TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

We use many applications in smart phones which can be downloaded from google play store but it only shows apps based on our search name and keyword it does not show application on the user requirements, there are many problems for user while finding a best application.

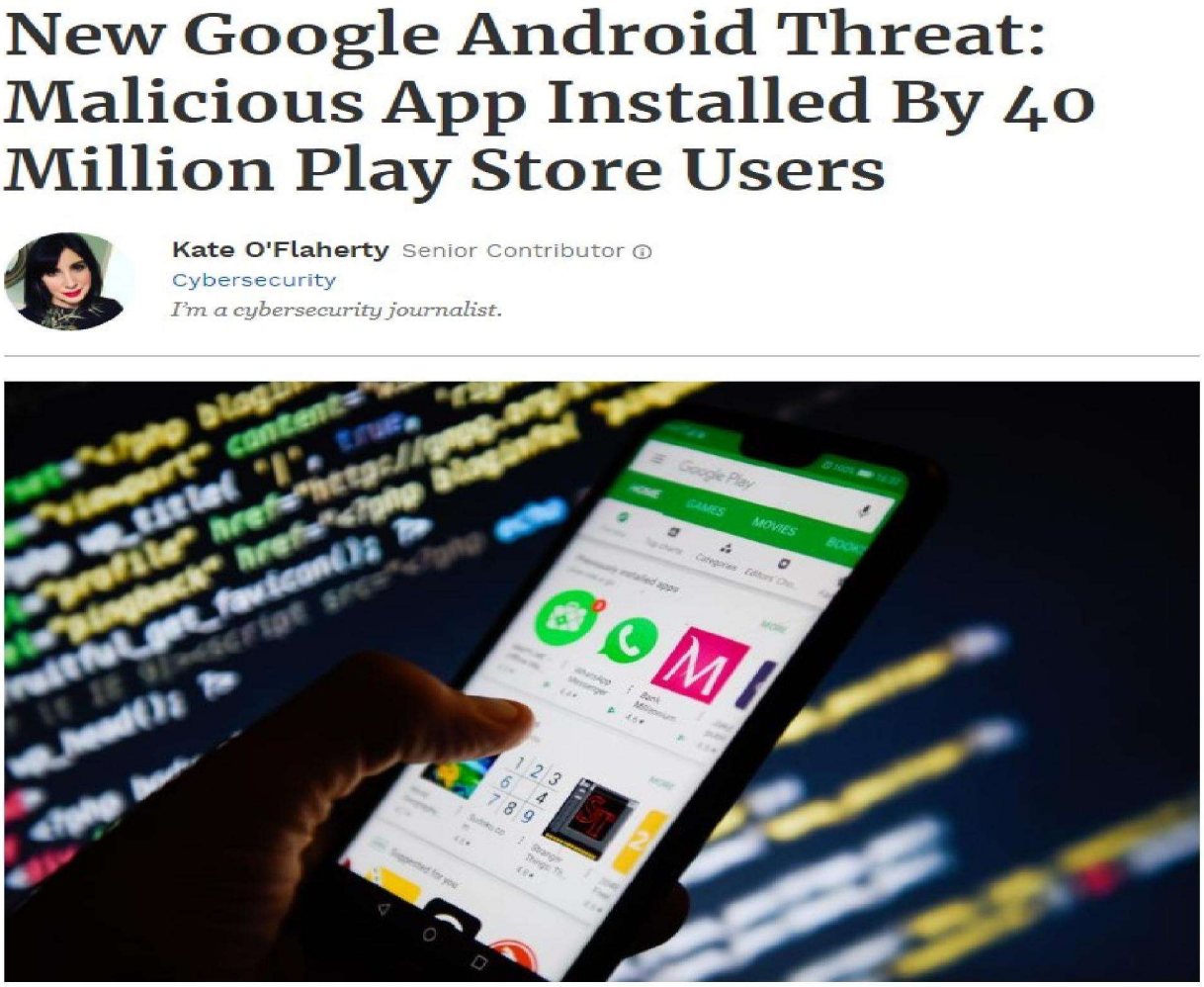


Fig 1.1: *News about malware apps in google Playstore*

/

Recently, google Playstore removed some of the malicious and harmful apps from the Playstore which has millions of downloads, there are many more apps which users download for their requirements but that particular app they choose from those many may not meet their requirement in turn there will be some of the apps which are harmful and steals the privacy of the users.

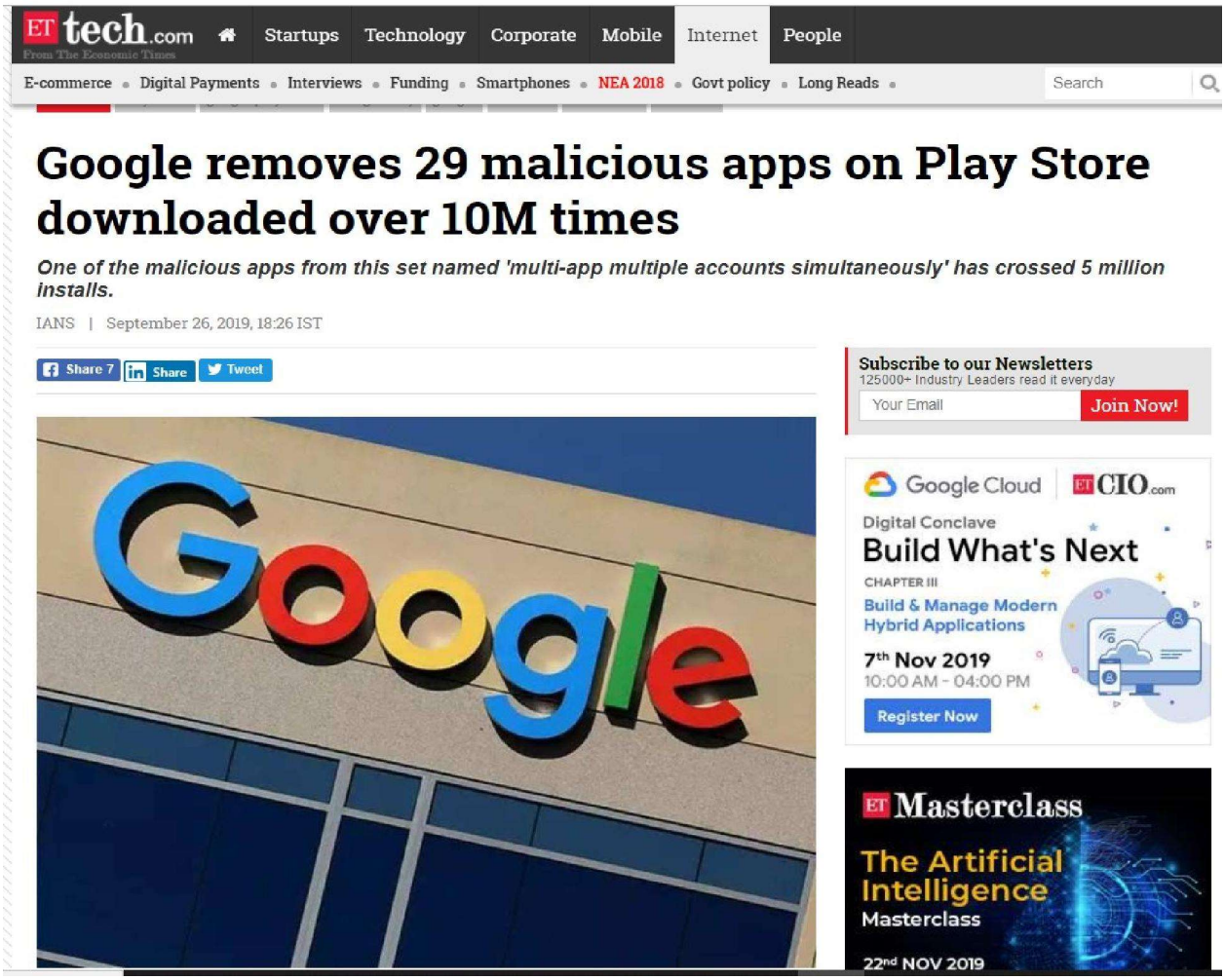


Fig 1.2: News showing google recently removed harmful apps

Fig 1.3 Sample image showing apps with similar names

In Playstore it can be noticed that we find a lot of apps with different names but same features but some may meet the user requirement.

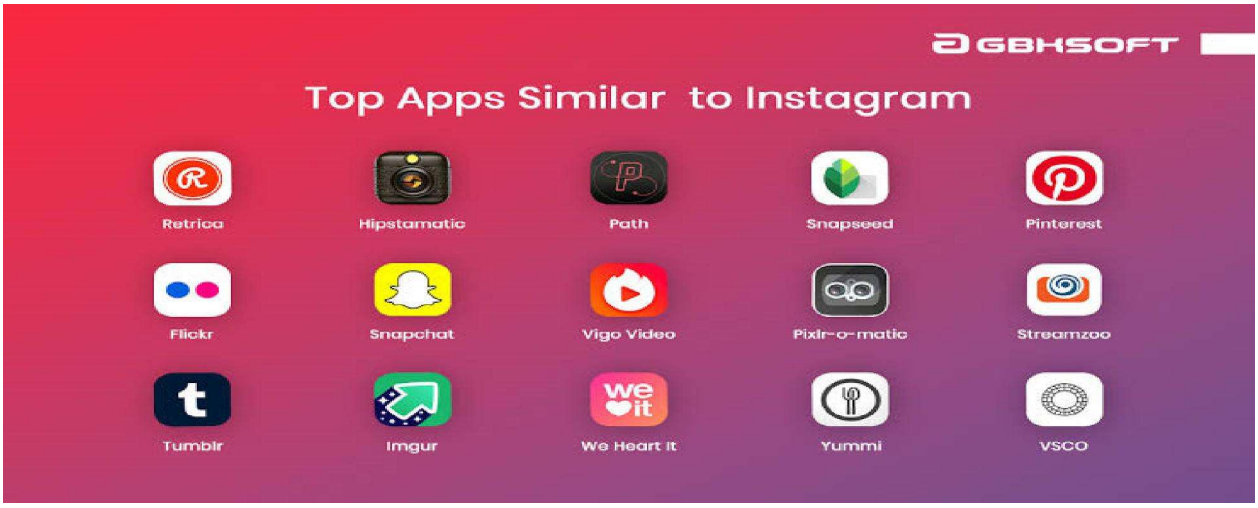


Fig 1.4 Sample image showing example of apps with similar names

There are many apps in the Playstore that show almost same names and user wastes a lot of time in downloading and installing which leads to loss of time and money.

There are also many security issues while installing android applications,

In early March 2011, Droid Dream, an exploit, was released to the then-named Android Market in the form of several free applications that were, in many cases pirated version of existing priced apps.

This exploit allowed hackers to steal information such as imei and imsi numbers, phone model, user ID, and service provider. The exploit also installed a backdoor that allowed the hackers to download more code to the infected device. The exploit only affected devices running Android versions earlier than 2.3 "Gingerbread. Google removed the apps from the Market immediately after being alerted, but the apps had already been downloaded more than 50,000 times, according to *Android Police*'s estimate.

Android Police wrote that the only method of removing the exploit from an infected device was to reset it to a factory state, although community-developed solutions for blocking some aspects of the exploit were created. A few days later, Google confirmed that 58 malicious apps had been uploaded to Android Market, and had been downloaded to 260,000 devices before being removed from the store.

In all of 2017, over 700,000 apps were banned from Google Play due to abusive contents; this is a 70% increase over the number of apps banned in 2016.

Link: https://en.wikipedia.org/wiki/Google_Play#Security_issues

So we are building a software that predicts and shows best application for user based on his requirements.

1.1 PROBLEM STATEMENT

The usage of smartphones has been increasing exponentially. This has resulted in a wide range of applications in the google play store. We see a lot of applications with similar names and features and also a lot of malware apps that steal the privacy of the users data Naive users would get into troubles by installing the applications which don’t meet their requirements and eventually result in wasting up a lot of time and cost of the user. Nowadays there are lot of apps that steal the confidential data of users and which leads to theft of money termed as cybercrimes.

1.2 OBJECTIVES

The objectives of SeWiCo are as follows:

a) **To predict best application that meet user requirement:** We see many applications and install it and use it for some period and they find it is not useful in turn leads to waste of time and money so we predict best application to the user which meets respective requirements based on user search query.

b) **To predict best app with similar features they used before:** Suppose, user already issuing an application and no longer interested in using the application we predict the best app with similar features.

c) **To provide link for downloading an application:** After predicting the best application for the user providing a link to download the application.

1.3 MOTIVATION

While downloading applications from Playstore most of the times after downloading it shows Error, try again and even after downloading and using the applications there is cases that it may or many not meet the user requirements he/she is interested in so there is loss of data, time and privacy.

So we come up with a solution SeWiCo, The software predicts the best application for user that serves their requirement and it does not show any errors or steal privacy of user's data and it results in saving lot of time and money of the users.

1.4 EXISTING SYSTEM

There is google Playstore but it only shows apps based on our search name it do not shows applications based on user requirements there are millions of applications in the play store of different categories in which each category of applications serves specific requirements But users are not able to choose apps that could better serve their requirements.

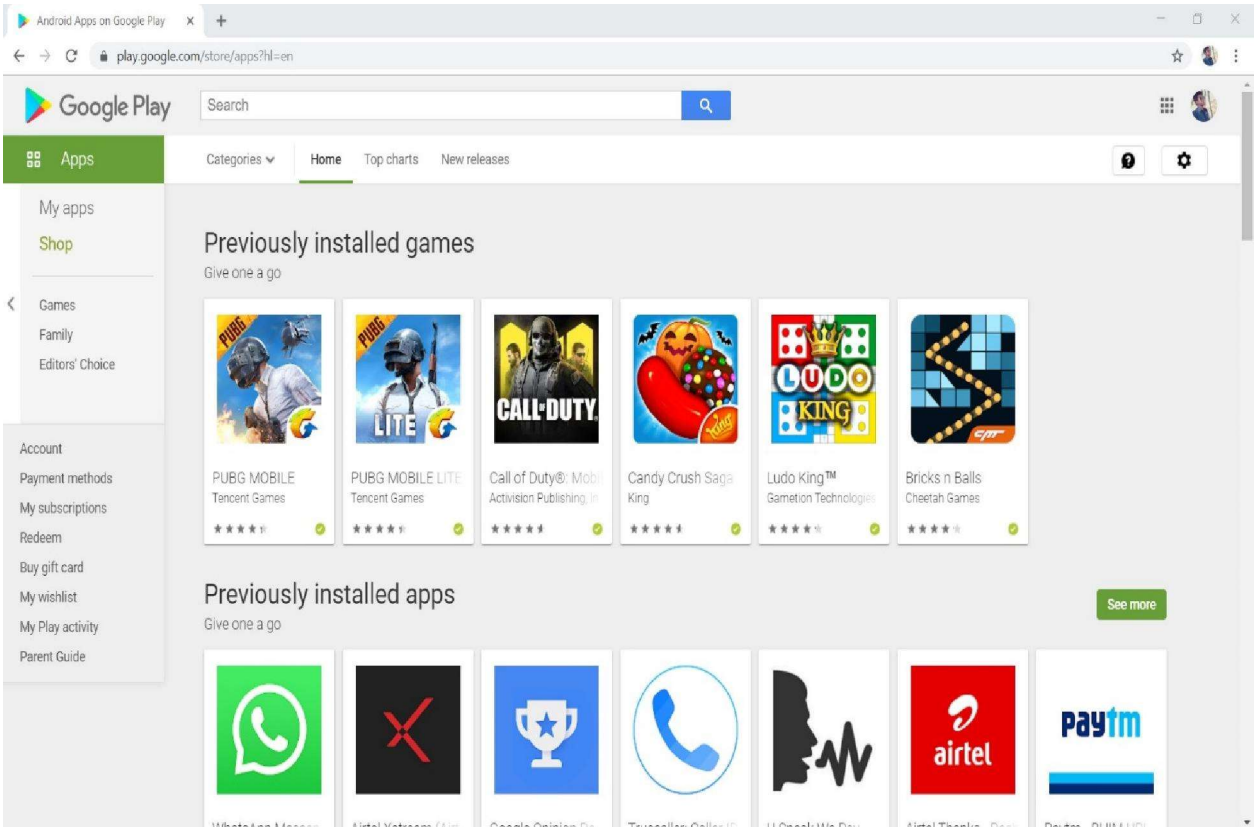


Fig 1.4 Existing system of Playstore

Google Play, formerly Android Market, is a digital distribution service operated and developed by Google. It serves as the official app store for the Android operating system, allowing users to browse and download applications developed with the Android software development kit (SDK) and published through Google.

Google Play also serves as a digital media store, offering music, books, movies, and television programs. It previously offered Google hardware devices for purchase until the introduction of a separate online hardware retailer, Google Store, on March 11, 2015, and it also offered news publications and magazines before the revamp of Google News on May 15, 2018.

As of 2017, Google Play features over 3.5 million Android applications. Users in over 145 countries can purchase apps, although Google notes on its support pages that "Paid content may not be available in some provinces or territories, even if the governing country is listed above. Developers in over 150 locations can distribute apps on Google Play.

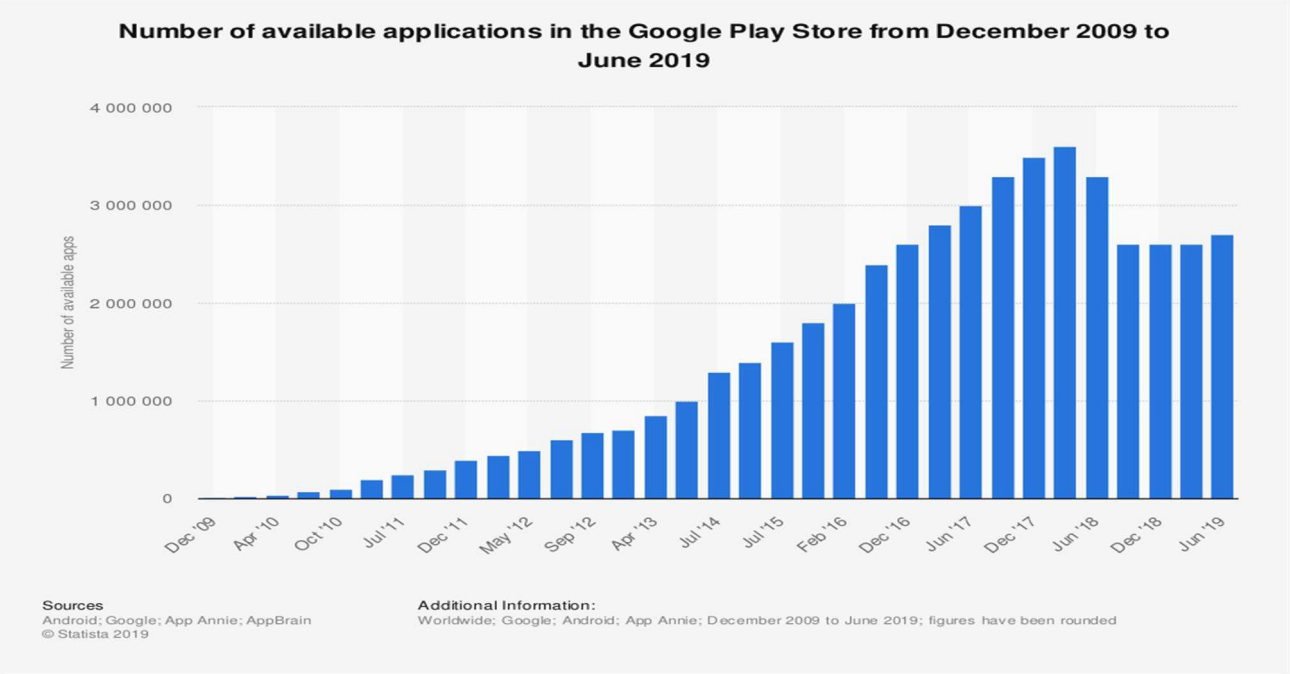


Fig 1.5 No of installs of apps from Playstore

The Number of apps in Playstore are increasing in large numbers and number of installs of applications are also increasing but there are many applications which do not serve user requirements and there are many applications which are harmful apps.

1.5 PROPOSED SYSTEM

This project helps to predict dynamically a better application that could serve the user requirements based on the past reviews, ratings, resource consumption, genres, content rating etc., and displays top 5 applications for a given key search that reduces hassle for the user.

Then the software provides link of the application to be downloaded and android application is developed for user interface. The project implementation is planned using AI & ML techniques and Big Data Analytics The application continuously learns by itself based on user reward.

1.6 SCOPE

The application we are developing can be used by all the persons who are using android smartphones only and have access to internet. The application can be used by every person who uses an android application. The application cannot be used in windows and ios platforms. The application is useful to users downloading applications from Playstore and is also useful to users who had already downloaded app directly from Playstore without using sewico (as the app downloaded from Playstore does not meet their requirements they can use sewico for apps with similar features.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING SYSTEM

There is google Playstore but it only shows apps based on our search name it do not shows applications based on user requirements. In the Playstore there are millions of application some applications with same names and there are many applications with similar features.

The user enters an application name or any keyword and the system displays applications but all the users may not know the application names or keywords and those apps may or may not meet user requirements.

Then user downloads and installs the applications there may be lot of problems while downloading and using the applications

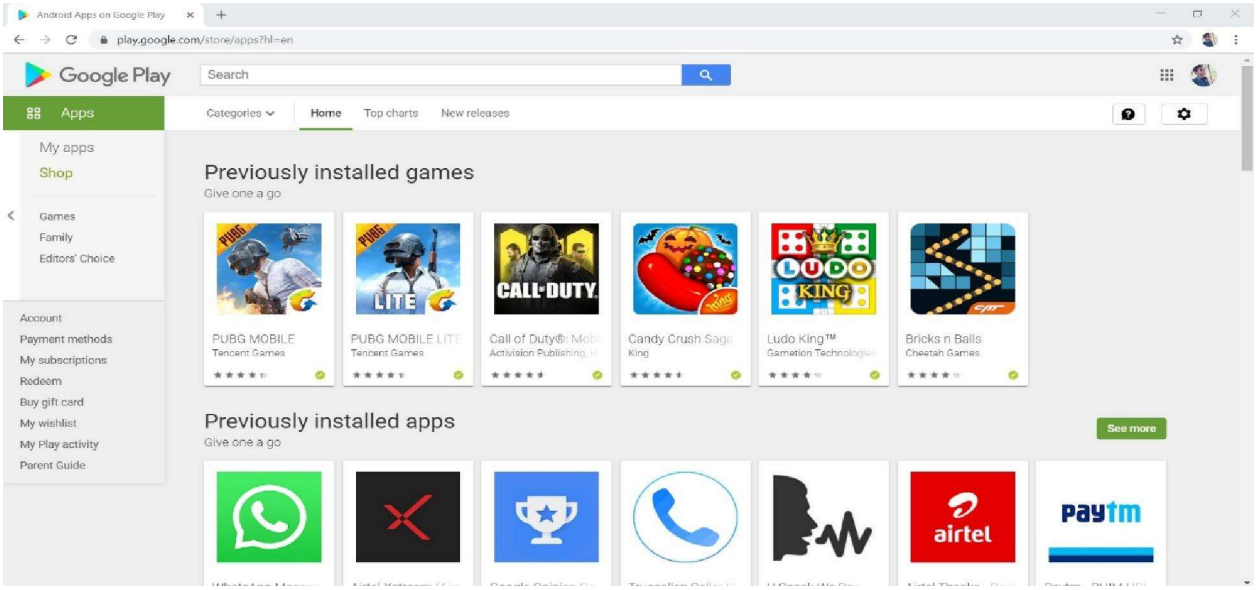


Fig 2.1 searching of applications in Playstore

2.2 PROPOSAL

To address the problem discussed we come up with a solution and that is “SeWiCo”. This project helps to predict dynamically a better application that could serve the user requirements based on the past reviews, ratings, resource consumption, genres, content rating etc., and displays top 5 applications for a given key search that reduces hassle for the user.

