based on his work.

In this testing all modules performed well and produced expected results.

SYSTEM TESTING

System testing is a type of software testing which is performed on a fully developed application.

System testing is important and it plays a significant role in delivering a quality product to the end user.

System testing has been applied on fully developed application to ensure application performs as expected.

Verified if user is able to login/signup into application, create user profile page, database connection, posting project details, Functionality of Apply, Hire, Accept, Rate and Notification.

In this testing all modules performed well and produced expected results.

CONCLUSION

HYRE provides a platform where both the entities- Freelancer and Company can get their task fulfilled. A Freelancer can work on projects that suit his skills and domain. Company can find skilled freelancers that are willing to work on their projects. This binary relationship between the two entities is an important factor in our app. The features in HYRE app such as hiring, rating, to name a few distinguish our platform from existing platforms. HYRE provides an opportunity to every individual who possess true skills to put his skills into work and help other set of users to get their job done. As a team we are proud to have worked on a project like HYRE which has a potential to create significant impact on the society by catering as a valuable platformto both Freelancers and Company.

APPENDIX

Java code of Login module

```
 public class MainActivity extends AppCompatActivity {
 DatabaseHelper hdb;
 EditText et3,et4,et5;
 Button signup;
 TextView tvlogin;
 CheckBox c1,c2;
 private FirebaseAuth firebaseAuth;
private FirebaseFirestore fstore;
@Override
protected void onCreate(Bundle savedInstanceState) {
Intent intent=getIntent();
super.onCreate(savedInstanceState);
et3 = (EditText) findViewById(R.id.et3);
et4 = (EditText) findViewById(R.id.et4);
et5 = (EditText) findViewById(R.id.et5);
setContentView(R.layout.activity_main);
firebaseAuth=firebaseAuth.getInstance();
fstore = FirebaseFirestore.getInstance();
c1 = (CheckBox)findViewById(R.id.c1);
c1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        if(compoundButton.isChecked()){
            c2.setChecked(false);
        }
    }
});
c2 = (CheckBox)findViewById(R.id.c2);
c2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
```

```
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                if(compoundButton.isChecked()){
                    c1.setChecked(false);
                }
            }
        });
        signup = (Button) findViewById(R.id.b2);
        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(validate()){
                    String username=et3.getText().toString();
                    String password=et4.getText().toString();
                    String name=et5.getText().toString();
                  firebaseAuth.createUserWithEmailAndPassword(username,       password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if(task.isSuccessful()) {
                                FirebaseUser user=firebaseAuth.getCurrentUser();
                                Toast.makeText(MainActivity.this, "Success",Toast.LENGTH_LONG).show();
                                DocumentReference df=fstore.collection("Users").document(user.getUid());
                                Map<String,Object>userinfo=new HashMap<>();
                                userinfo.put("USERNAME",username);
                                userinfo.put("PASSWORD",password);
                                userinfo.put("NAME",name);
                                if(c1.isChecked()) {
                                    userinfo.put("isFreelancer", "1");
                                }
                                else if(c2.isChecked()){
                                    userinfo.put("isCompany", "1");
                                }
                                df.set(userinfo);
                                Intent intent=new Intent(MainActivity.this, MainActivity2.class);
                    startActivity(intent);
                }
                else{
                    Toast.makeText(MainActivity.this, "failed", Toast.LENGTH_LONG).show();
                }
            }
        });
    }
  }
});
tvlogin=(TextView)findViewById(R.id.tvlogin);
tvlogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this, MainActivity2.class);
        startActivity(intent);
    }
});
}
public boolean validate(){
    et3 = (EditText) findViewById(R.id.et3);
    et4 = (EditText) findViewById(R.id.et4);
    et5 = (EditText) findViewById(R.id.et5);
    Boolean result = true;
    String username=et3.getText().toString();
    String password=et4.getText().toString();
    String name=et5.getText().toString();
    if(username.isEmpty() ) {
        et3.setError("Enter Username");
        result=false;
    }
    else if(password.length()<6 || password.isEmpty()){
        et4.setError("Invalid Password");
        result=false;
    }
    else if(name.isEmpty()){
        et5.setError("Enter name");
        result=false;
    }
    return result;
}
```

