.

i
i

CHAPTER 1

INTRODUCTION

1.1 Introduction to Mobile Application Development

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and IOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Mobile app development is rapidly growing. From retail, telecommunications and e-commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transactions and access information.

Today, mobile devices—and the mobile applications that unlock their value—are the most popular way for people and businesses to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employee's demand.

1.1.1 What is Mobile App?

A mobile application, also referred to as a mobile app or simply an app, is a computer program or software application designed to run on a mobile device such as a phone, tablet, or watch. Apps were originally intended for productivity assistance such as email, calendar, and contact databases, but the public demand for apps caused rapid expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchases, so that there are now millions of apps available.

Apps are generally downloaded from application distribution platforms which are operated by the owner of the mobile operating system, such as the App Store (iOS) or Google Play Store. Some apps are free, and others have a price, with the profit being split between the application's creator and the distribution platform. Mobile applications often stand in contrast to desktop applications which are designed to run on desktop computers, and web applications which run in mobile web browsers rather than directly on the mobile device.

1.1.2 What is Mobile OS?

A mobile operating system (OS) is software that allows smartphones, tablet PCs (personal computers) and other devices to run applications and programs. A mobile OS typically starts up when a device powers on, presenting a screen with icons or tiles that present information and provide application access. Mobile operating systems also manage cellular and wireless network connectivity, as well as phone access.

1.1.3 Types of Mobile Apps

Native Apps: Such apps are developed for a single mobile operating system exclusively, therefore they are "native" for a particular platform or device. App built for systems like iOS, Android, Windows phone, Symbian, Blackberry can't be used on a platform other than their own.

Hybrid Apps: They are built using multi-platform web technologies (for example HTML5, CSS and JavaScript). So-called hybrid apps are mainly website applications disguised in a native wrapper. Apps possess usual pros and cons of both native and web mobile applications.

Web Apps: These are software applications that behave in a fashion similar to native applications. Web apps use a browser to run and are usually written in HTML5, JavaScript or CSS. These apps redirect a user to URL and offer "install" option by simply creating a bookmark to their page.

1.1.4 Different categories of Mobile Apps

Educational Apps: Educational and informative apps do just that–educate and inform. While the purpose of this type of app is fairly straightforward, there is a lot of diversity when it comes to educational apps, like news and language apps. If you're looking to break into this crowded space, you'll need to serve up news or other information in a fun and unique format for learners of all ages, interests, and levels.

Lifestyle Apps: This app category covers a lot of ground, literally. Where you're going, how you're getting there, what you're going to order off the menu–it all falls under lifestyle apps. Think of apps you use for convenience, like fitness, dating, food, and travel. Lifestyle and leisure apps are increasingly popular, especially for tasks that require an extra step aside from the search itself (e.g., the scary action of actually picking up the phone to make a call).

Social Media Apps: Social media apps give users the opportunity to connect with people inside or outside their social circles. For the most part, social media apps are universal and havea very diverse user base. These apps are used to share live video, post images, facilitate conversations, and more. Social media apps have quickly become part of our everyday lives.

Productivity Apps: Also known as business apps, productivity apps typically organize and complete complex tasks for you, anything from sending an email to figuring out the tip on your dinner bill. Most productivity apps serve a single purpose and are built with a very intuitive interface and design to increase efficiency and improve user experience.

Entertainment Apps: This category of apps has one sole focus–keeping you busy. Entertainment apps are often used to fill your time, whether you are jet-setting across the country, lounging at home, or really anywhere in-between. Along with their websites, a lot of popular streaming services have mobile applications so users can access their library wherever they are. Entertainment apps can include video, text, or audio content.

Game Apps: This app category is pretty self-explanatory and represents the biggest portion of app downloads by far. With such a crowded category, it makes sense that there are so many types of game apps for different target audiences, such as arcade games, brain training puzzles, or just plain silly games, like launching tiny birds at pigs. Some mobile app games are played solo, others online, and occasionally in-person with a group of friends.