The project implementation is planned using AI & ML techniques and Big Data Analytics. The user enters a keyword and user requirements in search and then Playstore retrieves all the applications matching the respective keyword.

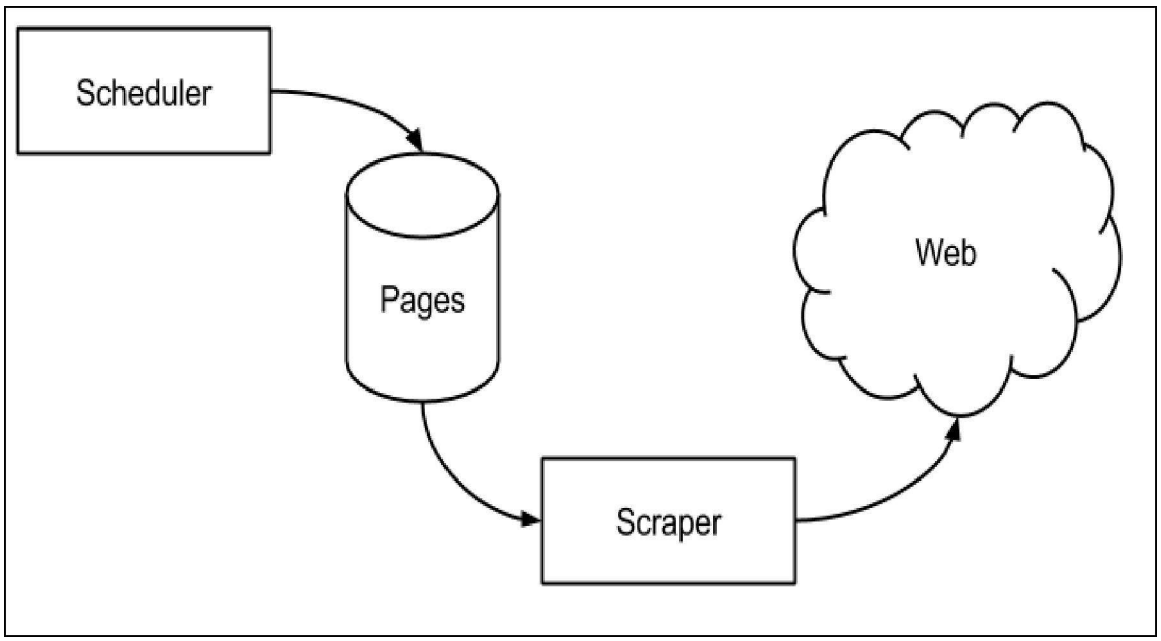


Fig 2.2 web scrapping technique architecture

The Dataset of Google Playstore is collected using web scarping, by using a python Library "Beautiful soup".

By performing web scrapping the data of 70,000 applications is collected from play store

The applications of 62 Categories of Playstore is collected and features Include:

- App name
- Category
- Company
- Rating
- Size
- Content Rating
- Number of installs
- Last Updated
- Android version support
- Current Version
- Ads(yes/no)
- Interactive Elements
- Editor's Choice(yes/no)
- Type(free/paid)
- Price
- Permissions(yes/no)
- Link of app
- Number of reviews
- Number of ratings
- updates(yes/no)
- Supports ios(yes/no)
- Supports windows(yes/no)
- What's new
- website available(yes/no)
- Percentage of reviews per downloads
- Percentage of ratings per downloads
- number of 5 star ratings
- number of 4 star ratings
- number of 3 star ratings

There are many categories of applications in Playstore and some may be free apps and other apps to use should be paid , the below pie chart shows percentage of applications of each category present in Playstore in outer layer and inner layer consists of percentage of paid and free apps in each category.

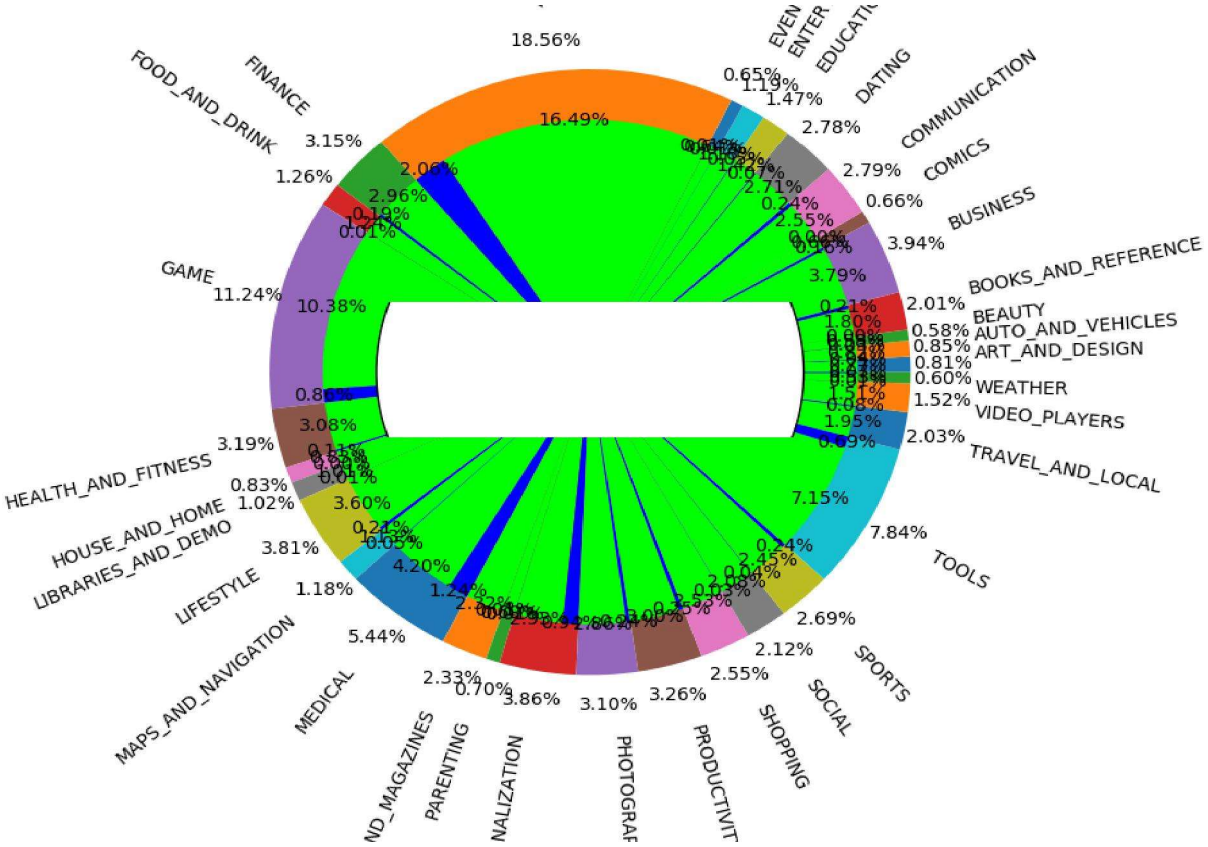


Fig: 2. 3 Percentage of apps of different categories and percentage of paid/free apps

On this dataset one of the reinforcement learning algorithm, i.e. Q learning algorithm is used and values are predicted.

The user requirements are collected and the similarity between the features of every app in the playstore dataset collected is calculated and random forest and Q learning algorithms are applied to predict the best application to user based on his key search.

The user enters the android application by registering and user enters a search query i.e. what are his requirements that will be the state in our model.

The user enters the search keyword in the application then the application suggests the application names matching the keyword the user is given option to select a app then user enters his requirements.

The similarity between user requirements and features of every app in dataset collected is calculated that acts as Q values for every app.

Then based on user search key all apps which have similar names and features and suggestions matching are processed and Q values are predicted based on q values of dataset using random forest algorithm.

The app is recommended to the user which has maximum Q value and link is provided to download the app.

Then user is must give Reward(feedback) to the app, Based on the user feedback the Q learning algorithm is applied and the model learns again and predicts the best application.

The model based on certain rules returns possible Q actions and the output application with highest Q action is regarded as best application.

The model learns by itself based on the Q values and predicts the best application.

Each time the model is trained it receives a feedback from environment in two states of values either reward (1) or punish (0) based on the values to the respective q values the feedback is added and in the next iteration the model predicts best output with new q values, so the model learns by itself continuously and gives the best output based on environment (user) .The algorithm is a type of reinforcement learning algorithm.

REINFORCEMENT LEARNING: A reinforcement learning task is about training an **agent** which interacts with its **environment**. The agent arrives at different scenarios known as **states** by performing **actions**. Actions lead to rewards which could be positive and negative.

The agent has only one purpose here – to maximize its total reward across an **episode.** This episode is anything and everything that happens between the first state and the last or terminal state within the environment.

We reinforce the agent to learn to perform the best actions by experience. This is the strategy or **policy**.

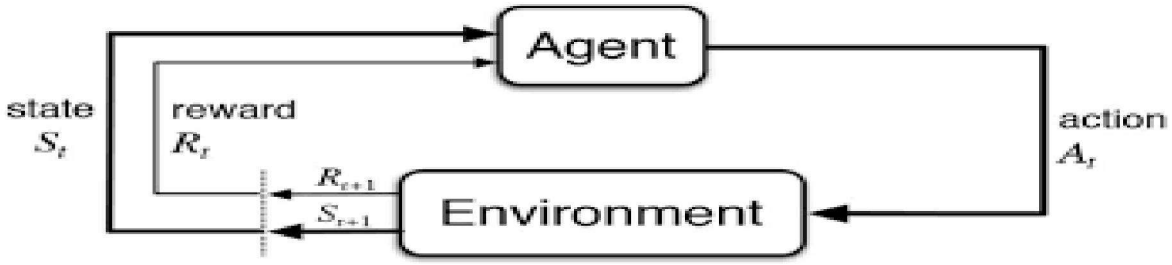


Fig: 2.3 Reinforcement learning working

What is Q learning?

Let's say we know the expected reward of each action at every step. This would essentially be like a cheat sheet for the agent! Our agent will know exactly which action to perform.

It will perform the sequence of actions that will eventually generate the maximum total reward. This total reward is also called the Q-value and we will formalize our strategy as:

The above equation states that the Q-value yielded from being at state *s* and performing action *a* is the immediate reward r(s', a) plus the highest Q-value possible from the next state'*s'*. Gamma here is the discount factor which controls the contribution of rewards further in the future. Q(s', a) again depends on Q(s', a) which will then have a coefficient of gamma squared.

So, the Q-value depends on Q-values of future states as

shown here:

Equation No. 1: $Q(s, a) = yQ(s', a) + y^2 Q(s'', a) \ldots \ldots \ldots y^n Q(s''^{\ldots n}, a)$

Adjusting the value of gamma will diminish or increase the contribution of future rewards. Since this is a recursive equation, we can start with making arbitrary assumptions for all q-values. With experience, it will converge to the optimal policy.

In practical situations, this is implemented as an update where alpha is the learning rate or step size. This simply determines to what extent newly acquired information overrides old information.

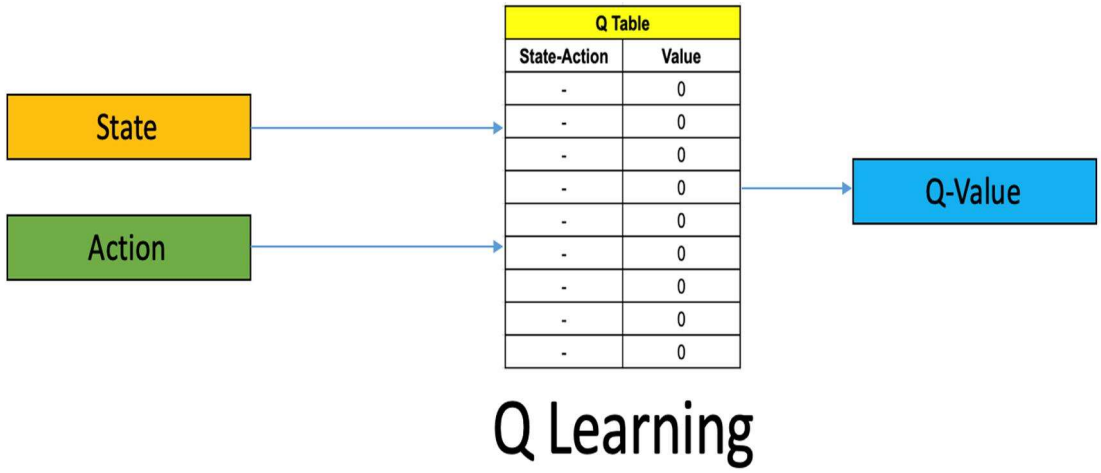


Fig 2.4: Q learning (reinforcement learning algorithm architecture)

State → user search query keywords

Actions → possible best applications

Output → Application name and link with highest Qvalue

Then an android application is designed for user interaction in which the user has to first register in the application using his email already logged in phone and then login into application then the user has to enter a search query for his requirements the system predicts top 5 best applications that meet user requirement.

If the user is already using an application and want to know some more applications with similar features user should enter app name and any keywords system predicts the best application. The application also provides some basic details of application and link to download the application.

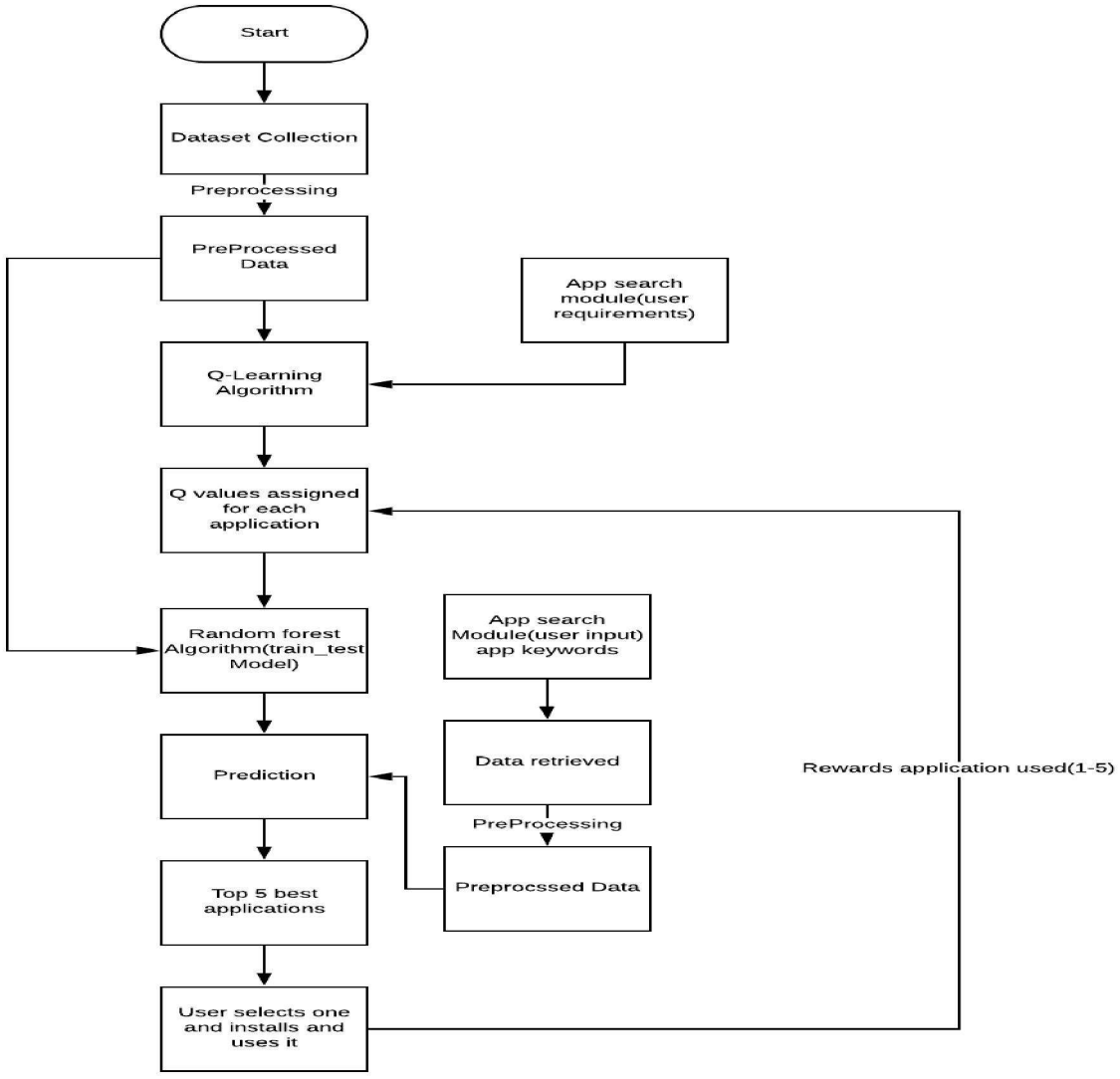


Fig 2.5:Process applied

2.3APPLICATIONS

The sewico application can be used in all android smartphones it can be an inbuilt application in Smartphone:

- Suppose, A user want to use an application for his requirement but from millions of applications from Playstore the user cannot find which application is suitable so user has to download sewico application,
- For example user is already using an application of some kind but is no longer interested in using that application so the user installs sewico and finds out other apps with similar features and downloads it.
- The application is useful to naïve users who do not have prior skills in using applications and Smartphones.
- The application is useful to naive users who do not have prior skills in using applications and Smartphones.

2.4 SUMMARY

Suppose we are buying a product in a store we see whether the product is meeting our requirement or not, same applies in when purchasing a land or any other services we do prior enquiry whether it meets our requirement or not and whether it is beneficial to us. If the respective thing does not meet our requirement we do not choose but we choose an Alternative one.

But in case of availing services of android applications we do not check whether it meets requirement or not. So, which results is installing the app and using it then we find that is not useful to us, and there are also many apps that steal data of users which are harmful to this problem we come up with a solution and that is “SeWiCo”. This project helps to predict dynamically a better application that could serve the user requirements based on the past reviews, ratings, resource consumption, genres, content rating etc.,

CHAPTER 3

REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

- Windows 7, Windows 8, windows 10 for building android application.
- Android Studio
- Android 4.22 or above API 17
- Python version 3.6x
- Jupiter, Spyder tools

3.2 HARDWARE REQUIREMENTS

- Processor (CPU) with 2 gigahertz (GHz) frequency or above.
- A minimum of 2 GB of RAM.
- Monitor Resolution 1024 X 768 or higher.
- A minimum of 20 GB of available space on the hard disk. Internet Connection Broadband (high-speed) Internet connection with a speed of 4 Mbps or higher.
- Phone screen resolution 1280 X 800 or higher.

CHAPTER 4
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

An Android Application is developed in which the user initially is new user so registers in the app by entering credentials username, email, password and they are stored in firebase server. Then user is given access to login into the app.

Then user can enter his search key word in the app and the user input is sent to server and processed and it returns all possible suggestions then user will be able to enter his requirements.

Based on user requirements and key search a app is predicted and given to download and after user uses app will be given option to give feedback so that at next instance the app recommendation the new best output is predicted.

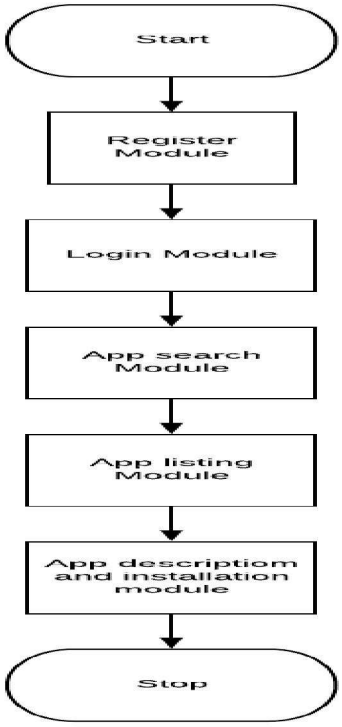


Fig4.1: Android application Interface block diagram

When the new user installs the application the user has to register in the app with his credentials username, password, email they are stored to firebase database server.

Then the user is given access to login, the user enters email and password the firebase server validates and gives user access and it redirects to app module

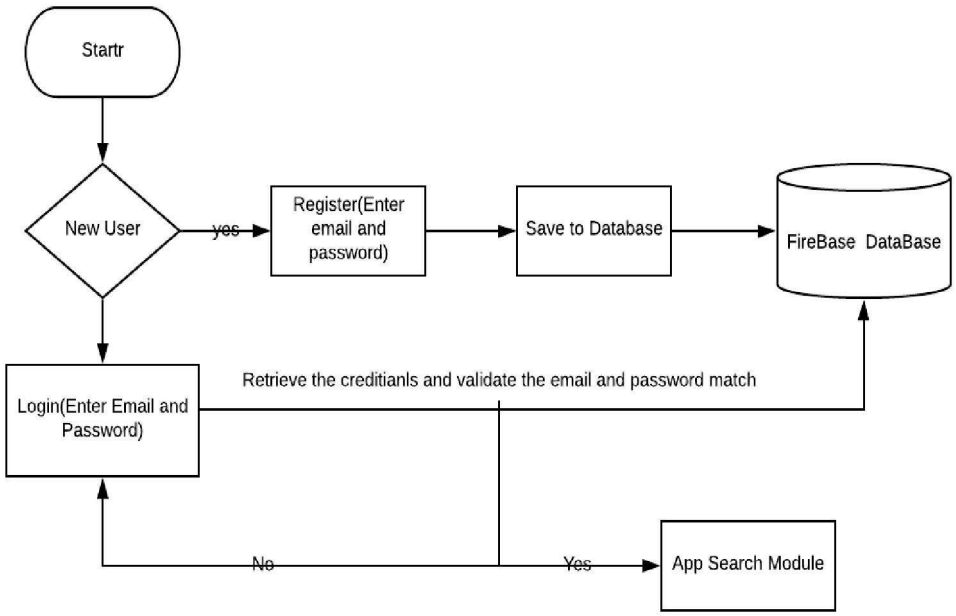


Fig 4.2 : Login and register block diagram

In the App search module the user enters a search keyword and the user input is stored in firebase and it is sent to backend python code the python code returns the app suggestions to the app through firebase, Flask Framework is used to run python backend code.

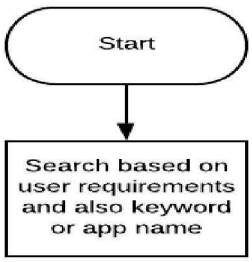


Fig 4.3: App search module block diagram

Then the user is allowed to enter the user requirements, the user requirements are sent to server and the data is processed and algorithm is applied and output is predicted.

The algorithm is developed and model is built and trained and the model continuously learns on the user feedback each time the user gives an feedback.