Java code of Rating module

```
public class freelancerfeedback extends AppCompatActivity {
TextView ffname,ffexp,ffskills,ffusername,ffrate;
ImageView ffpimg;
Button rate;
private FirebaseAuth firebaseAuth;
private FirebaseFirestore fstore;
String l,a,b,c,d,e,s,f,g,k,fuid;
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                Intent intent=getIntent();
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_freelancerfeedback);
                firebaseAuth=firebaseAuth.getInstance();
                fstore = FirebaseFirestore.getInstance();
                Toolbar toolbar1 = findViewById(R.id.toolbar1);
                setSupportActionBar(toolbar1);
                BottomNavigationView cn9 = findViewById(R.id.company_bottom_navigation);
                cn9.setOnNavigationItemReselectedListener(new
              BottomNavigationView.OnNavigationItemReselectedListener() {
                    @Override
                    public void onNavigationItemReselected(@NonNull MenuItem item) {
```

```
        switch (item.getItemId()) {
            case R.id.company_navigation_home:{
                Intent v1=new Intent(freelancerfeedback.this,companyhomepage.class);
                startActivity(v1);
                break;
            }
            case R.id.company_navigation_search: {
                Intent v2 = new Intent(freelancerfeedback.this, freelancersearch.class);
                startActivity(v2);
                break;
            }
            case R.id.company_navigation_newproject: {
                Intent v3 = new Intent(freelancerfeedback.this, newproject.class);
                startActivity(v3);
                break;
            }
            case R.id.company_navigation_notifications:{
                Intent v4 = new Intent(freelancerfeedback.this, companynotification.class);
                startActivity(v4);
                break;
            }
        }
    }
});
ffname=(TextView)findViewById(R.id.ffname);
ffexp=(TextView)findViewById(R.id.ffexp);
ffskills=(TextView)findViewById(R.id.ffskills);
ffrate=(TextView)findViewById(R.id.ffrate);
ffusername=(TextView)findViewById(R.id.ffusername);
```

```
 ffpimg=(ImageView)findViewById(R.id.ffpimg);
 rate=(Button)findViewById(R.id.b13) ;
 Intent d=getIntent();
 String ffname1=d.getStringExtra("Name");
 fuid=d.getStringExtra("FUID");
 DocumentReference df1=fstore.collection("Freelancer Profile").document(fuid.toString());
 df1.addSnapshotListener(new EventListener<DocumentSnapshot>() {
     @Override
     public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value,
@Nullable @org.jetbrains
             .annotations.Nullable FirebaseFirestoreException error) {
         a=value.getString("Freelancer Phone No");
         b=value.getString("Address");
         c=value.getString("Date of Birth");
         s=value.getString("Img_URL");
         e=value.getString("Description");
         f=value.getString("Skills");
         g=value.getString("Username");

         ffname.setText(ffname1);
         ffexp.setText(e);
         ffskills.setText(f);
         ffusername.setText(g);
         Picasso.get().load(s).into(ffpimg);
     }
 });
```

```
            rate.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {


                    Intent i=new Intent(freelancerfeedback.this,editfreelancerfeedback.class);
                    i.putExtra("Name",ffname1);
                    i.putExtra("FUID",fuid);
                    startActivity(i);


                }
            });
            DocumentReference df=fstore.collection("Freelancer Profile").document(fuid);
            df.addSnapshotListener(new EventListener<DocumentSnapshot>() {
                @Override
                public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value,
          @Nullable @org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {
                    l=value.getString("Ratings");
                    ffrate.setText(l);


                }
            });
        }
}
```