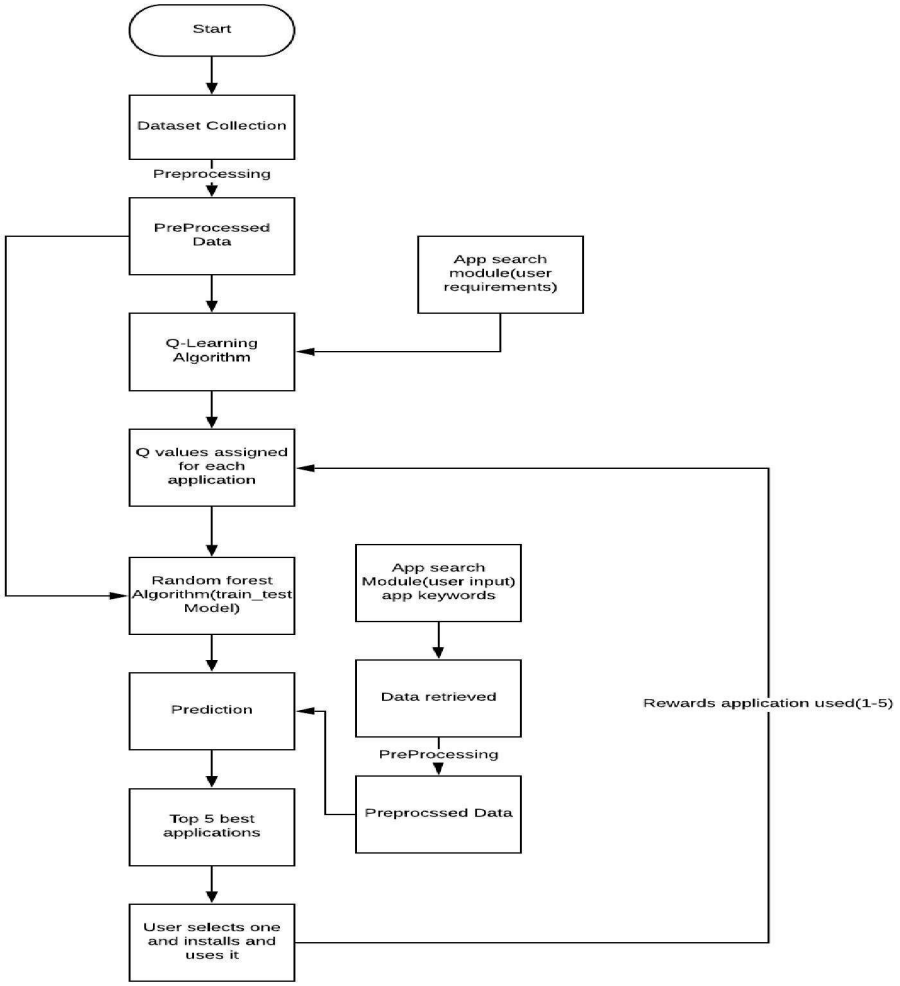


Fig 4.3: Algorithm block diagram

The dataset is collected using play store website using concept called web scrapping, 62 categories of apps with 26 features are collected

Fig 4.4: Web scrapping block diagram

The data is preprocessed and based on user requirements provided by the user the similarities between every feature of apps collected is calculated and algorithm is applied , random forest is applied to built the model and predict the value.

Then based on user feedback q learning is applied to predict the more accurate value and app with highest q value is given as output and link is given to download the app.

Each time the user gives a feedback the q learning is applied and it learns again so model learns in each case dynamically for the user requirements and feedback after using app so that the better recommendations are given.

Random forest algorithm is applied to predict the q value for given user search and app is predicted.

Then based on users feedback q learning is applied that makes the model learn again and predicts new output.

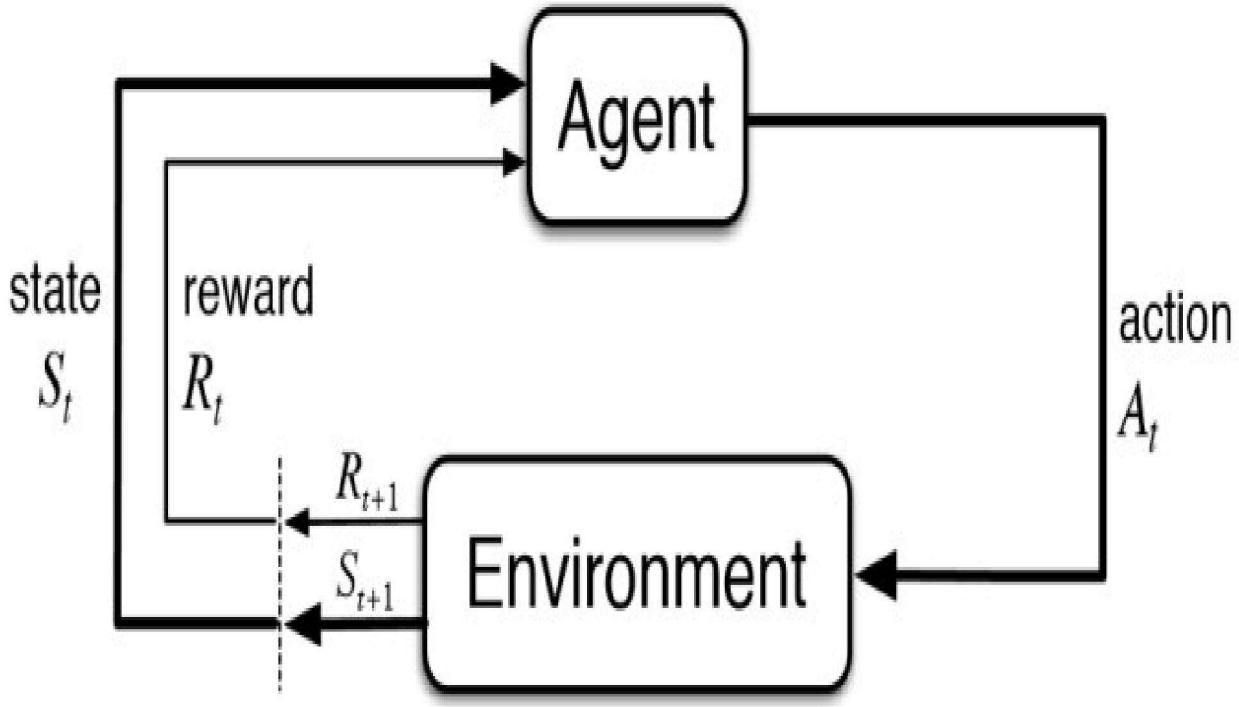


Fig 4.5: Q learning algorithm block diagram

Equation 4.1:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

CHAPTER 5
IMPLEMENTATION

5.1 Environmental setup

The android studio is used for application development the code is written in java and xml for user interface.

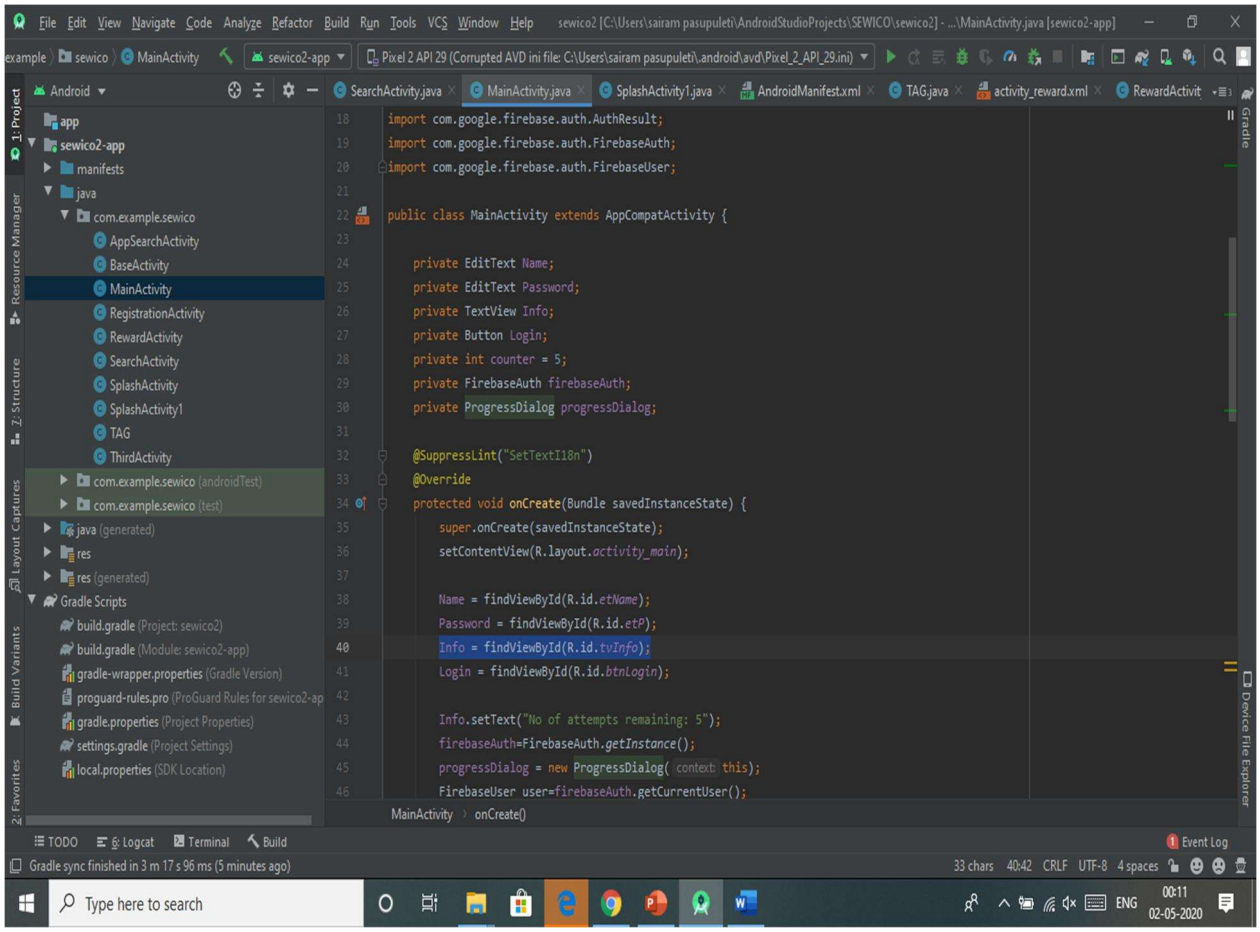


Fig 5.1: android studio setup

Firebase is used as backend server for user authentication and user registration and also for taking request from android application and sending to python backend code written in flask framework and return response back to app.

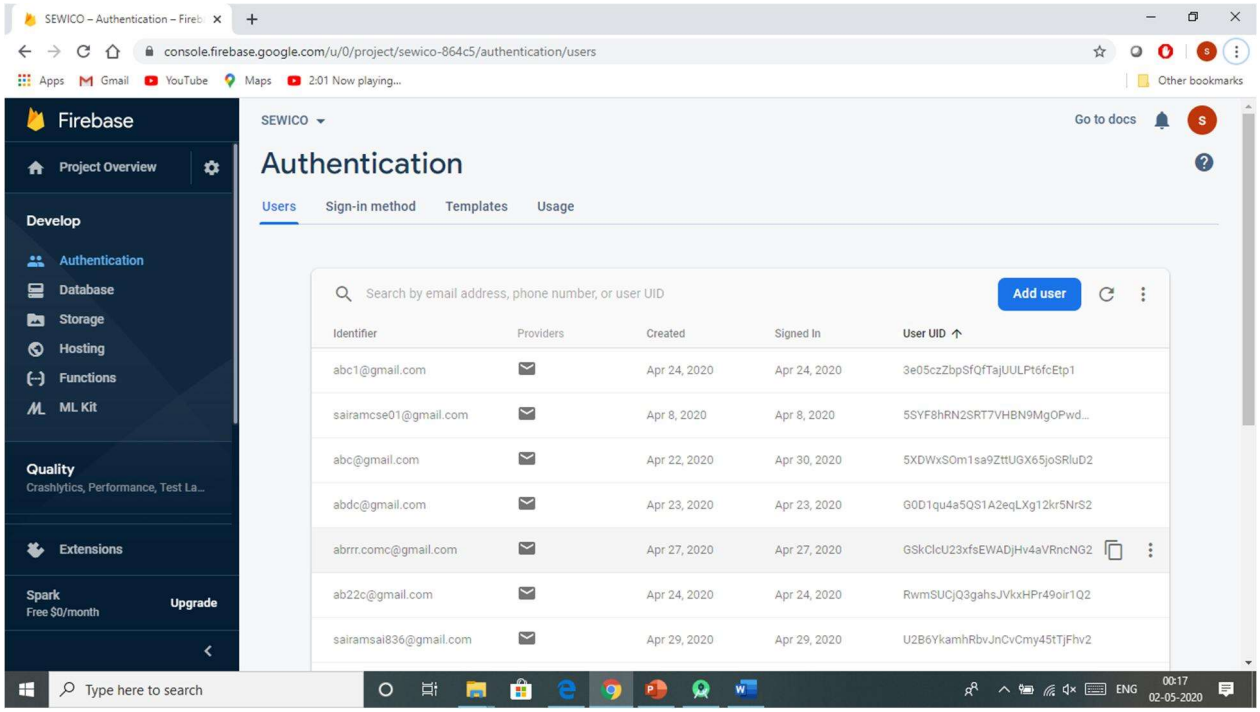


Fig 5.2: Firebase server providing user registration and authentication

Then the firestore a database of firebase server is used that acts as medium between the python code written in flask framework and android app for accepting responses from android app sending them to server and return response back to app.

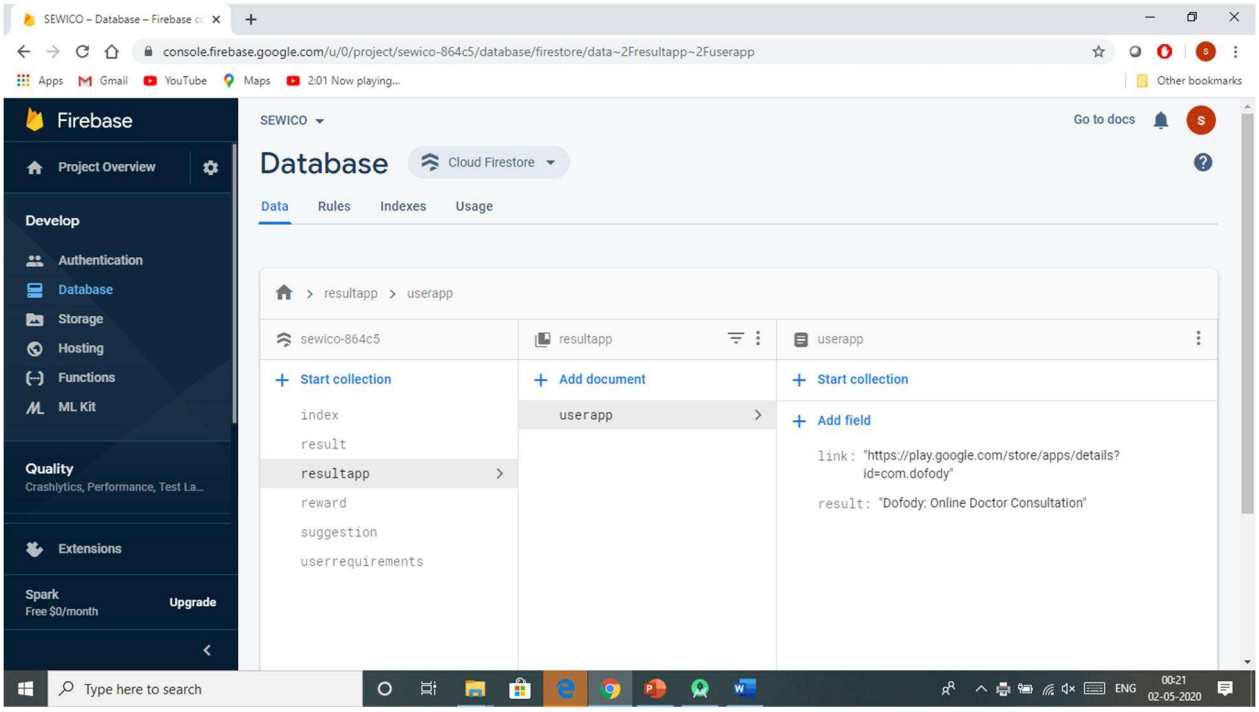


Fig 5.4: Firestore Databse storage fields

The firestore database is used as it consumes low storage and is also fast and it also used to accept request and give response efficiently.

The algorithm and backend code is written in python(python-3.7) and multiple libraries are used.

Flask framework is used to create the server which takes python backend code and imports in it so that whenever there is a request it is handled and response is returned.

Web scrapping is done and the code is written in python to collect dataset.

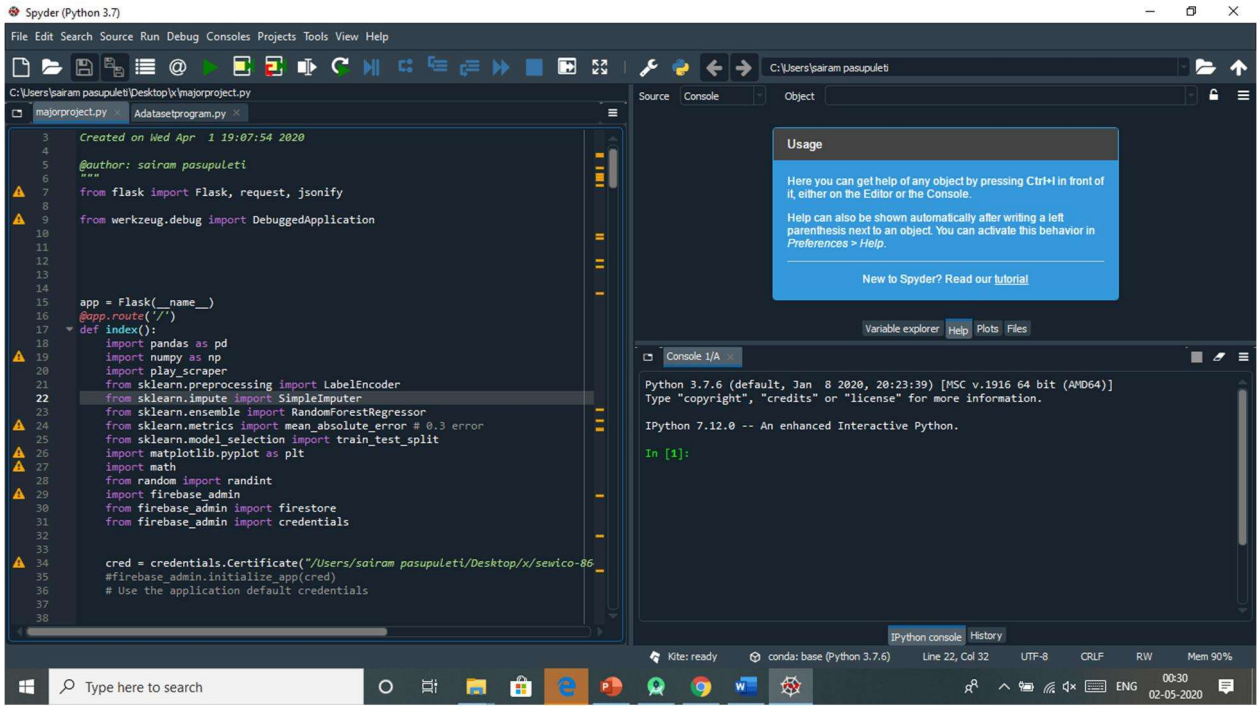


Fig 5.5: Python interface to write python backend code

The python backend code which is embedded into flask server which runs on my local ip server the process gets started.

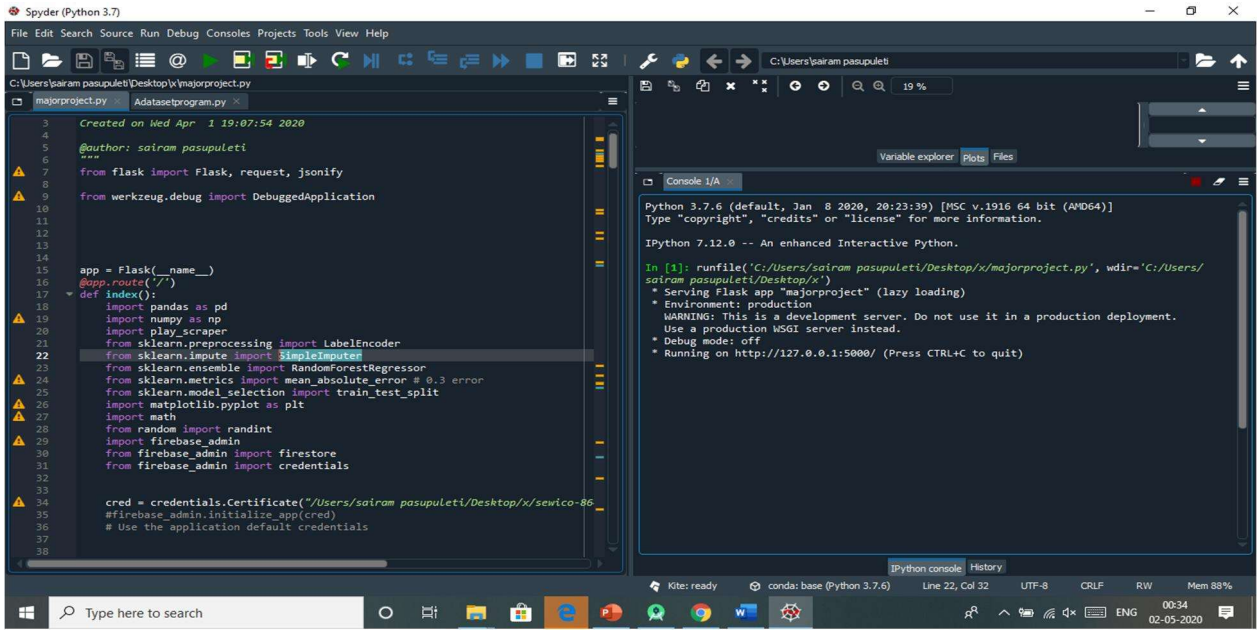


Fig 5.7: Flask server started

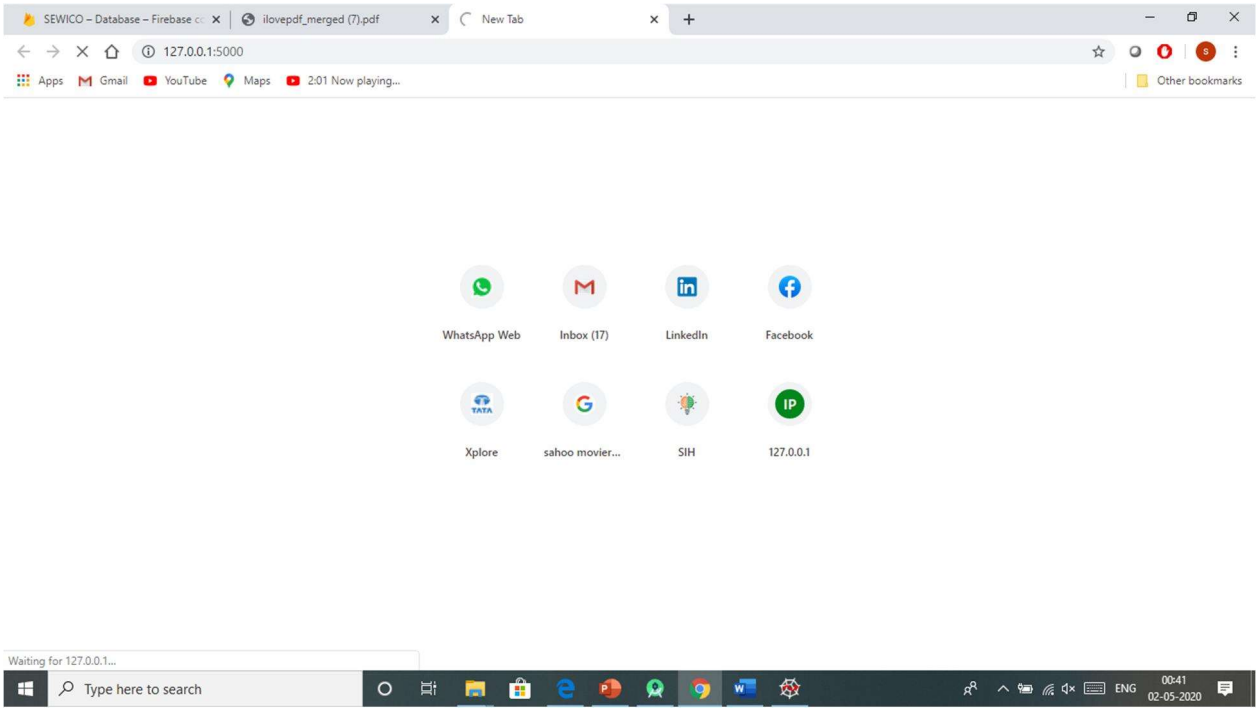


Fig 5.8: Server started in local IP Adress

5.2 Module Description

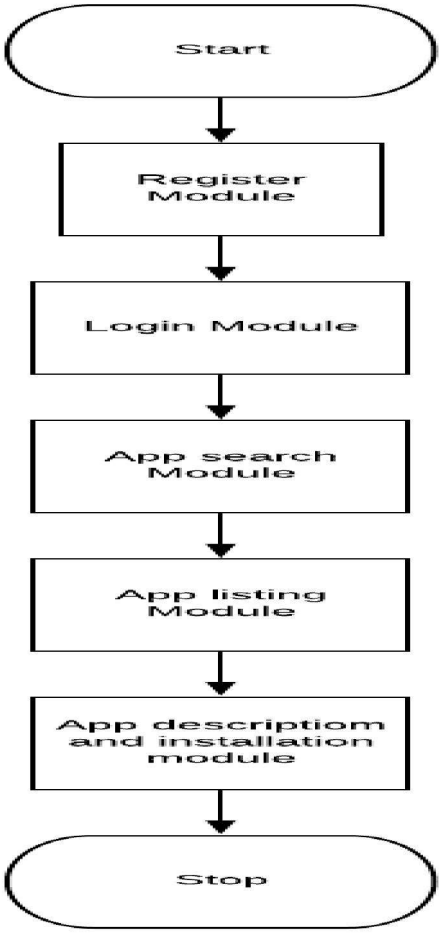


Fig 5.9: Modules flowchart

When a user installs an application the user will be new user so the user has to register in the application, then user is given access to login and is authenticated and given access to app and the app search module and user requirement module and app listing module is used and the user can logout if his work is completed.