Java Code of New Project

```
public class newproject extends AppCompatActivity {
EditText pt1,pt2,pt3,pt4,pt5;
TextView tvcname,profimg;
Button addproj;
private FirebaseFirestore fstore;
private FirebaseAuth firebaseAuth;
StorageReference sr,profref;
String a;
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                Intent intent=getIntent();
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_newproject);
                pt1=(EditText)findViewById(R.id.pt1);
                pt2=(EditText)findViewById(R.id.pt2);
                pt3=(EditText)findViewById(R.id.pt3);
                pt4=(EditText)findViewById(R.id.pt4);
                pt5=(EditText)findViewById(R.id.pt5);
                tvcname=(TextView) findViewById(R.id.tvcname);
                firebaseAuth=firebaseAuth.getInstance();
                fstore = FirebaseFirestore.getInstance();
                BottomNavigationView cn5=findViewById(R.id.company_bottom_navigation);
               cn5.setOnNavigationItemReselectedListener(new
              BottomNavigationView.OnNavigationItemReselectedListener() {
                    @Override
                    public void onNavigationItemReselected(@NonNull MenuItem item) {
                        switch(item.getItemId()){
                            case R.id.company_navigation_search: {
```

```
                Intent t1 = new Intent(newproject.this, freelancersearch.class);
                startActivity(t1);
                break;
            }
            case R.id.company_navigation_home :{
                Intent t2 = new Intent(newproject.this, companyhomepage.class);
                startActivity(t2);
                break;
            }
            case R.id.company_navigation_notifications: {
                Intent t3 = new Intent(newproject.this, companynotification.class);
                startActivity(t3);
                break;
            }
        }
    }
});
FirebaseUser user1=firebaseAuth.getCurrentUser();
DocumentReference df1 = fstore.collection("Company Profile").document(user1.getUid());
df1.addSnapshotListener(newproject.this, new EventListener<DocumentSnapshot>() {
            @Override
            public void onEvent(@Nullable DocumentSnapshot value1, @Nullable
FirebaseFirestoreException error) {
                tvcname.setText(value1.getString("Name"));
            }
        });

addproj= (Button) findViewById(R.id.b4);
addproj.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View view) {


    String name=tvcname.getText().toString();
    String title=pt1.getText().toString();
    String desc=pt2.getText().toString();
    String skills=pt3.getText().toString();
    String role=pt4.getText().toString();
    String duration=pt5.getText().toString();
    if(title.isEmpty()){
        pt1.setError("Enter Project Title");
    }
    else if(desc.isEmpty()){
        pt2.setError("Enter Project Description");
    }
    else if(role.isEmpty()){
        pt3.setError("Enter Project Role");
    }
    else if(skills.isEmpty()){
        pt4.setError("Enter Project Skills");
    }
    else if(duration.isEmpty()){
        pt5.setError("Enter Project Duration");
    }
    else{
        FirebaseUser user=firebaseAuth.getCurrentUser();
        DocumentReference df=fstore.collection("Projects").document(user.getUid());


        Map<String,Object> projectinfo=new HashMap<>();
```

```
                        projectinfo.put("Name",name);

                        projectinfo.put("Project_Title",title);

                        projectinfo.put("Project_Description",desc);

                        projectinfo.put("Project_Role",role);

                        projectinfo.put("Project_Skills",skills);

                        projectinfo.put("Project_Duration",duration);

                        projectinfo.put("PID",user.getUid());

                        df.set(projectinfo);

                        Toast.makeText(newproject.this, "Project added successfully",
                 Toast.LENGTH_LONG).show();


                      }

                    }

                 });

               }

}
```

Java Code of Freelancer Profile

```
public class editfreelancerprofile extends AppCompatActivity {
EditText pp11,pp13,pp14,pp15,pp16,pp17,pp18;
TextView pp12;
Button psave;
private FirebaseAuth firebaseAuth;
private FirebaseFirestore fstore;
ImageView fprofpic11;
StorageReference sr,profref,profref1;
String pimgurl,rate,sum,count;

@Override
protected void onCreate(Bundle savedInstanceState) {
   Intent intent=getIntent();
```

```
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_editfreelancerprofile);
        BottomNavigationView fn4=findViewById(R.id.freelancer_bottom_navigation);
        fn4.setOnNavigationItemReselectedListener(new
BottomNavigationView.OnNavigationItemReselectedListener() {
            @Override
            public void onNavigationItemReselected(@NonNull MenuItem item) {
                switch(item.getItemId()){
                    case R.id.freelancer_navigation_search:{
                        Intent d1=new Intent(editfreelancerprofile.this,companysearch.class);
                        startActivity(d1);
                        break;
                    }
                    case R.id.freelancer_navigation_home:{
                        Intent d2=new Intent(editfreelancerprofile.this,freelancerhomepage.class);
                        startActivity(d2);
                        break;
                    }
                    case R.id.freelancer_navigation_notifications:{
                        Intent d3=new Intent(editfreelancerprofile.this,freelancernotification.class);
                        startActivity(d3);
                        break;
                    }
                }
            }
        });
        firebaseAuth=firebaseAuth.getInstance();
        fstore = FirebaseFirestore.getInstance();
        sr= FirebaseStorage.getInstance().getReference();
        profref=sr.child("Users/"+firebaseAuth.getCurrentUser().getUid()+"/profile pic");
        profref1=sr.child("Users/"+firebaseAuth.getCurrentUser().getUid()+"/profile pic");

        pp11=(EditText)findViewById(R.id.pp11);
        pp12=(TextView)findViewById(R.id.pp12);
        pp13=(EditText)findViewById(R.id.pp13);
        pp14=(EditText)findViewById(R.id.pp14);
        pp15=(EditText)findViewById(R.id.pp15);
        pp16=(EditText)findViewById(R.id.pp16);
        pp17=(EditText)findViewById(R.id.pp17);

        String pname11=intent.getStringExtra("Name");
        String pusername11=intent.getStringExtra("Username");
        String pdob11=intent.getStringExtra("Date of Birth");
        String pno11=intent.getStringExtra("Freelancer Phone No");
        String paddress11=intent.getStringExtra("Address");
        String pexp11=intent.getStringExtra("Description");
        String pskills11=intent.getStringExtra("Skills");
        pp11.setText(pname11);
        pp12.setText(pusername11);
        pp13.setText(pdob11);
        pp14.setText(pno11);
        pp15.setText(paddress11);
        pp16.setText(pexp11);
        pp17.setText(pskills11);
        fprofpic11=findViewById(R.id.pimg1);
        FirebaseUser user=firebaseAuth.getCurrentUser();

        fprofpic11.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent opengalleryintent=new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(opengalleryintent,1000);
            }
        });
        profref.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri uri) {
                Picasso.get().load(uri).into(fprofpic11);
                pimgurl=uri.toString();
            }
        });
        DocumentReference df1=fstore.collection("Freelancer Profile").document(user.getUid());
        df1.addSnapshotListener(new EventListener<DocumentSnapshot>() {
            @Override
            public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value, @Nullable
@org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {
                rate =value.getString("Ratings");
                sum=value.getString("Sum");
                count=value.getString("Count");
            }
        });
        psave=(Button)findViewById(R.id.b9);
        psave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String pname1=pp11.getText().toString();
                String pusername1=pp12.getText().toString();
                String pdob1=pp13.getText().toString();
                String pno1=pp14.getText().toString();
                String paddress1=pp15.getText().toString();
                String pexp1=pp16.getText().toString();
                String pskills1=pp17.getText().toString();

                DocumentReference df=fstore.collection("Freelancer Profile").document(user.getUid());
                Map<String,Object> pprofileinfo=new HashMap<>();
                pprofileinfo.put("Name",pname1);
                pprofileinfo.put("Username",pusername1);
                pprofileinfo.put("Date of Birth",pdob1);
                pprofileinfo.put("Freelancer Phone No",pno1);
                pprofileinfo.put("Address",paddress1);
                pprofileinfo.put("Description",pexp1);
                pprofileinfo.put("Skills",pskills1);
                pprofileinfo.put("Img_URL",pimgurl);
                pprofileinfo.put("Ratings",rate);
                pprofileinfo.put("FUID",user.getUid().toString());
                if(sum==null){
                    pprofileinfo.put("Sum","0");
                    pprofileinfo.put("Count","0");
                }
                else{
                    pprofileinfo.put("Sum",sum);
                    pprofileinfo.put("Count",count);
                }
                df.set(pprofileinfo);
                Intent i=new Intent(editfreelancerprofile.this,freelancerprofile.class);
                startActivity(i);

            }
        });
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode==1000){
            if(resultCode== Activity.RESULT_OK){
                Uri imageuri=data.getData();
                uploadimagetofirebase(imageuri);
            }
        }

    }
    private void uploadimagetofirebase(Uri imageuri) {
        final ProgressDialog pd=new ProgressDialog(editfreelancerprofile.this);
        pd.setTitle("Uploading Photo");
        pd.show();
        StorageReference fileref=sr.child("Users/"+firebaseAuth.getCurrentUser().getUid()+"/profile pic");
        fileref.putFile(imageuri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                Toast.makeText(editfreelancerprofile.this, "Image Successfully uploaded",Toast.LENGTH_LONG);
                pd.dismiss();
                fileref.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                    @Override
                    public void onSuccess(Uri uri) {
                        Picasso.get().load(uri).into(fprofpic11);
                    }
                });
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(editfreelancerprofile.this, "Failed",Toast.LENGTH_LONG);
            }
        }).addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onProgress(@NonNull UploadTask.TaskSnapshot snapshot) {
                double p=(100.0*snapshot.getBytesTransferred())/snapshot.getTotalByteCount();
                pd.setMessage("Uploaded: "+(int)p+"%");
            }
        });

    }
```