Splash Module:

When user installs and opens the application the splash activity is displayed it shows information/logo or any options regarding the application.

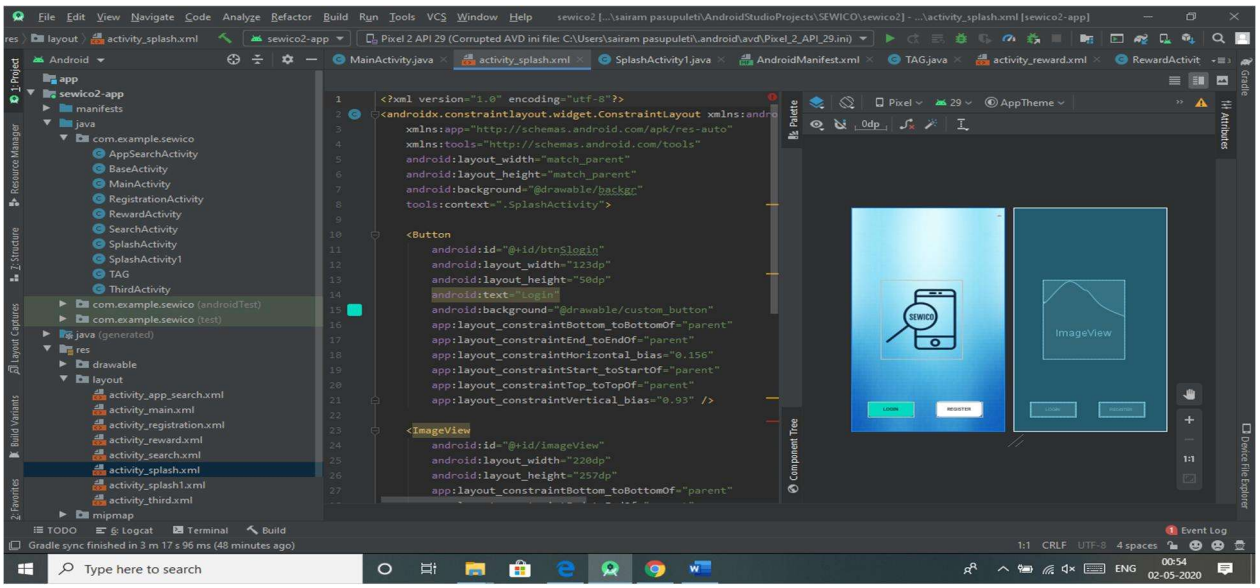


Fig 5.10: SplashActivity.xml

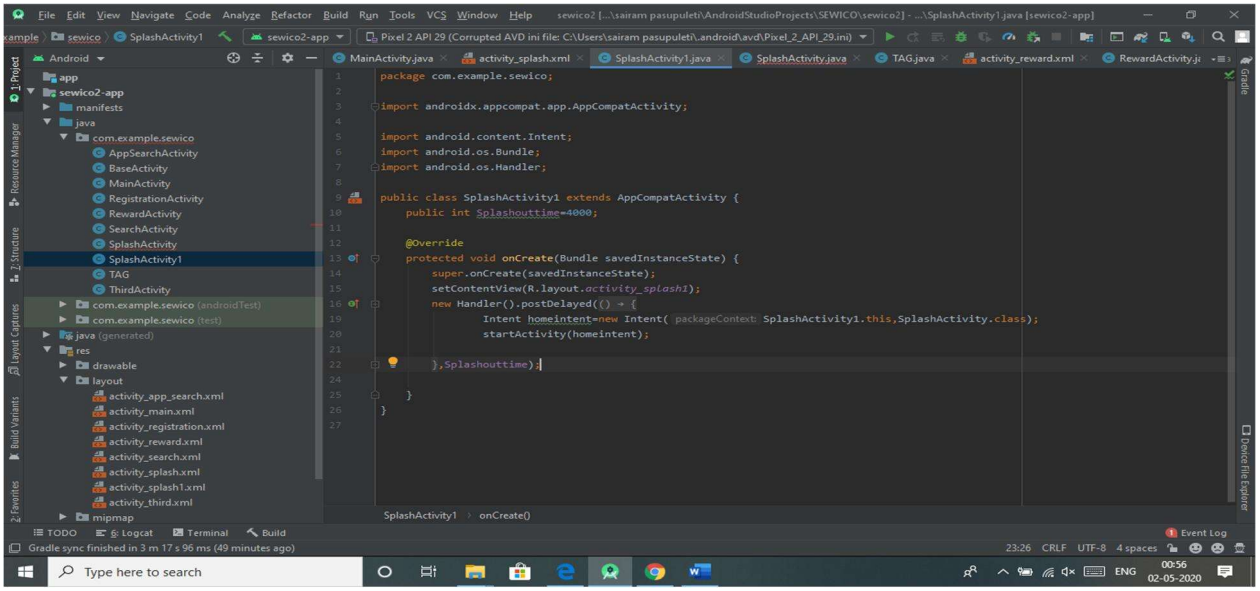


Fig 5.11 SplashActivity.java

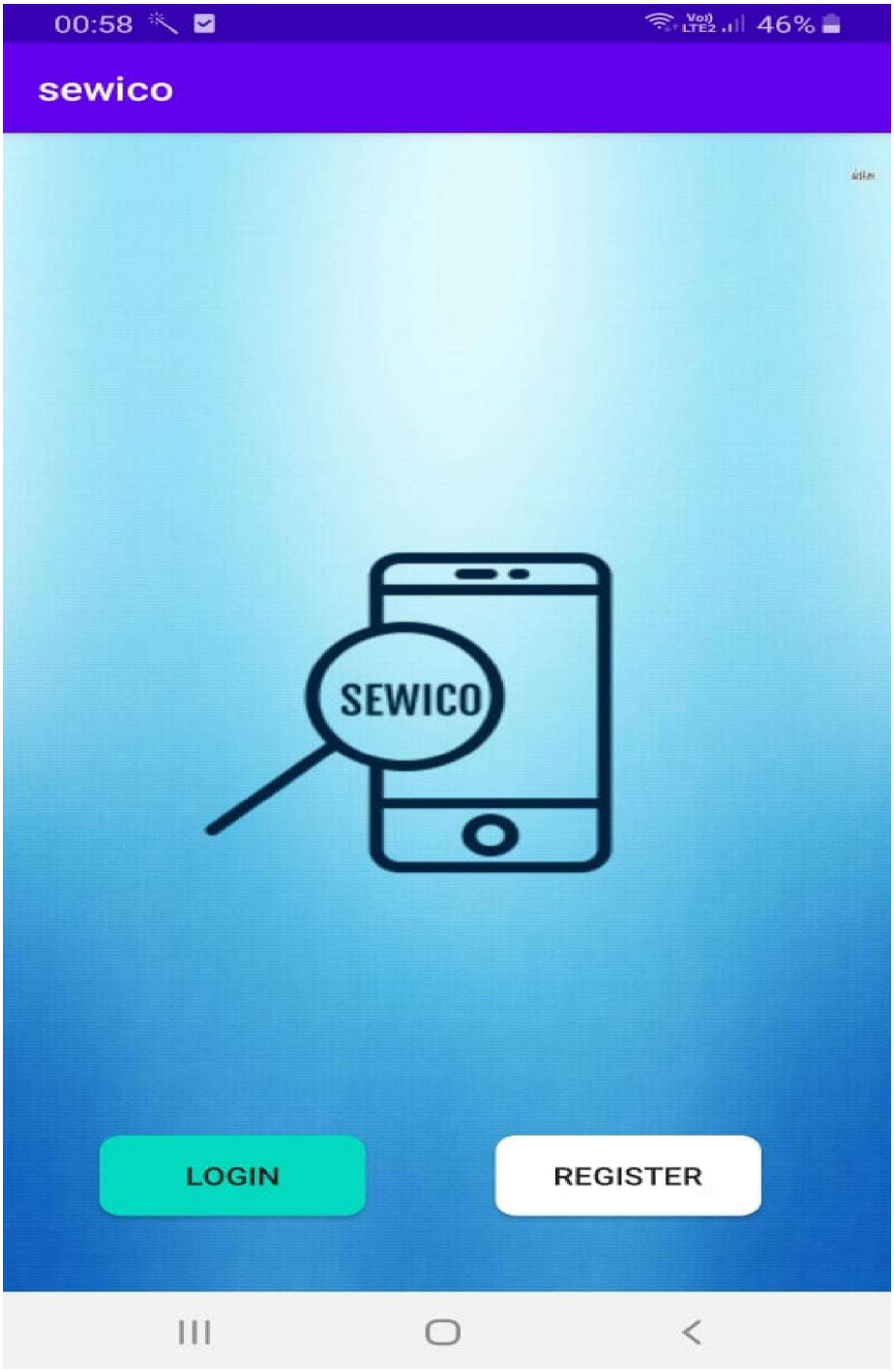


Fig 5.12 Splashscreen

As the new user installs the app clicks register button and registers in the application.

When user clicks register button it redirects to Register module if the user is already registered clicks login button and redirects to login module.

Register Module:

The user has to register in the app by entering credentials like username, email, password and they are sent to firebase server and validated and then user is given access to login.

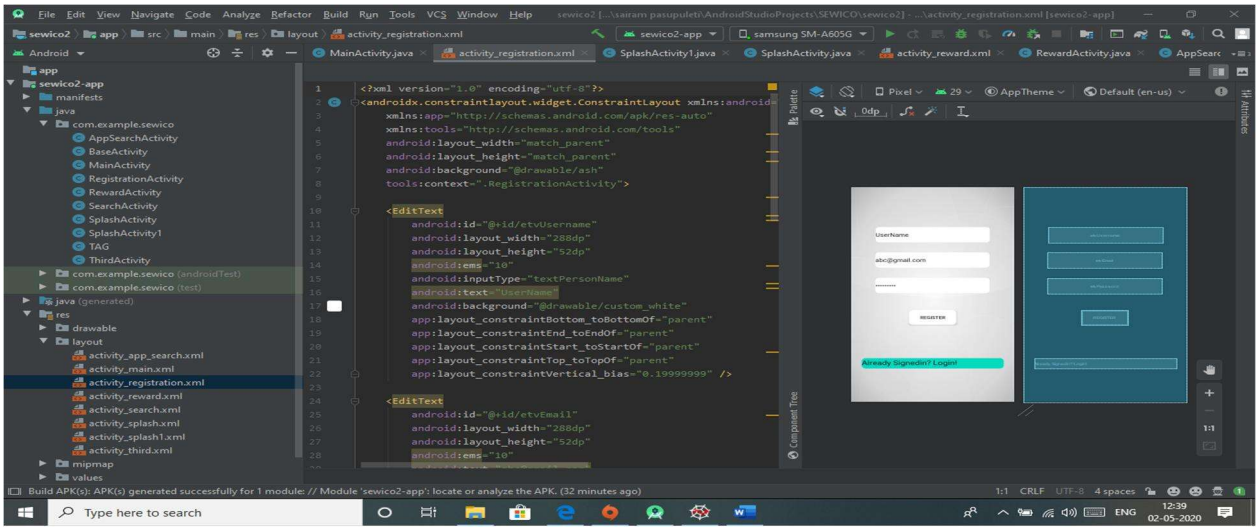


Fig 5.16: RegisterActivity.xml

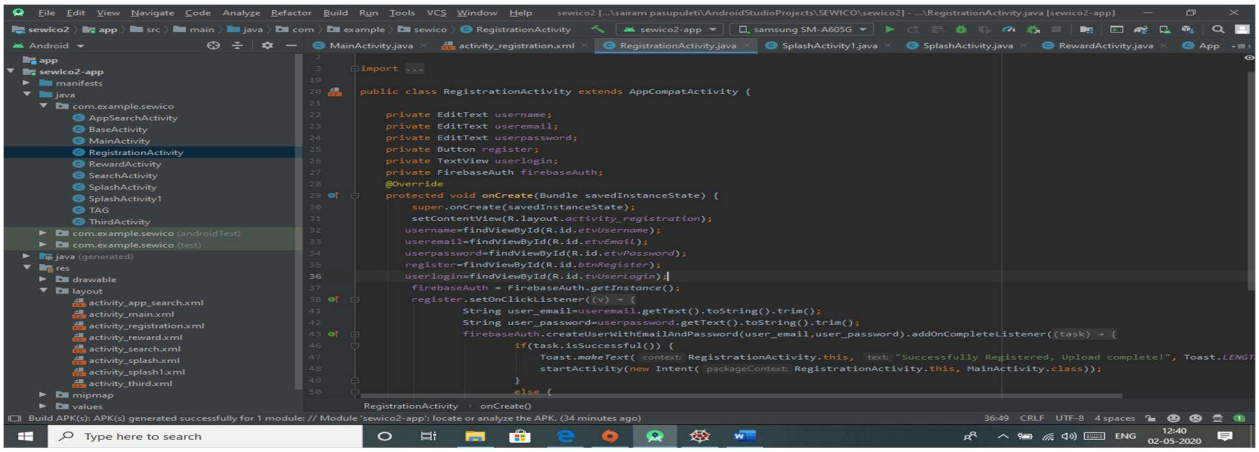


Fig 5.17 RegistrationActvity.java

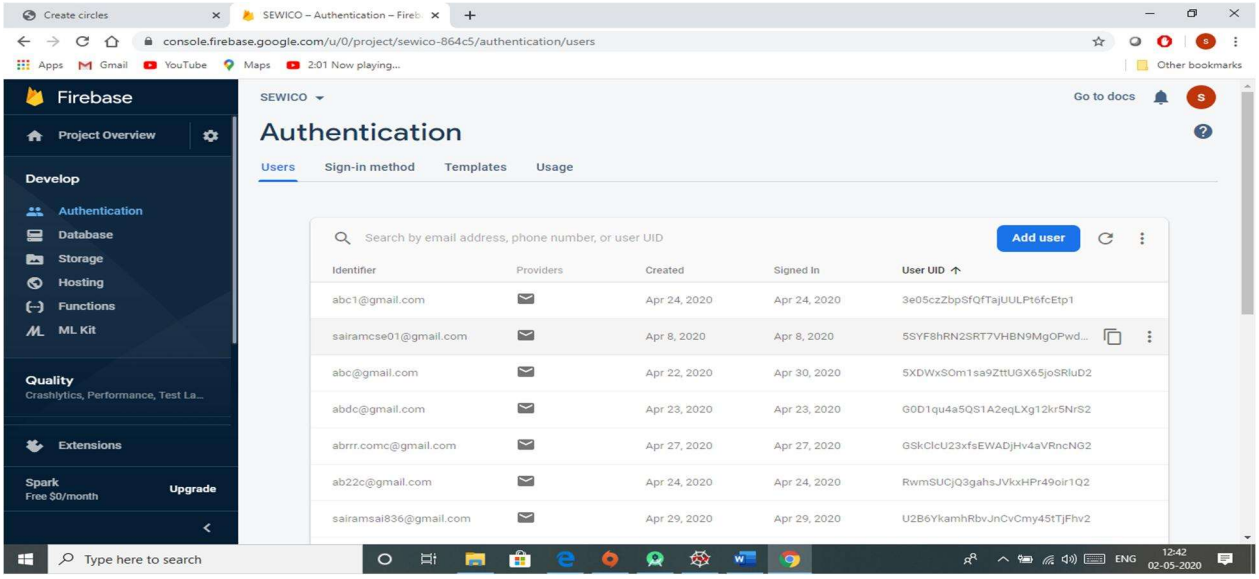


Fig 5.18 Users Credentials stored in firebase server for validation

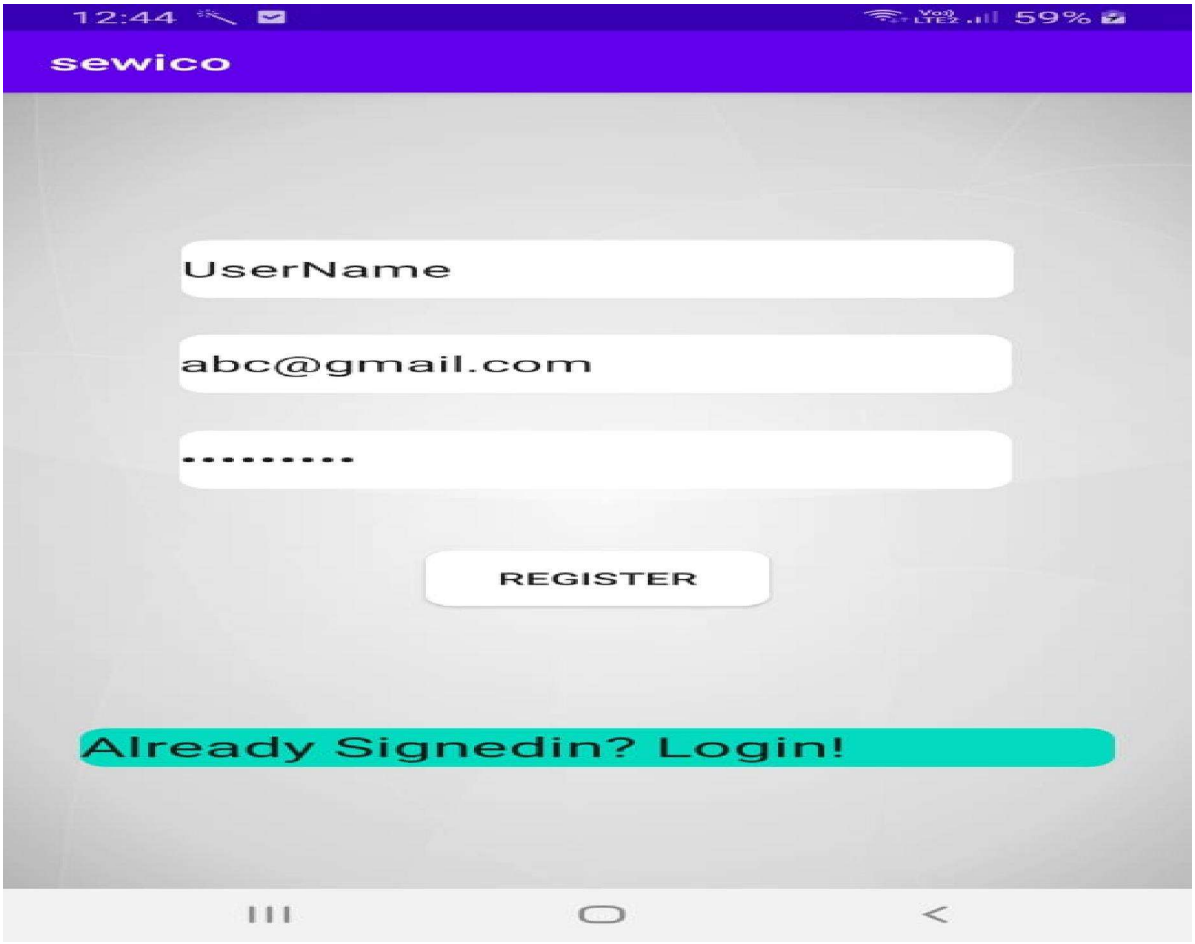


Fig 5.19 Register Screen

Upon successful registration the user is given access to login and if the credentials match then user is redirected to app search module to use the app.

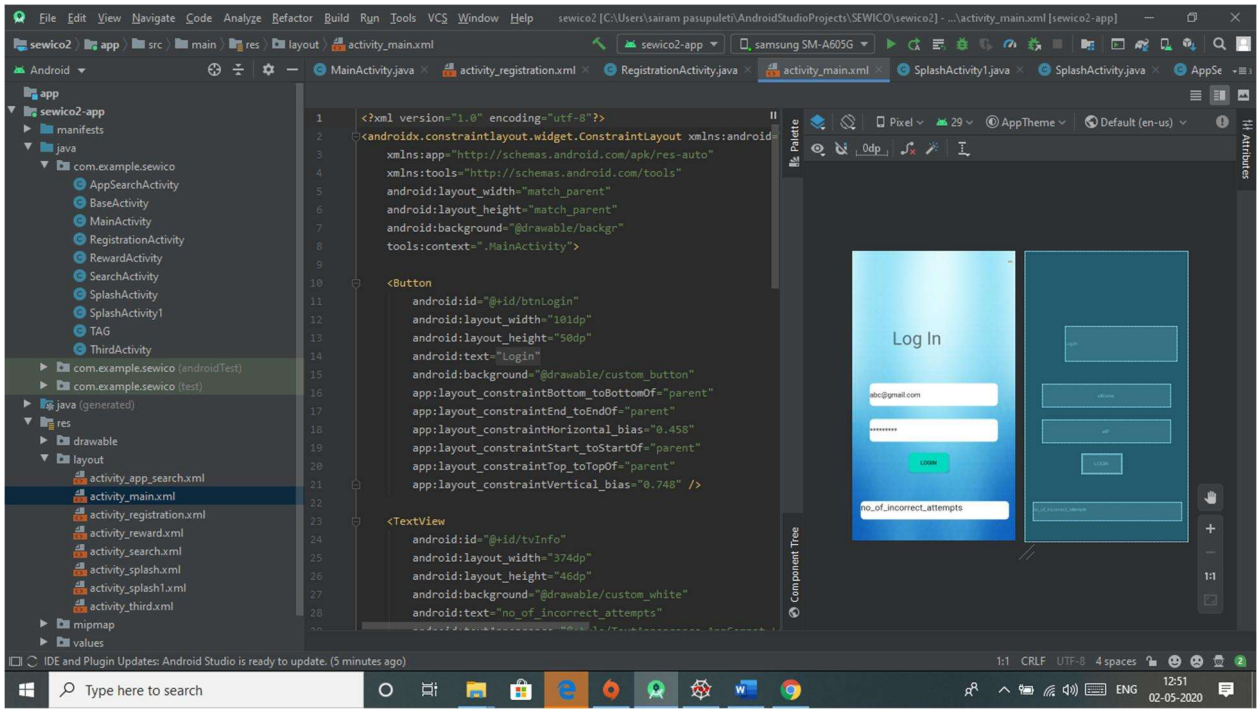
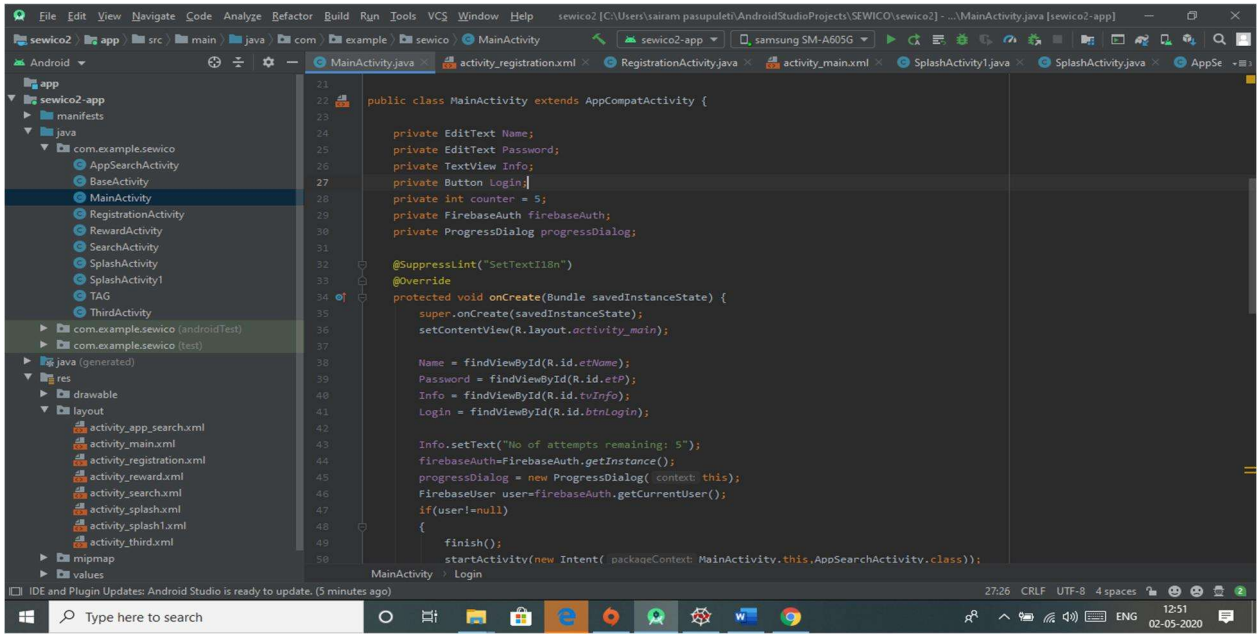