Java Code of Freelancer Notification

```
public class freelancernotification extends AppCompatActivity {
private FirebaseAuth firebaseAuth;
private FirebaseFirestore fstore;
RecyclerView calis;
FirebaseRecyclerAdapter adt;
DatabaseReference dr;
String img,exp,sk,us;

@Override
 protected void onCreate(Bundle savedInstanceState) {
 Intent intent = getIntent();
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_freelancernotification);
 firebaseAuth=firebaseAuth.getInstance();
 fstore = FirebaseFirestore.getInstance();
 Toolbar toolbar = findViewById(R.id.toolbar);
 setSupportActionBar(toolbar);
 BottomNavigationView fn5=findViewById(R.id.freelancer_bottom_navigation);
 fn5.setOnNavigationItemReselectedListener(new
BottomNavigationView.OnNavigationItemReselectedListener() {
     @Override
     public void onNavigationItemReselected(@NonNull MenuItem item) {
         switch(item.getItemId()){
             case R.id.freelancer_navigation_search:{
                 Intent e1=new Intent(freelancernotification.this,companysearch.class);
                 startActivity(e1);
                 break;
             }
             case R.id.freelancer_navigation_home:{
                 Intent e2=new Intent(freelancernotification.this,freelancerhomepage.class);
                 startActivity(e2);
                 break;
             }
```

```
            }
        }
    });
    calis=(RecyclerView)findViewById(R.id.calist);
    dr=FirebaseDatabase.getInstance().getReference("Connected").child(firebaseAuth.
            getCurrentUser().getUid().toString()).child("Company");
    FirebaseRecyclerOptions<calist> options1=new FirebaseRecyclerOptions.Builder<calist>().
            setQuery(dr,calist.class).build();
    adt= new FirebaseRecyclerAdapter<calist, hiredviewholder>(options1) {
        @NonNull
        @NotNull
        @Override
        public hiredviewholder onCreateViewHolder(@NonNull @NotNull ViewGroup parent, int viewType) {
            View view= LayoutInflater.from(parent.getContext()).inflate(R.layout.calist_item,parent,false);
            return new hiredviewholder(view);
        }
        @Override
        protected void onBindViewHolder(@NonNull @NotNull hiredviewholder holder, int position, @NonNull
@NotNull calist model) {

            DocumentReference df=fstore.collection("Company Profile").document(model.getCUID().toString());
            df.addSnapshotListener(new EventListener<DocumentSnapshot>() {
                @Override
                public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value,
@Nullable @org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {
                    holder.caname.setText(model.getName()+" has requested for a connection.");
                    img=value.getString("Img_URL");
                    us=value.getString("Username");
                    Picasso.get().load(img).into(holder.caprofimg);
                }
            });
            Picasso.get().load(model.getImg_URL()).into(holder.caprofimg);
            holder.caloyout.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Intent d=new Intent(freelancernotification.this,companydetails.class);
                    d.putExtra("Name",model.getName());
                    d.putExtra("CUID",model.getCUID().toString());
                    startActivity(d);
                }
            });
        }
    } ;
    calis.setHasFixedSize(true);
    calis.setLayoutManager(new LinearLayoutManager(this));
    calis.setAdapter(adt);
}
private class hiredviewholder extends RecyclerView.ViewHolder {
    TextView caname;
    ImageView caprofimg;
    LinearLayout caloyout;

    public hiredviewholder(@NonNull @NotNull View itemView) {
        super(itemView);
        caname=itemView.findViewById(R.id.caname);
        caprofimg=itemView.findViewById(R.id.caprofimg);
        caloyout=itemView.findViewById(R.id.calayout);
    }
}

@Override
protected void onStart() {
    super.onStart();
    adt.startListening();
}

@Override
protected void onStop() {
        super.onStop();
        adt.stopListening();
    }
}
```