Fig 5.20 LoginActivity.xml



```
public class MainActivity extends AppCompatActivity {

    private EditText Name;
    private EditText Password;
    private TextView Info;
    private Button Login;
    private int counter = 5;
    private FirebaseAuth firebaseAuth;
    private ProgressDialog progressDialog;

    @SuppressLint("SetTextI18n")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Name = findViewById(R.id.etName);
        Password = findViewById(R.id.etP);
        Info = findViewById(R.id.tvInfo);
        Login = findViewById(R.id.btnLogin);

        Info.setText("No of attempts remaining: 5");
        firebaseAuth=FirebaseAuth.getInstance();
        progressDialog = new ProgressDialog( context: this);
        FirebaseUser user=firebaseAuth.getCurrentUser();
        if(user!=null)
        {
            finish();
            startActivity(new Intent( packageContext: MainActivity.this,AppSearchActivity.class));
```

Fig 5.21 Loginactvity.java

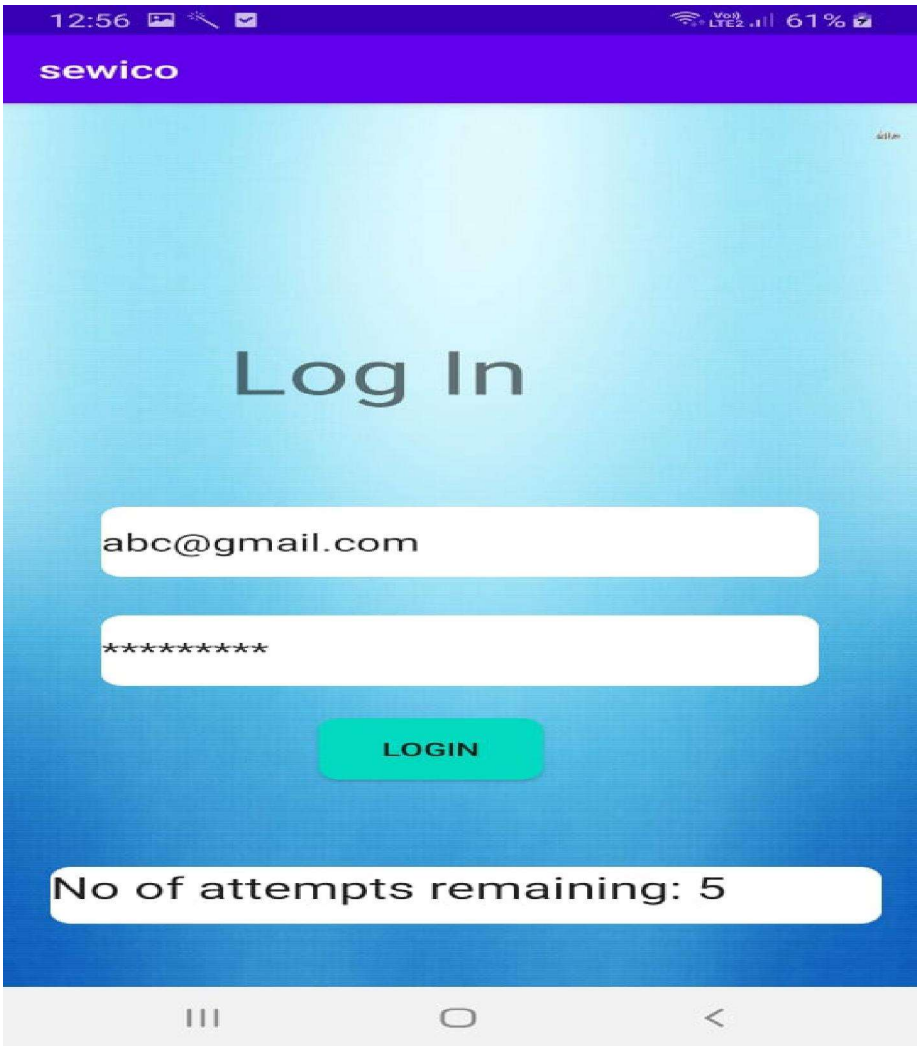


Fig 5.22 : Login Screen

APP SEARCH MODULE:

The user has to enter a key search or any name he knows then the app names related to key search is printed and user selects a name and all the suggestions that are similar to that app and has similar names and similar features are taken into consideration and it is redirected to user requirement module.

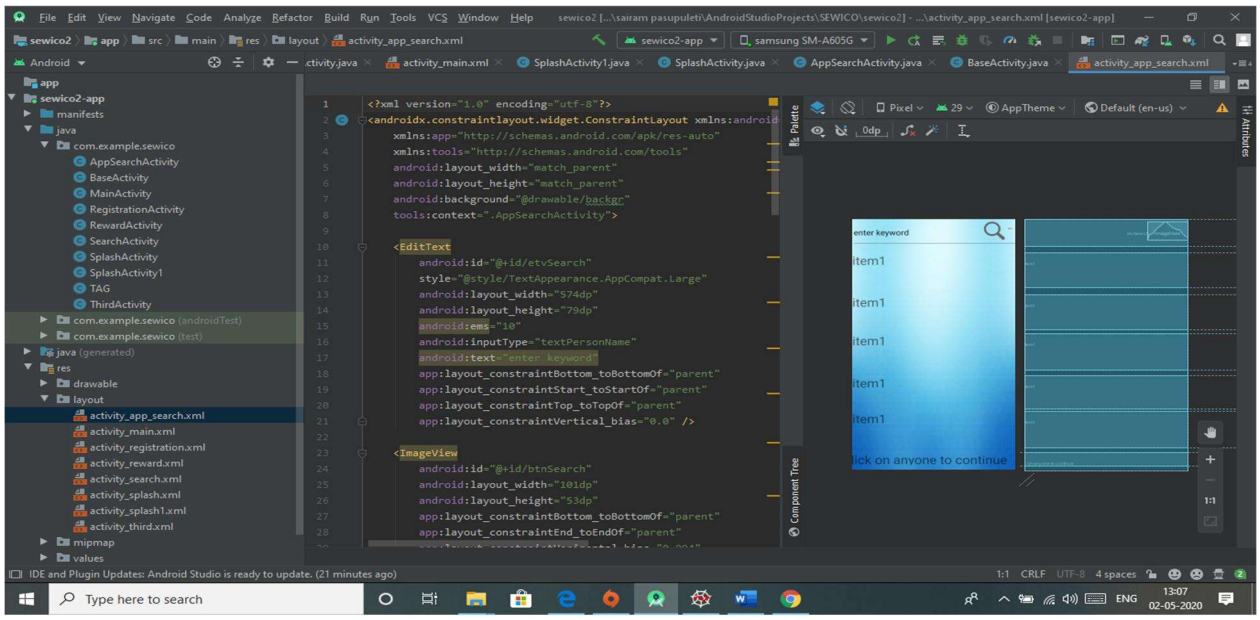


Fig 5.23 AppSearch.xml

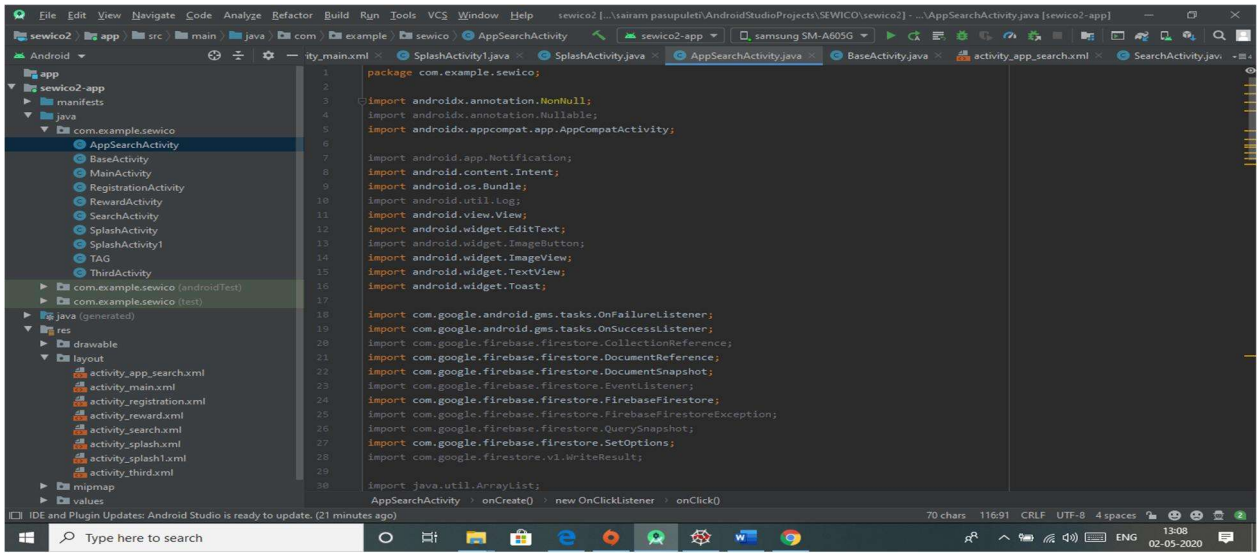


Fig 5.24 AppSearch.java

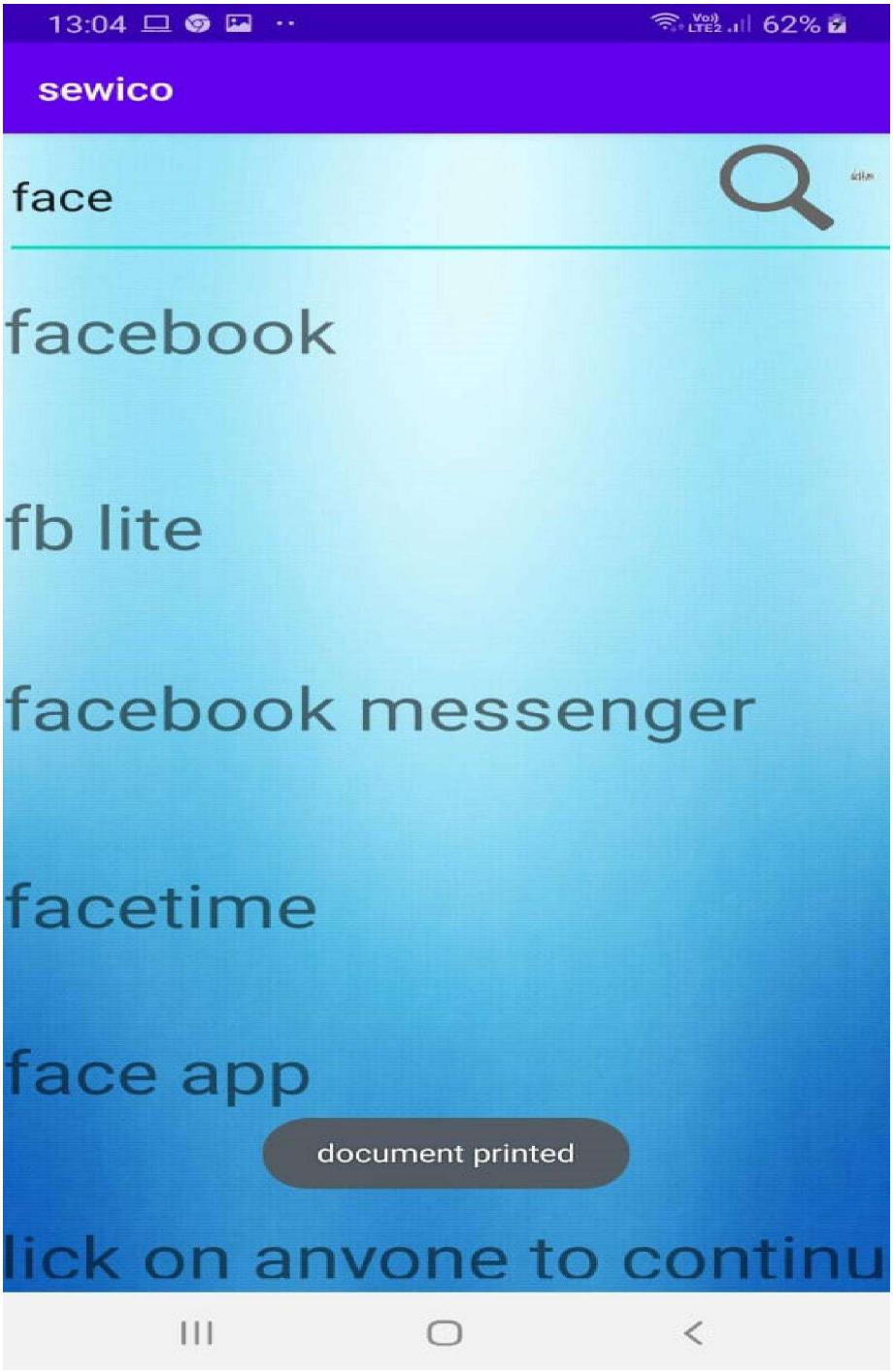


Fig 5.25 AppSearch Screen

User Requirements Module:

In user requirement module the user has to enter the user requirements and they are processes and sent to server. The server applies algorithm and processes the data and is redirected to app list module in which app name and app link to download is given based on user requirements and intrests.

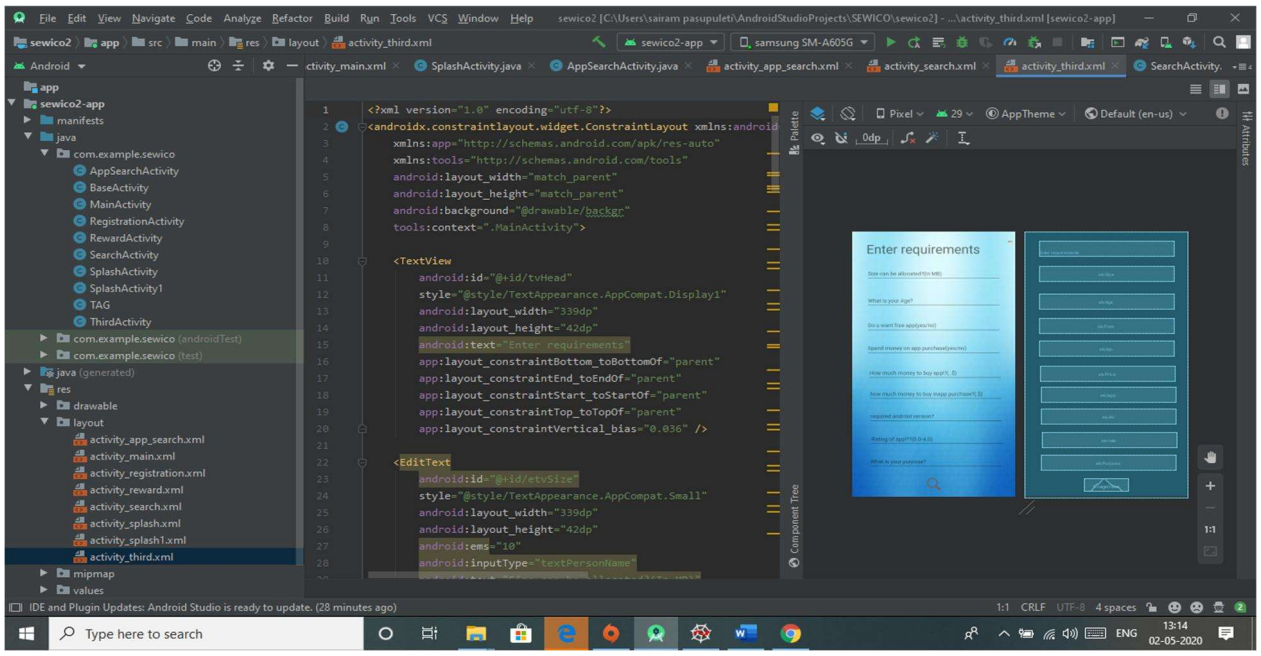


Fig 5.26: User requirements.xml

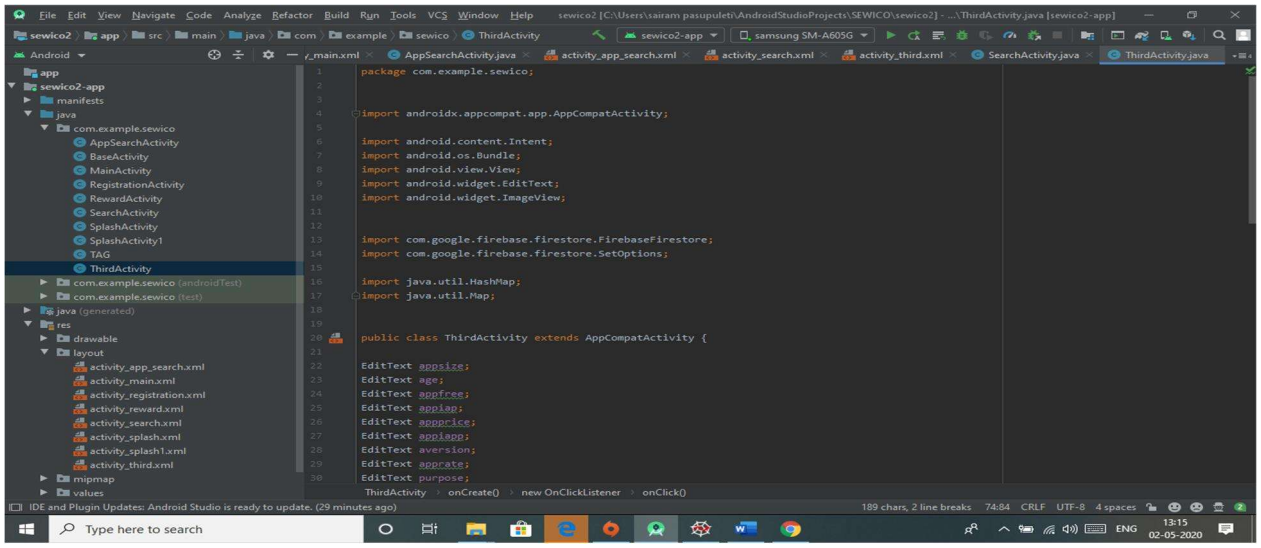


Fig 5.27 UserRequirements.java

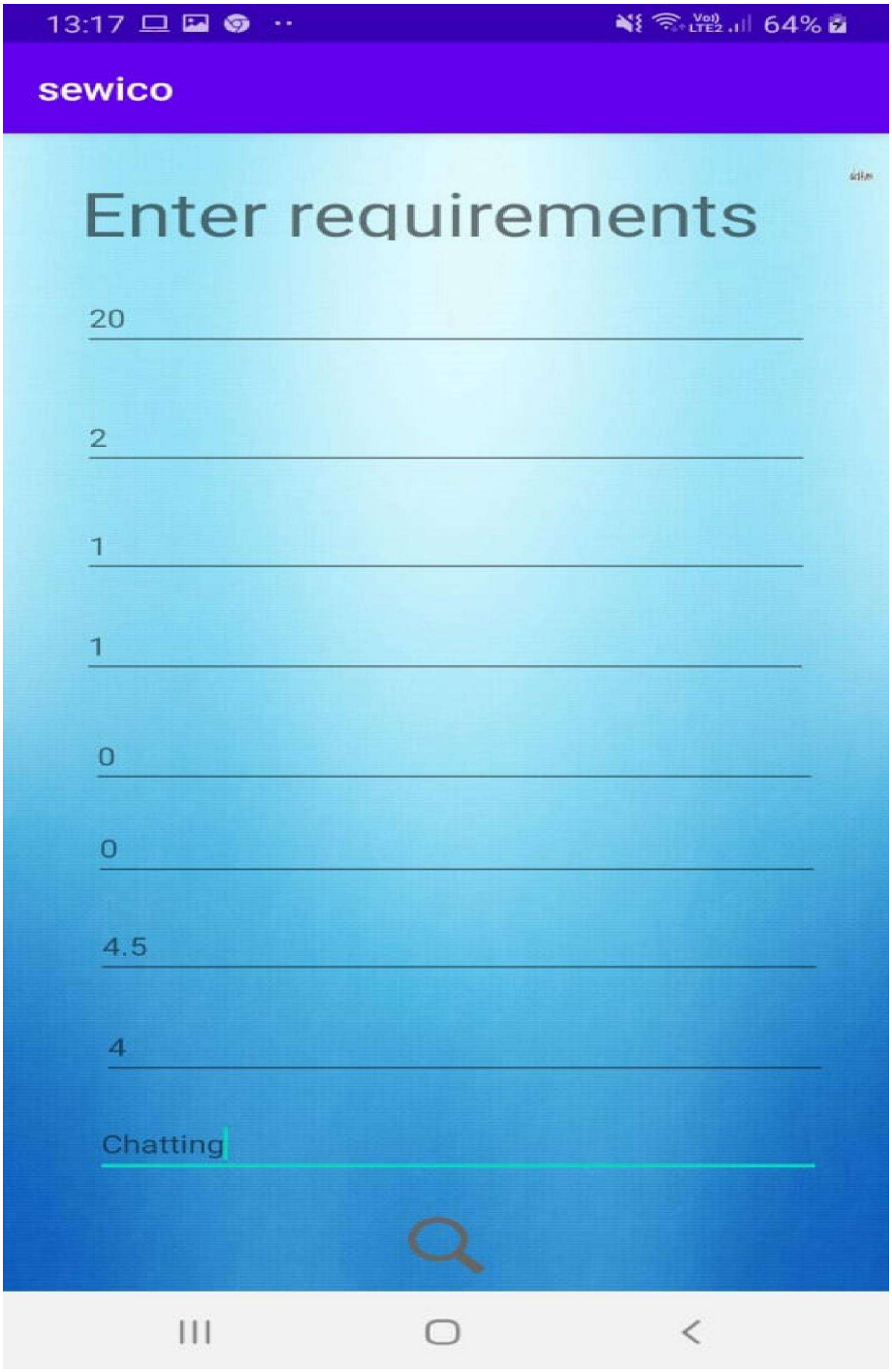


Fig 5.27 User requirements Screen

APP LISTING MODULE:

In this module the app recommendation is given and link to download app is provided in which the user by clicking on the link can install the app.

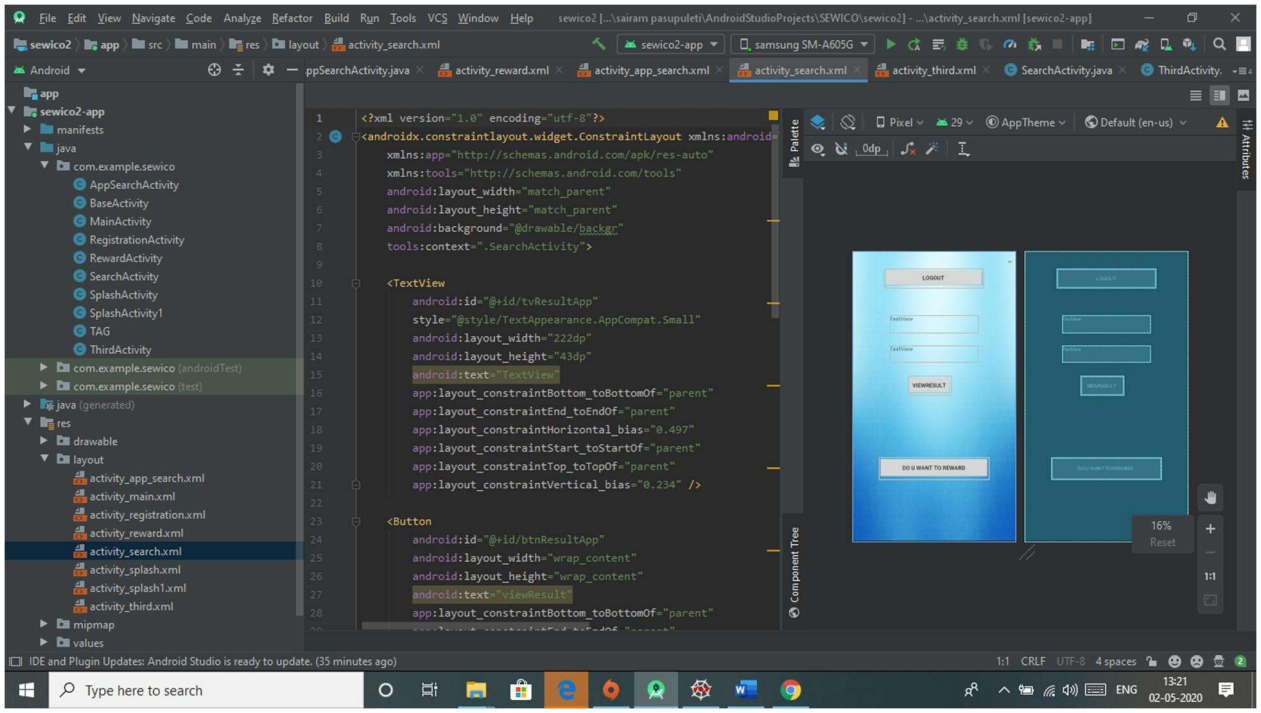


Fig 5.28 App Listing activity.xml

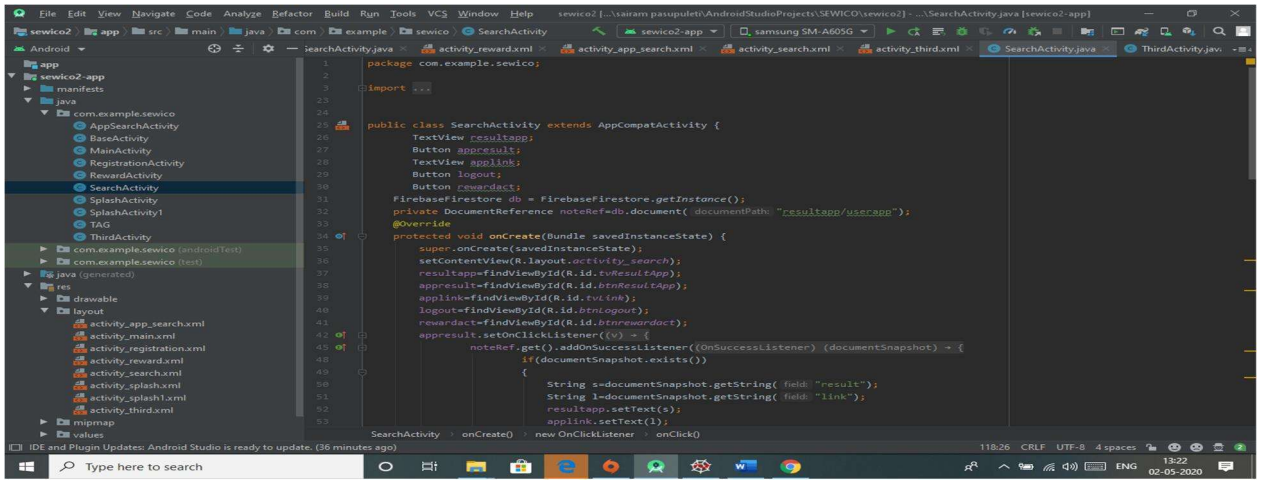


Fig 5.29 APP listing.java

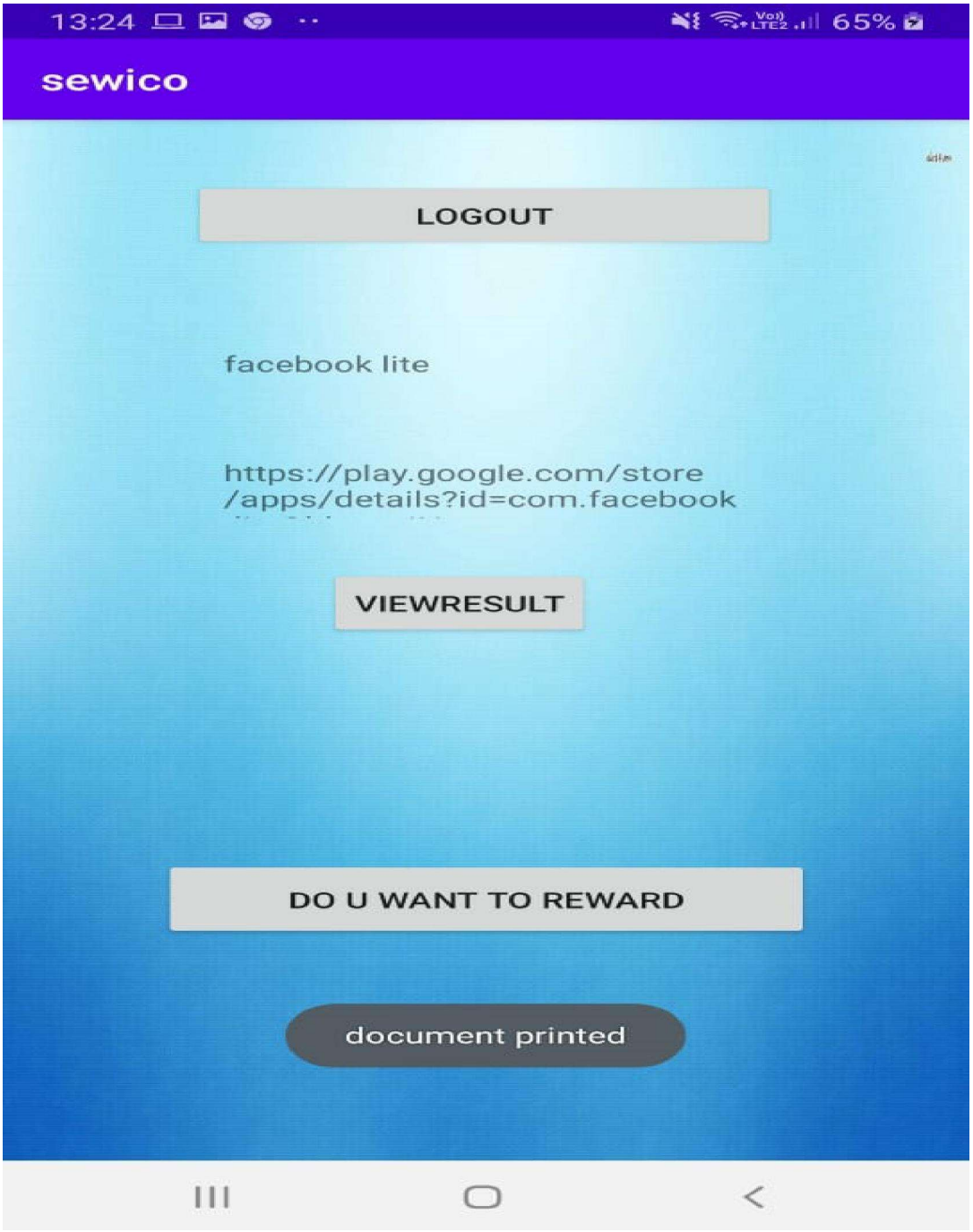


Fig 5.30 App listing Screen

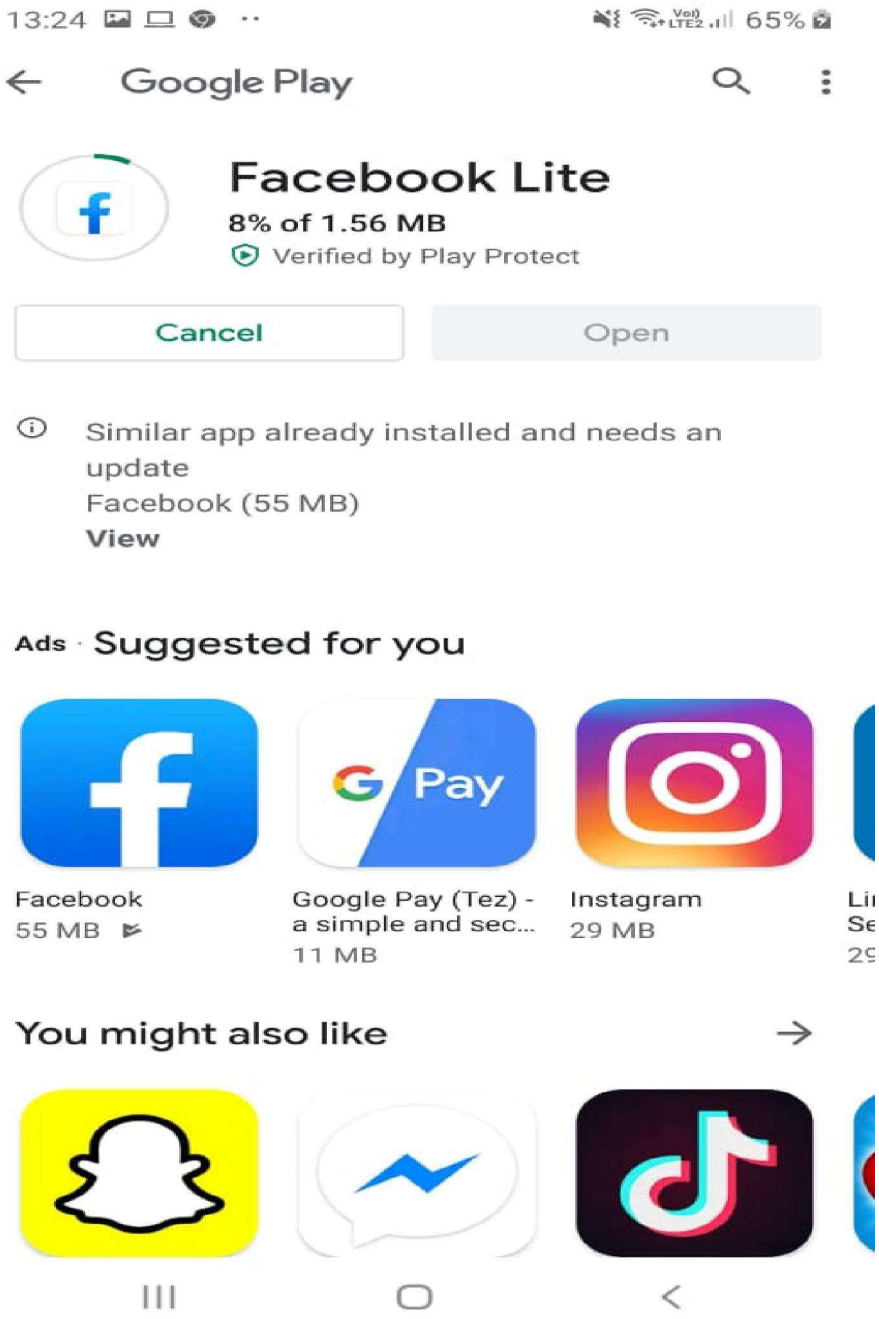


Fig 5.31 App install

REWARD Module:

The user can give reward/feedback to the app based on his experience and usage. The reward will be considered and processes and the app learns again to give more best output next time and user is given functionality to logout from the app.