Java Code for Hire Module

```
    public class freelancerdetails extends AppCompatActivity {
    TextView fdname,fdexp,fdskills,fdusername,fdrate;
    ImageView fdpimg;
    Button hire;
    private FirebaseAuth firebaseAuth;
    private FirebaseFirestore fstore;
    StorageReference sr,profref;
    String p,q,r,x,y,h,fdusername1,fdpimg1,fdskills1,fdexp1,fdrate1;
    List<String> list;
    boolean test = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Intent z=getIntent();
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_freelancerdetails);
        firebaseAuth=firebaseAuth.getInstance();
        fstore = FirebaseFirestore.getInstance();
        Toolbar toolbar1 = findViewById(R.id.toolbar1);
        setSupportActionBar(toolbar1);
        BottomNavigationView cn7 = findViewById(R.id.company_bottom_navigation);
        cn7.setOnNavigationItemReselectedListener(new
BottomNavigationView.OnNavigationItemReselectedListener() {
            @Override
            public void onNavigationItemReselected(@NonNull MenuItem item) {
                switch (item.getItemId()) {
                    case R.id.company_navigation_home:{
```

```
                Intent v1=new Intent(freelancerdetails.this,companyhomepage.class);
                startActivity(v1);
                break;
            }
            case R.id.company_navigation_search: {
                Intent v2 = new Intent(freelancerdetails.this, freelancersearch.class);
                startActivity(v2);
                break;
            }
            case R.id.company_navigation_newproject: {
                Intent v3 = new Intent(freelancerdetails.this, newproject.class);
                startActivity(v3);
                break;
            }
            case R.id.company_navigation_notifications:{
                Intent v4 = new Intent(freelancerdetails.this, companynotification.class);
                startActivity(v4);
                break;
            }
        }
    }
});
fdname=(TextView)findViewById(R.id.fdname);
fdpimg=(ImageView)findViewById(R.id.fdpimg);
fdskills=(TextView)findViewById(R.id.fdskills);
fdexp=(TextView)findViewById(R.id.fdexp);
fdusername=(TextView)findViewById(R.id.fdusername);
fdrate=(TextView)findViewById(R.id.fdrate);
Intent d=getIntent();
String fdname1=d.getStringExtra("Name");
r=d.getStringExtra("FUID");
DocumentReference df=fstore.collection("Freelancer Profile").document(r);
df.addSnapshotListener(new EventListener<DocumentSnapshot>() {
    @Override
```

```
        public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value, @Nullable
@org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {
            fdusername1=value.getString("Username");
            fdpimg1=value.getString("Img_URL");
            fdexp1=value.getString("Description");
            fdskills1=value.getString("Skills");
            fdrate1=value.getString("Ratings");
            fdusername.setText(fdusername1);
            fdname.setText(fdname1);
            Picasso.get().load(fdpimg1).into(fdpimg);
            fdexp.setText(fdexp1);
            fdskills.setText(fdskills1);
            fdrate.setText(fdrate1);
        }
    });
    hire=(Button)findViewById(R.id.b12);
    list = new ArrayList<>();
    fstore.collection("Hired").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @SuppressLint("LongLogTag")
        @Override
        public void onComplete(@NonNull @NotNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {

                for (QueryDocumentSnapshot document : task.getResult()) {
                    list.add(document.getId());
                }
                Log.d( "DOClist",list.toString());
                for(String idl:list) {
                    if(idl.equals(r)){
                        test = true;
                        Log.d("DOCTrue",list.toString());
                        break;
                    }
                }
                if(test==true){
                    hire.setEnabled(false);
                }
                else{
                    hire.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            hire.setText("Hired");
                            FirebaseUser user1=firebaseAuth.getCurrentUser();
                            DocumentReference df1 = fstore.collection("Company Profile").document(user1.getUid());
                            df1.addSnapshotListener(freelancerdetails.this, new EventListener<DocumentSnapshot>() {
                                @Override
                                public void onEvent(@Nullable DocumentSnapshot value1, @Nullable
FirebaseFirestoreException error) {
                                    p=value1.getString("Name");
                                    q=value1.getString("Img_URL");
                                    x=value1.getString("Description");
                                    y=value1.getString("Username");
                                }
                            });
                            DocumentReference tf=fstore.collection("Hired").document(r);
                            Map<String,Object> hireinfo=new HashMap<>();
                            hireinfo.put("Name",p);
                            hireinfo.put("CUID",firebaseAuth.getCurrentUser().getUid().toString());
                            tf.set(hireinfo);

                            DatabaseReference ref=FirebaseDatabase.getInstance().getReference().child("CompanyHired")
                                    .child(firebaseAuth.getCurrentUser().getUid().toString())
                                    .child("Freelancers Hired").child(r.toString());
                            Map<String,Object> chireinfo=new HashMap<>();
                            chireinfo.put("Name",fdname1);
                            chireinfo.put("FUID",r);
                            ref.setValue(chireinfo);
                        }
                    });
                }
            } else {
                Log.d( "Error getting documents: ", String.valueOf(task.getException()));
            }
        }
    });
  }
}
```

Java Code for Freelancer Homepage

```
  public class freelancerhomepage extends AppCompatActivity {
  private FirebaseAuth firebaseAuth;
  private FirebaseFirestore fstore;
  RecyclerView clist;
  FirestoreRecyclerAdapter adapter;
  String a,img;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      Intent z = getIntent();
      super.onCreate(savedInstanceState);
      firebaseAuth=firebaseAuth.getInstance();
      fstore = FirebaseFirestore.getInstance();
      setContentView(R.layout.activity_freelancerhomepage);
      Toolbar toolbar = findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);
      BottomNavigationView fn1=findViewById(R.id.freelancer_bottom_navigation);
      fn1.setOnNavigationItemReselectedListener(new
BottomNavigationView.OnNavigationItemReselectedListener() {
          @Override
          public void onNavigationItemReselected(@NonNull MenuItem item) {
            switch(item.getItemId()){
                case R.id.freelancer_navigation_search:{
                    Intent a1=new Intent(freelancerhomepage.this,companysearch.class);
                    startActivity(a1);
                    break;
                }
                case R.id.freelancer_navigation_notifications:{
                    Intent a3=new Intent(freelancerhomepage.this,freelancernotification.class);
                    startActivity(a3);
                    break;
                }
            }
        }
    });



    clist=(RecyclerView)findViewById(R.id.clist);
    Query query = fstore.collection("Projects");
    FirestoreRecyclerOptions <cpost> options=new
FirestoreRecyclerOptions.Builder<cpost>().setQuery(query,cpost.class).build();
    adapter= new FirestoreRecyclerAdapter<cpost, ProjectViewHolder>(options) {
        @NonNull
        @Override
        public ProjectViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
            View view=LayoutInflater.from(parent.getContext()).inflate(R.layout.cpost_item,parent,false);
            return new ProjectViewHolder(view);
        }


        @Override
        protected void onBindViewHolder(@NonNull ProjectViewHolder holder, int position, @NonNull cpost
model) {
            DocumentReference qq=fstore.collection("Company Profile").document(model.getPID());
            qq.addSnapshotListener(new EventListener<DocumentSnapshot>() {
                @Override
                public void onEvent(@Nullable @org.jetbrains.annotations.Nullable DocumentSnapshot value,
@Nullable @org.jetbrains.annotations.Nullable FirebaseFirestoreException error) {
                    img=value.getString("Img_URL");
                    Picasso.get().load(img).into(holder.profimg1);
                    holder.cname.setText(model.getName());
                    holder.fhptitle.setText(model.getProject_Title());
                    holder.fhpdesc.setText(model.getProject_Description());
                    holder.fhprole.setText(model.getProject_Role());
                    holder.fhpskills.setText(model.getProject_Skills());
                    holder.fhpduration.setText(model.getProject_Duration());
                }
            });
            holder.aply.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    holder.aply.setText("Applied");
                    FirebaseUser user1=firebaseAuth.getCurrentUser();
                    DocumentReference df1 = fstore.collection("Freelancer Profile").document(user1.getUid());
                    df1.addSnapshotListener(freelancerhomepage.this, new EventListener<DocumentSnapshot>() {