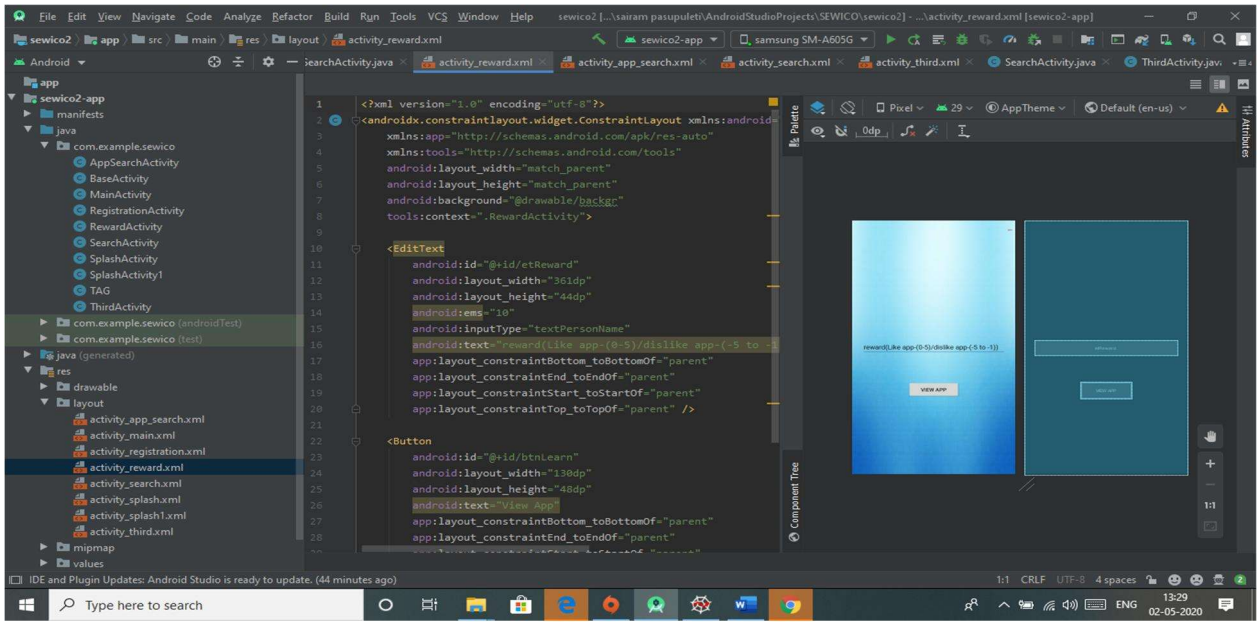


Fig 5.32 RewardActivity.xml

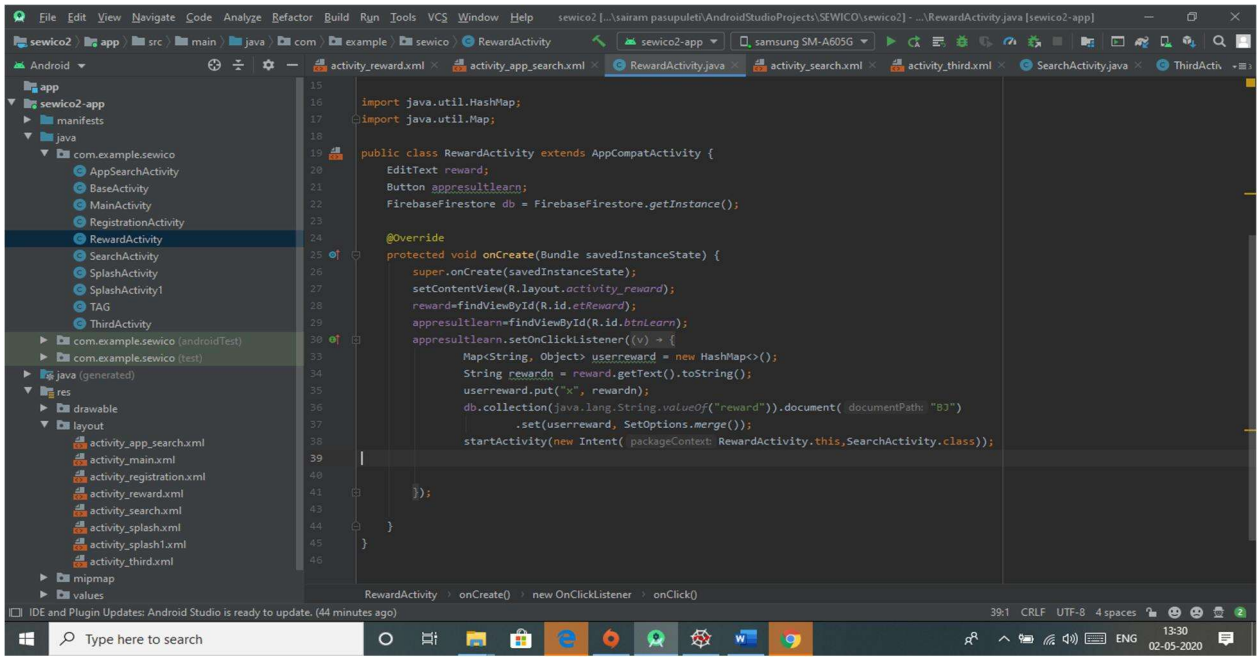


Fig 5.33 RewardActivity.java

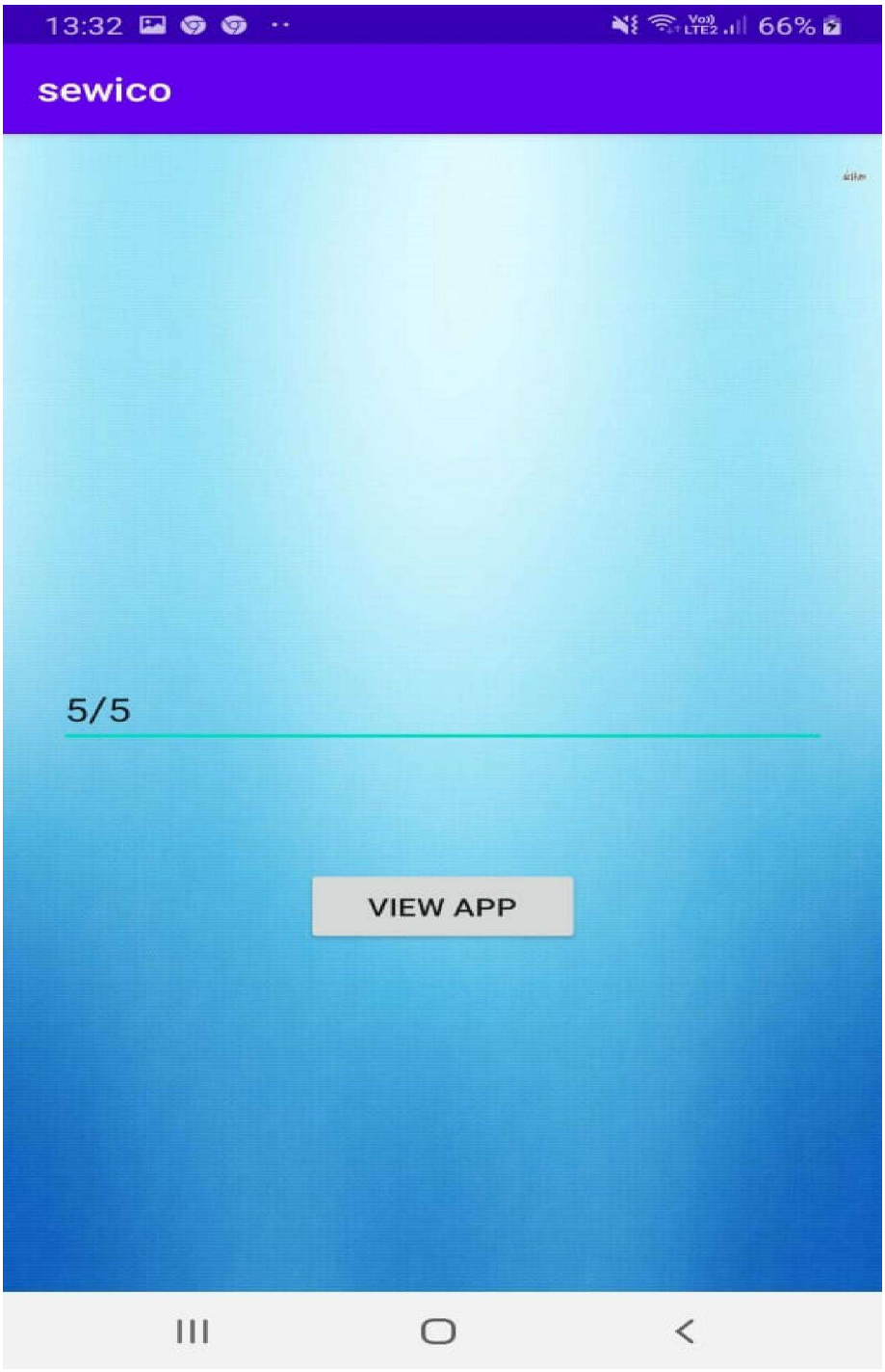


Fig 5.34 Reward Screen

TESTS AND RESULTS

6.1: TEST CASES AND INPUTS

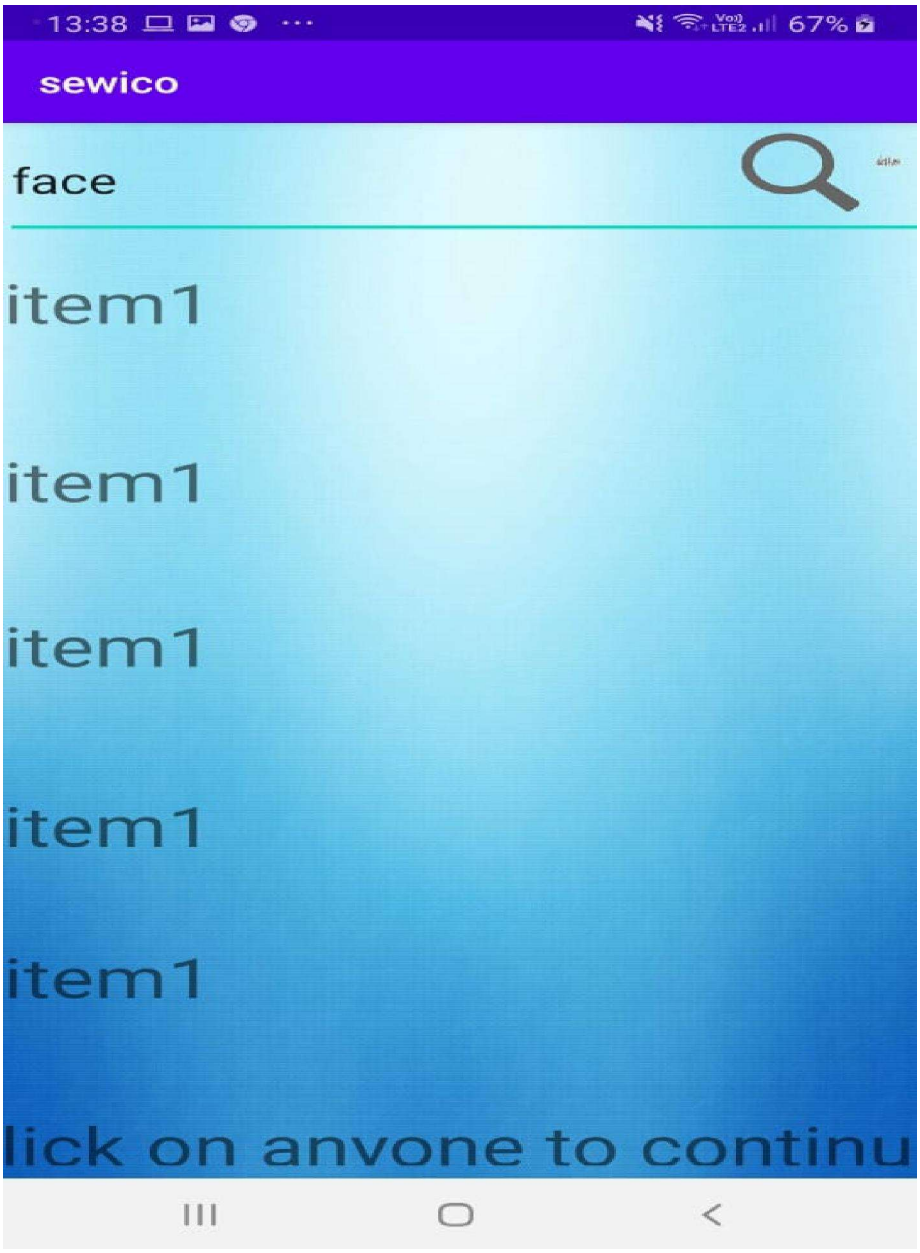


Fig 6.1 App key search

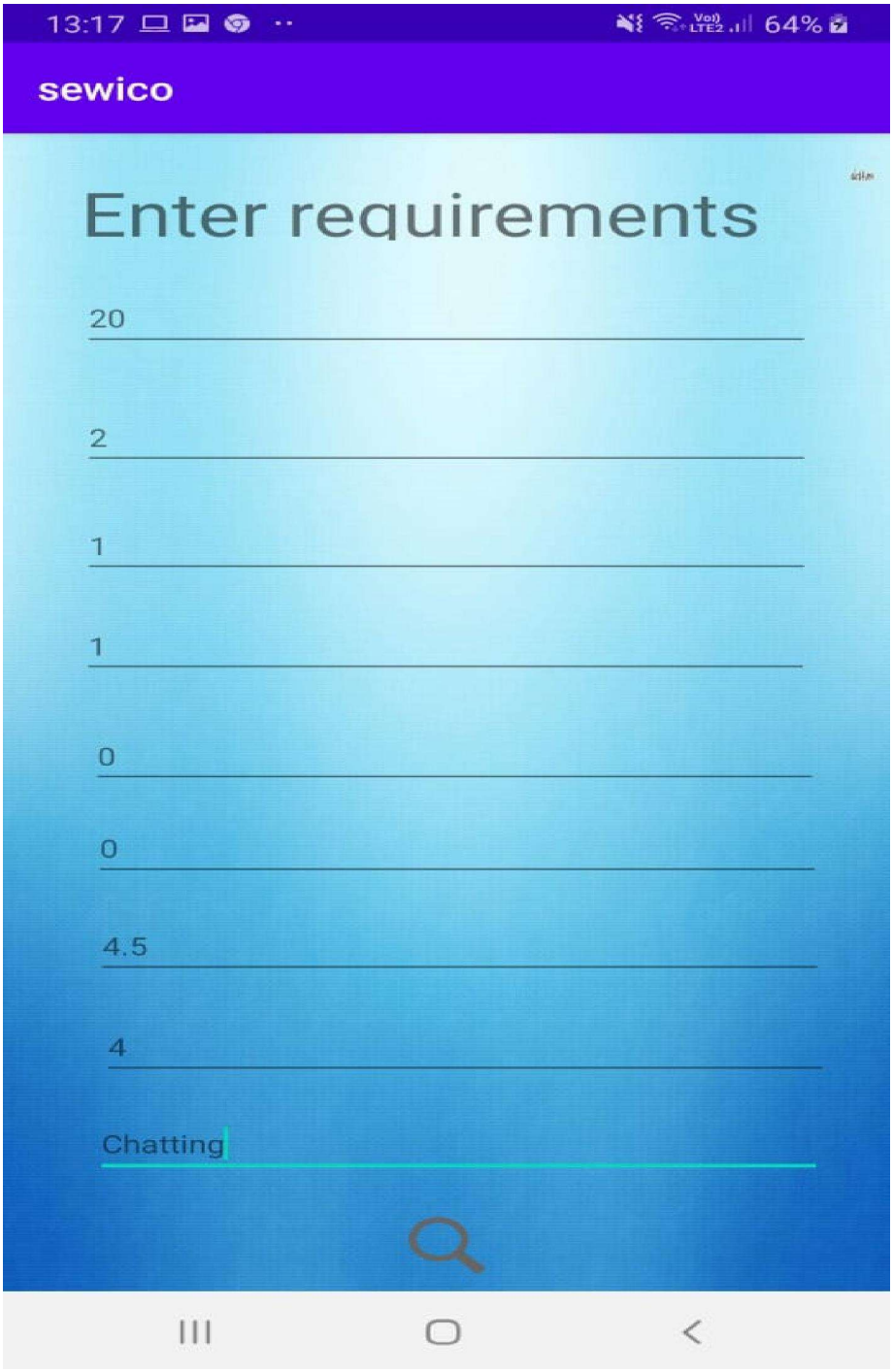


Fig 6.2 User Requirements

[

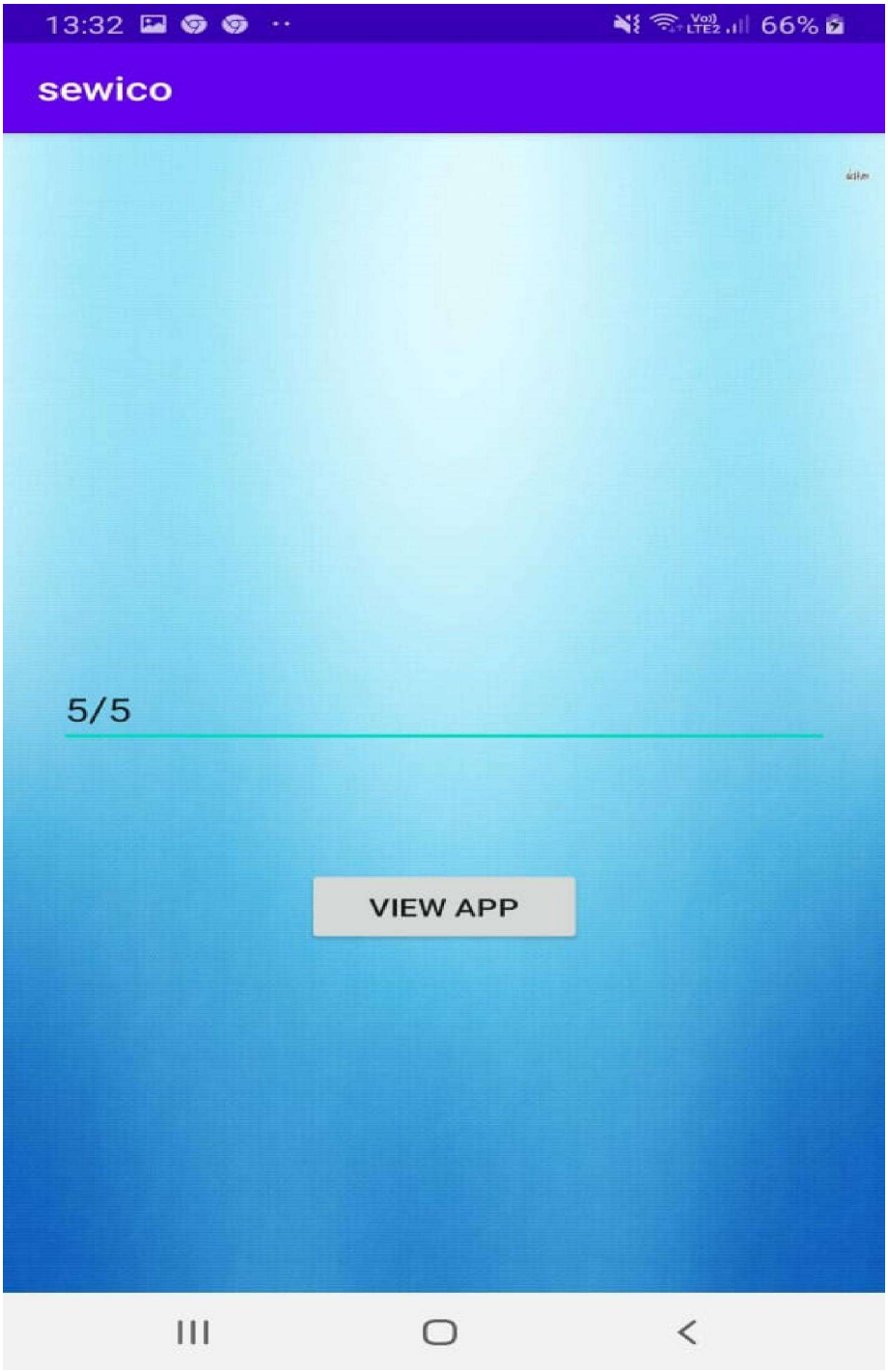


Fig 6.3 Reward and feedback

6.2 RESULTS

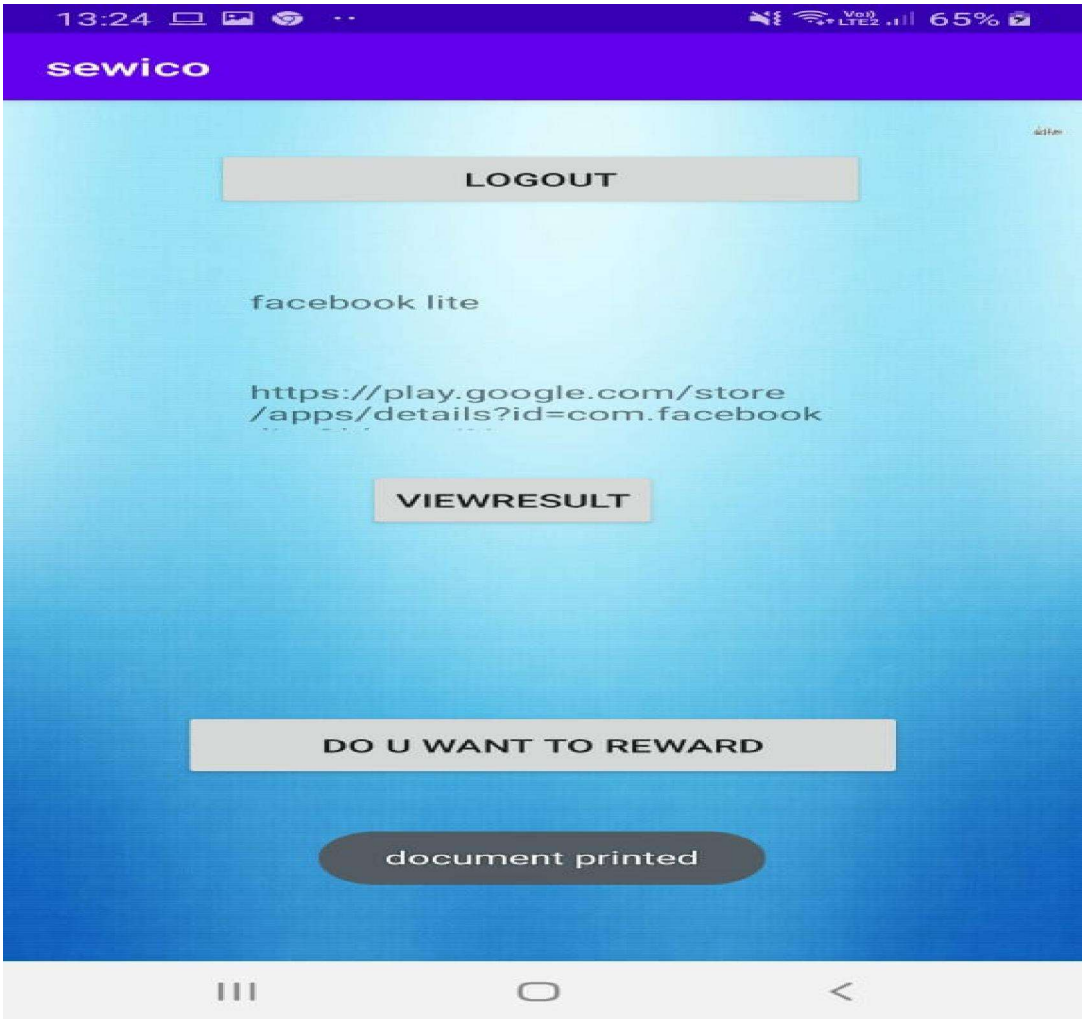


Fig 6.4 APP recommendation and link to download app

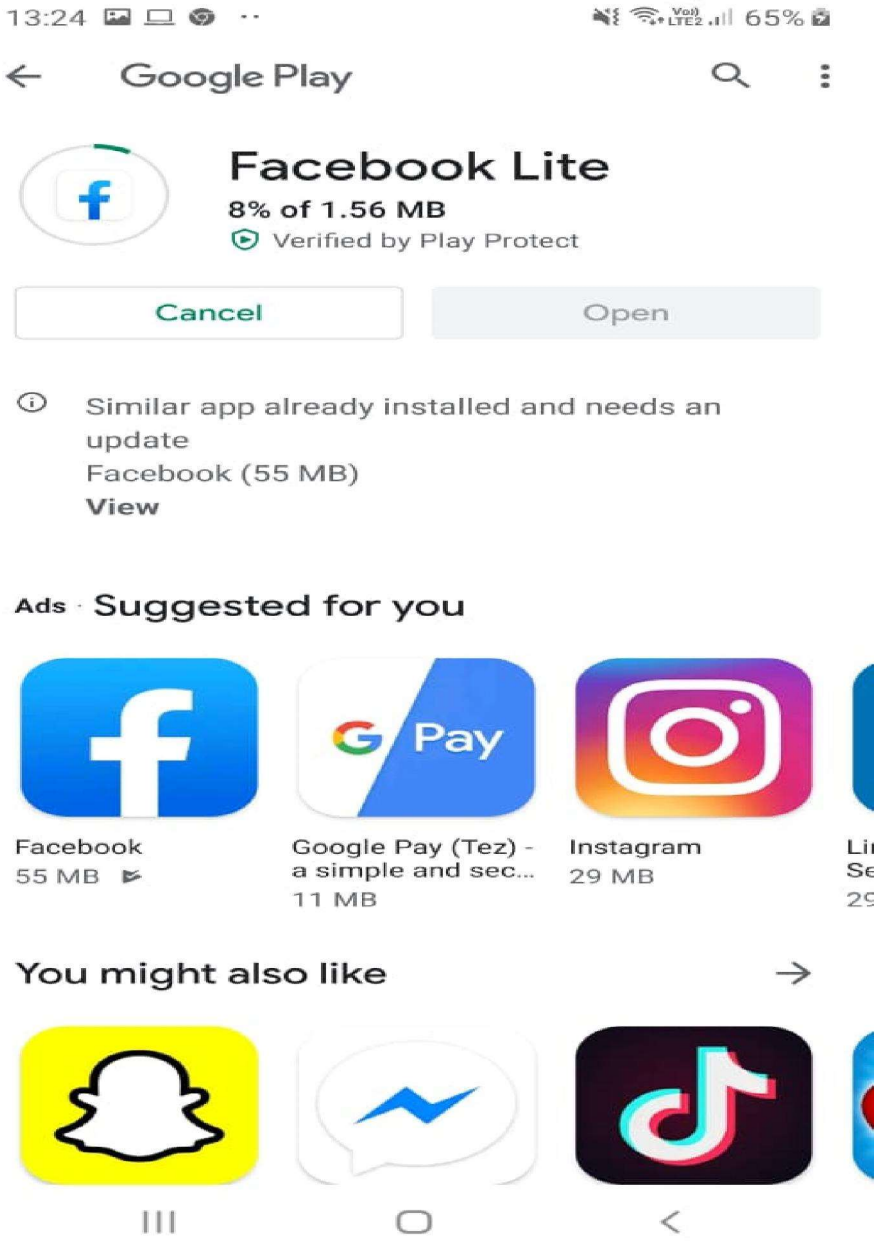


Fig 6.5: App install

CHAPTER 7
CONCLUSIONS AND FUTURE ENHANCEMENTS

The app is developed to serve the user requirements, suppose when we are suppose to buy a product we look for the best one, the one which meets the requirements. But in Play store there are millions of apps , there are many fake apps there are many apps with similar names, similar features, some with slight different features, the naïve user or even any user may not have correct way to install the best app that saves their time and cost.

For this we come up with a solution “sewico”-search the best with conveince that takes the users search key and their requirements and interests and recommends the best application for them to use.

Reinforcement learning is integrated with the recommendation system to produce the best predictions and recommendations to the user, based on the users feedback the app learns by itself and produces the best output the user, the app learns continuously to produce best and accurate output.

FUTURE ENHANCEMENTS:

The app works in android Operating system only to install android applications:

- The app for windows and ios apps can be developed
- A feature to personalize the apps based on the user can be developed
- A feature to download the content from web also can be developed
- A feature to use app in multiple platforms like other devices can be developed

APENDIX

A. SOURCE CODE/ PSUEDO CODE:

1)Splash activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/backgr"
    tools:context=".SplashActivity">

    <Button
        android:id="@+id/btnSlogin"
        android:layout_width="123dp"
        android:layout_height="50dp"
        android:text="Login"
        android:background="@drawable/custom_button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.156"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.93" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="220dp"
        android:layout_height="257dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.407"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />

    <Button
        android:id="@+id/btnSregister"
        android:layout_width="123dp"
        android:layout_height="50dp"
        android:background="@drawable/custom_white"
        android:text="Register"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/btnSlogin"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.93" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

SPLASH ACTIVITY.java

```
package com.example.sewico;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class SplashActivity extends AppCompatActivity {
    private Button Register;
    private Button slogin;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        Register=findViewById(R.id.btnSregister);
        slogin=findViewById(R.id.btnSlogin);
        Register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(SplashActivity.this,RegistrationActivity.class));
            }
        });
        slogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SplashActivity.this,MainActivity.class));
            }
        });
    }
}
```

```
<Button
        android:id="@+id/btnRegister"
        android:layout_width="119dp"
        android:layout_height="50dp"
        android:text="@string/register"
        app:layout_constraintBottom_toBottomOf="parent"
        android:background="@drawable/custom_white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/etvPassword"
        app:layout_constraintVertical_bias="0.181" />

    <TextView
        android:id="@+id/tvUserLogin"
        style="@style/TextAppearance.AppCompat.Large"
        android:layout_width="358dp"
        android:layout_height="35dp"
        android:background="@drawable/custom_button"
        android:text="Already Signedin? Login!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnRegister" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Registration activity.java

```
package com.example.sewico;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class RegistrationActivity extends AppCompatActivity {

    private EditText username;
    private EditText useremail;
    private EditText userpassword;
    private Button register;
    private TextView userlogin;
    private FirebaseAuth firebaseAuth;
    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
       username=findViewById(R.id.etvUsername);
       useremail=findViewById(R.id.etvEmail);
       userpassword=findViewById(R.id.etvPassword);
       register=findViewById(R.id.btnRegister);
       userlogin=findViewById(R.id.tvUserLogin);
        firebaseAuth = FirebaseAuth.getInstance();
        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String user_email=useremail.getText().toString().trim();
                String user_password=userpassword.getText().toString().trim();

firebaseAuth.createUserWithEmailAndPassword(user_email,user_password).addOnCompleteLi
stener(new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if(task.isSuccessful()) {
                            Toast.makeText(RegistrationActivity.this, "Successfully
Registered, Upload complete!", Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(RegistrationActivity.this,
MainActivity.class));
                        }
                        else {
                            Toast.makeText(RegistrationActivity.this, "failed
registration", Toast.LENGTH_SHORT).show();
                        }
                        }
                    });
                }
            });
        userlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(RegistrationActivity.this,MainActivity.class));
            }
        });
        }
    }
```

Login activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/backgr"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btnLogin"
        android:layout_width="101dp"
        android:layout_height="50dp"
        android:text="@string/login"
        android:background="@drawable/custom_button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.458"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.748" />

    <TextView
        android:id="@+id/tvInfo"
        android:layout_width="374dp"
        android:layout_height="46dp"
        android:background="@drawable/custom_white"
        android:text="no_of_incorrect_attempts"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.571"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnLogin"
        app:layout_constraintVertical_bias="0.58000004" />

    <EditText
        android:id="@+id/etP"
        android:layout_width="323dp"
        android:layout_height="57dp"
        android:autofillHints=""
        android:background="@drawable/custom_white"
        android:ems="10"
        android:hint="@string/_123456"
        android:inputType="textPersonName"
        android:text="*********"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.63" />
```

```
        Info.setText("No of attempts remaining: 5");
        firebaseAuth=FirebaseAuth.getInstance();
        progressDialog = new ProgressDialog(this);
        FirebaseUser user=firebaseAuth.getCurrentUser();
        if(user!=null)
        {
            finish();
            startActivity(new Intent(MainActivity.this,AppSearchActivity.class));
        }
        Login.setOnClickListener(new View.OnClickListener() {
                                     @Override
                                     public void onClick(View view) {
                                         validate(Name.getText().toString(),
Password.getText().toString());
                                     }
                                 });




    }

    @SuppressLint("SetTextI18n")
    private void validate(String userName, String userPassword){
        progressDialog.setMessage("You can subscribe to my channel until you are
verified!");
        progressDialog.show();
firebaseAuth.signInWithEmailAndPassword(userName,userPassword).addOnCompleteListener(
new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()) {
            progressDialog.dismiss();
            Toast.makeText(MainActivity.this,"Login
sucess",Toast.LENGTH_SHORT).show();
            checkEmailVerification();
            startActivity(new Intent(MainActivity.this,AppSearchActivity.class));
        }
        else {
            Toast.makeText(MainActivity.this,"Login
failed",Toast.LENGTH_SHORT).show();
            counter--;
            Info.setText("No of attempts remaining: " + counter);
            progressDialog.dismiss();
            if(counter == 0){
                Login.setEnabled(false);
            }
        }
        }
    });
}
    private void checkEmailVerification(){
        FirebaseUser firebaseUser = firebaseAuth.getInstance().getCurrentUser();
        Boolean emailflag = firebaseUser.isEmailVerified();
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvSearch"
        app:layout_constraintVertical_bias="0.034" />

    <TextView
        android:id="@+id/tv2"
        android:layout_width="679dp"
        android:layout_height="101dp"
        android:text="item1"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv1"
        app:layout_constraintVertical_bias="0.046" />

    <TextView
        android:id="@+id/tv3"
        style="@style/TextAppearance.AppCompat.Display1"
        android:layout_width="676dp"
        android:layout_height="107dp"
        android:text="item1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv2"
        app:layout_constraintVertical_bias="0.037" />

    <TextView
        android:id="@+id/tv4"
        android:layout_width="686dp"
        android:layout_height="97dp"
        android:text="item1"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv3"
        app:layout_constraintVertical_bias="0.084" />

    <TextView
        android:id="@+id/tv5"
        android:layout_width="670dp"
        android:layout_height="106dp"
        android:text="item1"
        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv4"
        app:layout_constraintVertical_bias="0.093" />
```

```
<TextView
        android:id="@+id/tvInfoc"
        style="@style/TextAppearance.AppCompat.Display1"
        android:layout_width="465dp"
        android:layout_height="39dp"
        android:text="click on anyone to continue"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.405"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv5" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Appsearch.java

```
package com.example.sewico;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Notification;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.QuerySnapshot;
import com.google.firebase.firestore.SetOptions;
import com.google.firestore.v1.WriteResult;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class AppSearchActivity extends AppCompatActivity {
EditText searchkey;
ImageView search;
TextView t1;
TextView t2;
```

```
TextView t3;
TextView t4;
TextView t5;
TextView info;
FirebaseFirestore db = FirebaseFirestore.getInstance();
private DocumentReference noteRef=db.document("suggestion/cJ");
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_app_search);
        searchkey=findViewById(R.id.etvSearch);
        search=findViewById(R.id.btnSearch);
        t1=findViewById(R.id.tv1);
        t2=findViewById(R.id.tv2);
        t3=findViewById(R.id.tv3);
        t4=findViewById(R.id.tv4);
        t5=findViewById(R.id.tv5);
        info=findViewById(R.id.tvInfoc);

        search.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // Create a new user with a first and last name
                Map<String, Object> user = new HashMap<>();
                String searchn = searchkey.getText().toString();
                user.put("x", searchn);
                db.collection(java.lang.String.valueOf("result")).document("BJ")
                        .set(user, SetOptions.merge());




            }
        });
        info.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                noteRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        if(documentSnapshot.exists())
                        {
                            String s1=documentSnapshot.getString("0");
                            String s2=documentSnapshot.getString("1");
                            String s3=documentSnapshot.getString("2");
                            String s4=documentSnapshot.getString("3");
                            String s5=documentSnapshot.getString("4");
                            t1.setText(s1);
                            t2.setText(s2);
                            t3.setText(s3);
                            t4.setText(s4);
                            t5.setText(s5);
                            Toast.makeText(AppSearchActivity.this,"document
printed",Toast.LENGTH_SHORT).show();
                        }
                        else
```

```
TextView t3;
TextView t4;
TextView t5;
TextView info;
FirebaseFirestore db = FirebaseFirestore.getInstance();
private DocumentReference noteRef=db.document("suggestion/cJ");
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_app_search);
        searchkey=findViewById(R.id.etvSearch);
        search=findViewById(R.id.btnSearch);
        t1=findViewById(R.id.tv1);
        t2=findViewById(R.id.tv2);
        t3=findViewById(R.id.tv3);
        t4=findViewById(R.id.tv4);
        t5=findViewById(R.id.tv5);
        info=findViewById(R.id.tvInfoc);

        search.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // Create a new user with a first and last name
                Map<String, Object> user = new HashMap<>();
                String searchn = searchkey.getText().toString();
                user.put("x", searchn);
                db.collection(java.lang.String.valueOf("result")).document("BJ")
                        .set(user, SetOptions.merge());




            }
        });
        info.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                noteRef.get().addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        if(documentSnapshot.exists())
                        {
                            String s1=documentSnapshot.getString("0");
                            String s2=documentSnapshot.getString("1");
                            String s3=documentSnapshot.getString("2");
                            String s4=documentSnapshot.getString("3");
                            String s5=documentSnapshot.getString("4");
                            t1.setText(s1);
                            t2.setText(s2);
                            t3.setText(s3);
                            t4.setText(s4);
                            t5.setText(s5);
                            Toast.makeText(AppSearchActivity.this,"document
printed",Toast.LENGTH_SHORT).show();
                        }
                        else
```

```
        });
        t5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Map<String, Object> index = new HashMap<>();
                index.put("index", 4);
                db.collection(java.lang.String.valueOf("index")).document("kJ")
                        .set(index, SetOptions.merge());
                startActivity(new
Intent(AppSearchActivity.this,ThirdActivity.class));
            }
        });


            }

            }
```

Userrequirement.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/backgr"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvHead"
        style="@style/TextAppearance.AppCompat.Display1"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:text="Enter requirements"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.036" />

    <EditText
        android:id="@+id/etvSize"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Size can be allocated?(In MB)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvHead"
        app:layout_constraintVertical_bias="0.046" />

    <EditText
        android:id="@+id/etvAge"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="What is your Age?"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvSize"
        app:layout_constraintVertical_bias="0.059" />

    <EditText
        android:id="@+id/etvFree"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Do u want free app(yes/no)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvAge"
        app:layout_constraintVertical_bias="0.054" />

    <EditText
        android:id="@+id/etvIap"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Spend money on app purchase(yes/no)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvFree"
        app:layout_constraintVertical_bias="0.051" />

    <EditText
        android:id="@+id/etvPrice"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:text="How much money to buy app!?(..$)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.552"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvIap"
        app:layout_constraintVertical_bias="0.077" />

    <EditText
        android:id="@+id/etvIapp"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="how much money to buy inapp purchase?(.$)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.568"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvPrice"
        app:layout_constraintVertical_bias="0.058" />

    <EditText
        android:id="@+id/etvAV"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="required android version?"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.585"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvIapp"
        app:layout_constraintVertical_bias="0.088" />

    <EditText
        android:id="@+id/etvrate"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Rating of app!??(0.0-4.0)"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.62"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvAV"
        app:layout_constraintVertical_bias="0.135" />

    <EditText
        android:id="@+id/etvPurpose"
        style="@style/TextAppearance.AppCompat.Small"
        android:layout_width="339dp"
        android:layout_height="42dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="What is your purpose?"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.579"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvrate"
        app:layout_constraintVertical_bias="0.209" />

    <ImageView
        android:id="@+id/btnURSearch"
        android:layout_width="111dp"
        android:layout_height="35dp"
        android:layout_marginTop="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvPurpose"
        app:srcCompat="@drawable/searchbutton" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Userrquirement.java

```
package com.example.sewico;


import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;


import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.SetOptions;

import java.util.HashMap;
import java.util.Map;


public class ThirdActivity extends AppCompatActivity {

EditText appsize;
EditText age;
EditText appfree;
EditText appiap;
EditText appprice;
EditText appiapp;
EditText aversion;
EditText apprate;
EditText purpose;
ImageView URsearch;
FirebaseFirestore db = FirebaseFirestore.getInstance();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_third);
        appsize=findViewById(R.id.etvSize);
        age=findViewById(R.id.etvAge);
        appfree=findViewById(R.id.etvFree);
        appiap=findViewById(R.id.etvIap);
        appprice=findViewById(R.id.etvPrice);
        appiapp=findViewById(R.id.etvIapp);
        aversion=findViewById(R.id.etvAV);
        apprate=findViewById(R.id.etvrate);
        purpose=findViewById(R.id.etvPurpose);
        URsearch=findViewById(R.id.btnURSearch);

        URsearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Map<String, Object> userr = new HashMap<>();
                String appsizen = appsize.getText().toString();
                userr.put("size", appsizen);
                String agen = age.getText().toString();
                userr.put("age", agen);
                String appfreen = appfree.getText().toString();
                userr.put("free", appfreen);
                String appiapn = appiap.getText().toString();
                userr.put("iap", appiapn);
                String appprican = appprice.getText().toString();
                userr.put("price", appprican);
                String appiappn = appiapp.getText().toString();
                userr.put("iapp", appiappn);
                String aversionn = aversion.getText().toString();
                userr.put("aversion", aversionn);
                String appraten = apprate.getText().toString();
                userr.put("rating", appraten);
                String purposen = purpose.getText().toString();
                userr.put("purpose", purposen);
                db.collection("userrequirements").document("cJ")
                        .set(userr, SetOptions.merge());
                startActivity(new Intent(ThirdActivity.this,SearchActivity.class));
            }
        });

    }
}
```

Reward.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/backgr"
    tools:context=".RewardActivity">

    <EditText
        android:id="@+id/etReward"
        android:layout_width="361dp"
        android:layout_height="44dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="reward(Like app-(0-5)/dislike app-(-5 to -1))"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnLearn"
        android:layout_width="130dp"
        android:layout_height="48dp"
        android:text="View App"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etReward"
        app:layout_constraintVertical_bias="0.256" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Reward.java

```
import java.util.HashMap;
import java.util.Map;

public class RewardActivity extends AppCompatActivity {
    EditText reward;
    Button appresultlearn;
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reward);
        reward=findViewById(R.id.etReward);
        appresultlearn=findViewById(R.id.btnLearn);
        appresultlearn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Map<String, Object> userreward = new HashMap<>();
                String rewardn = reward.getText().toString();
                userreward.put("x", rewardn);
```

```
                db.collection(java.lang.String.valueOf("reward")).document("BJ")
                        .set(userreward, SetOptions.merge());
                startActivity(new Intent(RewardActivity.this,SearchActivity.class));


            }
        });

    }
}
```

custom button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
    <solid android:color="#03DAC0"></solid>
    <corners android:radius="10dp"></corners>
</shape>
```

Android manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.sewico">

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="sewico"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="AllowBackup">
        <activity android:name=".RewardActivity"></activity>
        <activity android:name=".SearchActivity" />
        <activity android:name=".AppSearchActivity" />
        <activity android:name=".SplashActivity" />
        <activity android:name=".MainActivity" />
        <activity android:name=".ThirdActivity" />
        <activity android:name=".RegistrationActivity" />
        <activity android:name=".SplashActivity1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Buildgradle library

```
buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.2'
        classpath 'com.google.gms:google-services:4.3.3'




        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()

    }
}
```

build gradle app

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.example.sewico"
        minSdkVersion 16
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        multiDexEnabled true
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"


    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }

}

dependencies {
```

```
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.firebase:firebase-auth:19.3.0'
    implementation 'com.google.firebase:firebase-database:16.0.4'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:appcompat-v7:28.0.0'
    testImplementation 'junit:junit:4.12'
    implementation 'com.android.support:multidex:1.0.3'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.google.firebase:firebase-firestore:21.4.2'
}
apply plugin: 'com.google.gms.google-services'
```

WEBSCRAPPING.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 14 16:20:54
 2019

@author: sairam pasupuleti
"""

from requests import get
import xlwt
import play_scraper
from xlwt import Workbook
x=['ART_AND_DESIGN']
for psr in x:
    wb = Workbook()
    url='https://play.google.com/store/apps/'+'category/'+psr
    response = get(url)
    from bs4 import BeautifulSoup
    html_soup = BeautifulSoup(response.text, 'lxml')
    type(html_soup)
    linkss= html_soup.find_all('a', class_ = 'JC71ub')
    linkxx=[]
    for container in linkss:
    # If the movie has Metascore, then extract:
        if container.find_all('a', class_ = 'JC71ub') is not None:
    # The name
         linka = container['href']
         linkxx.append(linka)
    links= html_soup.find_all('a', class_ = 'LkLjZd ScJHi U8Ww7d xjAeve nMZKrb id-track-
click')
    linkx=[]
    for container in links:
```

WEBSCRAPPING.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov 14 16:20:54
 2019

@author: sairam pasupuleti
"""

from requests import get
import xlwt
import play_scraper
from xlwt import Workbook
x=['ART_AND_DESIGN']
for psr in x:
    wb = Workbook()
    url='https://play.google.com/store/apps/'+'category/'+psr
    response = get(url)
    from bs4 import BeautifulSoup
    html_soup = BeautifulSoup(response.text, 'lxml')
    type(html_soup)
    linkss= html_soup.find_all('a', class_ = 'JC71ub')
    linkxx=[]
    for container in linkss:
    # If the movie has Metascore, then extract:
        if container.find_all('a', class_ = 'JC71ub') is not None:
    # The name
         linka = container['href']
         linkxx.append(linka)
    links= html_soup.find_all('a', class_ = 'LkLjZd ScJHi U8Ww7d xjAeve nMZKrb id-track-
click')
    linkx=[]
    for container in links:
```

```
count=1
for i in linkxx:
  ks=i
  lx=ks[23:len(ks)]
  sd=play_scraper.similar(lx,0.1)
  for h in range(0,len(sd)):
     count1=0
     for g in range(0,len(nameinexcel)):
        psr.write(count,count1,sd[h][nameinexcel[g]])
        count1=count1+1
     count=count+1
     res=res+1
     print(res)



     wb.save(save+'1.xls')
```

```
from firebase_admin import credentials


cred = credentials.Certificate("/Users/sairam pasupuleti/Desktop/x/sewico-864c5-firebase-adminsdk-rek9r-1e0338962b.json")
#firebase_admin.initialize_app(cred)
# Use the application default credentials


#default_app=firebase_admin.initialize_app(cred)
db=firestore.client()

doc_ref = db.collection('suggestion').document('cJ')
users_ref = db.collection("userrequirements")
docs = users_ref.stream()
for doc in docs:
    xfire2=doc.to_dict()

user_ref=db.collection('result')
docs=user_ref.stream()
for doc in docs:
 xfire=doc.to_dict()
y={}
if xfire['x']!='loading':
    x=play_scraper.suggestions(xfire['x'])
    doc_ref.set({
    '0':x[0],
    '1':x[1],
    '2':x[2],
    '3':x[3],
    '4':x[4]})
    user_ref1=db.collection('index')
    docs1=user_ref1.stream()
```

```
        for doc1 in docs1:
         xfire1=doc1.to_dict()
        appsuggest=x[xfire1['index']]
        count=0
        if( (xfire2['purpose']!='loading') and (xfire2['age']!='loading' )and
           (xfire2['free']!='loading') and
           (xfire2['iap']!='loading') and
           (xfire2['price']!='loading') and
           (xfire2['rating']!='loading') and
           (xfire2['size']!='loading')):
           file ='/Users/sairam pasupuleti/Desktop/x/'+xfire2['purpose']+'1.xls'
           df = pd.read_excel(file)
           #print(df)
           app_id=df['app_id']
           #print(app_id)
           #print("content rating before cleaning")
           content_rating=df['content_rating']
           #print(content_rating)
           # Label encoding
           lb_make = LabelEncoder()
           # Create column for "numeric" Content Rating
           df["Content Rating NUM"] = lb_make.fit_transform(df["content_rating"])
           # Form dicitonary for Content Rating and numeric values
           dict_content_rating = {"Adults only 18+": 0, "Everyone": 1, "Everyone 10+": 2, "Mature
17+": 3, "Teen": 4}
           # Numeric value for Content Rating
           '''
           Adults only 18+ = 0
           Everyone = 1
           Everyone 10+ = 2
           Mature 17+ = 3
           Teen = 4
           '''
```

```
content_rating_num=df['Content Rating NUM']
print(content_rating_num)
#print("rating before")
current_version=df['current_version']
#print("ratings after")
# Replace "NaN" with mean
df['current_version'] =df['current_version'].map(lambda x: 0 if x.startswith('Varies') else x)
print(df['current_version'])
editors_choice=df['editors_choice']
df["EDITORS CHOICE NUM"] = lb_make.fit_transform(df["editors_choice"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["EDITORS CHOICE NUM"])
editors_choice_num=df["EDITORS CHOICE NUM"]
df["FREE NUM"] = lb_make.fit_transform(df["free"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
free_num=df["FREE NUM"]
print(df["FREE NUM"])
df["IAP NUM"] = lb_make.fit_transform(df["iap"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["IAP NUM"])
iap_num=df["IAP NUM"]
df['installs'] = df['installs'].map(lambda x: x.rstrip('+'))
df['installs'] = df['installs'].map(lambda x: ''.join(x.split(',')))
print(df['installs'])
installs=df['installs']
df['price'] = df['price'].map(lambda x: str(x).lstrip('$').rstrip())
price=df['price']
df['required_android_version']  =df['required_android_version'].map(lambda  x:  0  if
x.startswith('Varies') else x)
android_version=df['required_android_version']
```

```
df['reviews'] = pd.to_numeric(df['reviews'])
reviews=df['reviews']
imputer = SimpleImputer()
df['score'] = imputer.fit_transform(df[['score']])
# Rounding the mean value to 1 decimal place
df['score'].round(1)
df.dropna(axis=0, inplace=False)
print(df['score'])
# Change datatype
score=df['score']
df['size'] = df['size'].map(lambda x: x.rstrip('M'))
df['size'] = df['size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k'
else x)
df['size'] = df['size'].map(lambda x: 0 if x.startswith('Varies') else x)
print(df['size'])
size=df['size']
user_content_ip=int(xfire2['age'])
cr=content_rating_num
contdis=[]
for i in range(0,len(cr)):
    if user_content_ip>cr[i]:
        contdis.append(user_content_ip-cr[i])
    else:
        contdis.append(cr[i]-user_content_ip)
print(contdis)
print(df)
editors_choice_ip=int(xfire2['editor'])
er=editors_choice_num
editdis=[]
for i in range(0,len(er)):
    if editors_choice_ip>er[i]:
        editdis.append(editors_choice_ip-er[i])
    else:
```

```
            prdis.append(price_ip-float(pr[i]))
        print(prdis)
        reviews_ip=int(xfire2['review'])
        rrr=reviews
        rrrdis=[]
        for i in range(0,len(rrr)):
            rrrdis.append(int(rrr[i])-reviews_ip)
        print(rrrdis)
        score_ip=int(xfire2['rating'])
        srr=score
        srrdis=[]
        for i in range(0,len(srr)):
          if score_ip>int(srr[i]):
            srrdis.append(int(srr[i])-score_ip)
          else:
            srrdis.append(score_ip-int(srr[i]))
        print(srrdis)
        size_ip=int(xfire2['size'])
        sir=size
        sirdis=[]
        for i in range(0,len(sir)):
          if size_ip>float(sir[i]):
            sirdis.append(float(sir[i])-size_ip)
          else:
            sirdis.append(size_ip-float(sir[i]))
        print(sirdis)
        result=[]
        for i in range(0,len(editdis)):

result.append(contdis[i]+editdis[i]+freedis[i]+iapdis[i]+istdis[i]+prdis[i]+rrrdis[i]+srrdis[i]+sirdi
s[i])
        print(result)
        tmp_sorted = sorted(result, reverse=False)  # kudos to @Wondercricket
```

```
res = [tmp_sorted.index(x) + 1 for x in result]
print(res)
ss=df
for i in range(len(res),len(df)):
   res.append(randint(len(res),len(df)))
print(res)
features = ['reviews', 'size', 'installs', 'price', 'Content Rating NUM','IAP NUM','FREE NUM','score']
X = df[features]
# Label selection
y = res
train_X, test_X, train_y, test_y = train_test_split(X, y)
total_sum = []
for i in range(10):
 # Hypertuning of parameters for better prediction
 forest_model = RandomForestRegressor(n_estimators=100, max_features=3, min_samples_leaf=10)
 forest_model.fit(X, y)
 # For testing purpose
forest_model.fit(train_X, train_y)
from requests import get
import xlwt
from xlwt import Workbook
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
wb = Workbook()
psr= wb.add_sheet('x')
p=play_scraper.search(appsuggest,page=2)
#x=input("enter to suggest")
#y=play_scraper.suggestions(x)
```

```
#print(y)
#z=int(input("select one"))
#p=play_scraper.search(y[z], page=2)
print(p)
q=[]
w=[]
for i in range(0,len(p)):
 q.append(p[i]['app_id'])
for j in range(0,len(q)):
  w.append(play_scraper.details(q[j]))
ind=['title', 'icon', 'screenshots', 'video', 'category', 'score', 'reviews', 'description', 'editors_choice', 'price', 'free', 'iap', 'developer_id', 'updated', 'size', 'installs', 'current_version', 'required_android_version', 'content_rating', 'iap_range', 'interactive_elements', 'developer', 'developer_email', 'developer_url', 'developer_address', 'app_id', 'url']
for i in range(0,len(ind)):
        psr.write(0,i,ind[i])
for i in range(1,len(w)):
    for j in range(0,len(ind)):
      psr.write(i,j,w[i][ind[j]])
      print(i)
      wb.save('xop'+'1.xls')
fil ='xop1.xls'
df = pd.read_excel(fil)
print("before cleaning data")
print(df['size'])
print("after cleaning data")
df['size'] = df['size'].map(lambda x: x.rstrip('M'))
df['size'] = df['size'].map(lambda x: str(round((float(x.rstrip('k'))/1024), 1)) if x[-1]=='k' else x)
df['size'] = df['size'].map(lambda x: 0 if x.startswith('Varies') else x)
print(df['size'])
print("installs befor cleaning")
print(df['installs'])
```

```
        # Data cleaning for "Installs" column
        df['installs'] = df['installs'].map(lambda x: x.rstrip('+'))
        df['installs'] = df['installs'].map(lambda x: ''.join(x.split(',')))
        print(df['installs'])
        print("price before cleaning")
        print(df['price'])
        df['price'] = df['price'].map(lambda x: str(x).lstrip('$').rstrip())
        print(df['price'])
        print("content rating before cleaning")
        print(df['content_rating'])
        # Label encoding
        lb_make = LabelEncoder()
        # Create column for "numeric" Content Rating
        df["Content Rating NUM"] = lb_make.fit_transform(df["content_rating"])
        # Form dicitonary for Content Rating and numeric values
        dict_content_rating = {"Adults only 18+": 0, "Everyone": 1, "Everyone 10+": 2, "Mature
17+": 3, "Teen": 4}
        # Numeric value for Content Rating
        '''
        Adults only 18+ = 0
        Everyone = 1
        Everyone 10+ = 2
        Mature 17+ = 3
        Teen = 4
        '''
        print(df['Content Rating NUM'])
        print("rating before")
        print(df['score'])
        print("ratings after")
        # Replace "NaN" with mean
        imputer = SimpleImputer()
        df['score'] = imputer.fit_transform(df[['score']])
        # Rounding the mean value to 1 decimal place
```

```
df['score'].round(1)
df.dropna(axis=0, inplace=False)
print(df['score'])
# Change datatype
df['reviews'] = pd.to_numeric(df['reviews'])
df['installs'] = pd.to_numeric(df['installs'])
df['price'] = pd.to_numeric(df['price'])
df['current_version'] = df['current_version'].map(lambda x: 0 if x.startswith('Varies') else
x)
df["EDITORS CHOICE NUM"] = lb_make.fit_transform(df["editors_choice"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["EDITORS CHOICE NUM"])
df["FREE NUM"] = lb_make.fit_transform(df["free"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["FREE NUM"])
df["IAP NUM"] = lb_make.fit_transform(df["iap"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["IAP NUM"])
a=df[ 'reviews']
b=df['size']
c=df['installs']
d=df[ 'price']
e=df['Content Rating NUM']
f=df['IAP NUM']
g=df['FREE NUM']
h=df['score']
sss=df['title']
ld=df['url']
p=a
q=b
```

```
df['score'].round(1)
df.dropna(axis=0, inplace=False)
print(df['score'])
# Change datatype
df['reviews'] = pd.to_numeric(df['reviews'])
df['installs'] = pd.to_numeric(df['installs'])
df['price'] = pd.to_numeric(df['price'])
df['current_version'] = df['current_version'].map(lambda x: 0 if x.startswith('Varies') else
x)
df["EDITORS CHOICE NUM"] = lb_make.fit_transform(df["editors_choice"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["EDITORS CHOICE NUM"])
df["FREE NUM"] = lb_make.fit_transform(df["free"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["FREE NUM"])
df["IAP NUM"] = lb_make.fit_transform(df["iap"])
# Form dicitonary for Content Rating and numeric values
dict_editors_choice = {"TRUE":1,"FALSE":0}
print(df["IAP NUM"])
a=df[ 'reviews']
b=df['size']
c=df['installs']
d=df[ 'price']
e=df['Content Rating NUM']
f=df['IAP NUM']
g=df['FREE NUM']
h=df['score']
sss=df['title']
ld=df['url']
p=a
q=b
```

```
r=c
s=d
t=e
u=f
v=g
w=h
predres=[]
for i in range(0,len(p)):
 forest_pred = forest_model.predict([[p[i],q[i],r[i],s[i],t[i],u[i],v[i],w[i]]])
 predres.append(forest_pred)
print(predres)
xres=predres
for i in range(0,len(xres)):
  xres[i]=int(xres[i])
tmp_sortedd = sorted(xres, reverse=False)  # kudos to @Wondercricket
dres = [tmp_sortedd.index(x) + 1 for x in xres]
print(min(xres))
ddd=xres.index(min(xres))
users_refrew = db.collection('reward')
docsrew = users_refrew.stream()
for docrew in docsrew:
     xfire2rew=docrew.to_dict()
if xfire2rew['x']!='loading':
 xres[ddd]=xres[ddd]+(((randint(0,9))*(int(xfire2rew['x']))+((0.7*max(xres)))))
 for i in range(0,len(editdis)):

result[i]=xres[ddd]+result[i]+(((randint(0,i))*(int(xfire2rew['x']))+(((i/10)*max(xres)))))
else:
   xres[ddd]=xres[ddd]
count=count+1
print(min(xres))
dd1=xres.index(min(xres))
dd2=sss[dd1]
```

```
        ld2=ld[dd1]
        print(dd2)
        doc_ref1 = db.collection('resultapp').document('userapp')
        doc_ref1.set({
          'result':dd2,
          'link':ld2
                  })
    app.debug = True
    return app



if __name__ == '__main__':
   app.run()
```

App link to download:
https://drive.google.com/drive/u/0/my-drive

to create sever link
https://drive.google.com/drive/u/0/my-drive

THE END