                        @Override
                        public void onEvent(@Nullable DocumentSnapshot value1, @Nullable
FirebaseFirestoreException error) {
                            a=value1.getString("Name");
                        }
                    });
                    DatabaseReference dr=FirebaseDatabase.getInstance().getReference().child("Applied")

.child(model.getPID().toString()).child("Freelancers").child(firebaseAuth.getCurrentUser().getUid().toString());
                    Map<String,Object> appinfo=new HashMap<>();
                    appinfo.put("Name",a);
                    appinfo.put("FUID",firebaseAuth.getCurrentUser().getUid().toString());
                    dr.setValue(appinfo);
                }
            });
```

```
        }
    };
    clist.setHasFixedSize(true);
    clist.setLayoutManager(new LinearLayoutManager(this));
    clist.setAdapter(adapter);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater1 = getMenuInflater();
    inflater1.inflate(R.menu.freelancertoolbar, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.profile: {
            Intent intent = new Intent(freelancerhomepage.this, freelancerprofile.class);
            startActivity(intent);
            return true;
        }
        case R.id.signout: {
            Intent intent = new Intent(freelancerhomepage.this, MainActivity2.class);
            startActivity(intent);
            return true;
        }
        case R.id.message:{
            Intent intent = new Intent(freelancerhomepage.this, freelancermessage.class);
            startActivity(intent);
            return true;
        }
    }
    return super.onOptionsItemSelected(item);
}
private class ProjectViewHolder extends RecyclerView.ViewHolder{
    private TextView fhptitle,fhpdesc,fhprole,fhpskills,fhpduration,cname;
    private ImageView profimg1;
    private Button aply;
    public ProjectViewHolder(@NonNull View itemView) {
        super(itemView);
        profimg1=itemView.findViewById(R.id.fhcprofimg);
        cname=itemView.findViewById(R.id.fhcname);
        fhptitle=itemView.findViewById(R.id.fhptitle);
        fhpdesc=itemView.findViewById(R.id.fhpdesc);
        fhprole=itemView.findViewById(R.id.fhprole);
        fhpskills=itemView.findViewById(R.id.fhpskills);
        fhpduration=itemView.findViewById(R.id.fhpduration);
        aply=itemView.findViewById(R.id.b10);
    }
}
@Override
protected void onStart() {
    super.onStart();
    adapter.startListening();
}
@Override
protected void onStop() {
    super.onStop();
    adapter.stopListening();
}
private class imageViewHolder extends RecyclerView.ViewHolder{
    public imageViewHolder(@NonNull View itemView) {
        super(itemView);
    }
}}
```

REFERENCES

[1]. Chunnu Khawas, Pritam Shah.(2018) “Application of Firebase in Android App Development – A Study”, International Journal of Computer Applications, Vol.179, No.46,pp.0975-8887.

[2]. Harleen K. Flora, Xiaofeng Wang, Swati V. Chande.(2014) “An Investigation into Mobile Application Development Processes: Challenges and Best Practices”, I.J Modern Education and Computer Science, No.6, pp.1-9.

[3]. U.YelizEseryel, DenizEseryel, Richard Hendrik Booji.(2020) “Social Media Use for Online Interaction: Lessons Learned from Fortune 100 Companies on Job Applicant Attraction”, Journal of Leadership and Management, pp.1-18.

[4]. Top 25 Freelance Websites to Find Work in 2021 (dynomapper.com) “Top 25 Freelance websites in 2020”