1.1.5 What is Mobile Application Development?

Mobile app development is the act or process by which a mobile app is developed for mobile devices, such as personal digital assistants, enterprise digital assistants or mobile phones. These applications can be pre-installed on phones during manufacturing platforms, or delivered as web applications using server-side or client-side processing (e.g., JavaScript) to provide an "application-like" experience within a Web browser.

To develop apps using the SDK, use the Java programming language for developing the app and Extensible Markup Language (XML) files for describing data resources. By writing the code in Java and creating a single app binary, you will have an app that can run on both phone and tablet. To help you develop your apps efficiently, Google offers a full Java Integrated Development Environment (IDE) called Android Studio, with advanced features for developing, debugging, and packaging Android apps. Using Android Studio, you can develop apps on any available Android device, or create virtual devices that emulate any hardware configuration.

1.1.6 Mobile Application Development Challenges

While the Android platform provide rich functionality for app development, there are still a number of challenges you need to address, such as:

- Building for a multi-screen world
- Getting performance right
- Keeping your code and your users secure
- Remaining compatible with older platform versions
- Understanding the market and the user.

1.1.7 How to develop Mobile Apps

Step 1: Set a Goal

Step away from any form of technology and get out a pen and paper and define what it is you want to accomplish. The starting line in the app development word is a pen and paper, not complex coding and designing.

Step 2: Sketch your ideas

You need to use the pen and paper that has the answers to the questions about your apps purpose to develop a sketch of what it will look like. Here you move your clearly worded ideas into visual representations of your thoughts. Decide if you are going to give your app away and offer ads to generate money, or are you going to offer it as a paid download. You can also choose the option to offer in app purchases.

Step 3: Research

You have to dig deep and research the competition of your app idea. I know you think you have one of a kind idea, but the numbers are not in your favor—odds are someone has alreadytried it. You can look at this in two different ways. One you can become deflated and give up,or two, you can examine the competition and make your app better.

Step 4: Wireframe

In the technology world, a wireframe is a glorified story board. Here is where you take your sketch and your design idea, and you give your idea a little more clarity and functionality. This will become the foundation for your app's development, so it really is a crucial step.

Step 5: Start Defining the Back End of Your App

We left off with your wireframe, so at this point in your app development, you have a storyboard of how you want your app to function. Now it's time to use that storyboard to start examine functionality.

Step 6: Check Your Model

Here's where you need to call in the troops. Show your demo to friends, family, and anyone else who is willing to give you constructive criticism. Don't waste your time with people who will tell you, "Wow, that's neat." Seek out those cynics and critics. Brutal honesty is crucial at this phase.

Step 7: Get Building

With the foundation in place, you can start to put the puzzle together to building your app. First, your developer will set up your servers, databases, and APIs.

Step 8: Design the Look

Now it's time to employ the designers to create your UI, user interface. The user interface is a very important part of your app because people are attracted to how things look and how easy they are to navigate.

Step 9: Test Your App, again

A second round of testing is imperative. In this round, you will have both a functioning app as well as a user interface to test. All the screens of your app should properly work at this point, and your app should be visually appealing as well.

Step 10: Modify and Adjust

You've taken your prototype for a spin, and you've learned that there are still a few tweaks you need to make. Now that you've seen your app in it's fully functioning form, you need to call the troops back and ask they to do the same.

Step 11: Beta Testing

You've looked at your app through several different lenses, and you think you've managed to develop a smoothly functioning, aesthetically pleasing, problem solving app. Now, you need to examine how your app is going to function in a live environment.

Step 12: Release Your App

You've made it to the finish line. You've brought your idea to fruition, and the last step is to share it with the world. Hopefully, you've gone on to solve a major problem. If not, with any luck your app has some features that can simplify or bring enjoyment to someone's life. Regardless, you've accomplished something big. Now it's time to distribute it.

1.2 Introduction to Android Studio

Android is one of the most popular mobile device platforms. The Android platform allows developers to write managed code using Java to manage and control the Android device. Android Studio is a popular IDE developed by Google for developing applications that are targeted at the Android platform. Android Studio has replaced Eclipse as the IDE of choice for developing Android applications.

1.2.1 What is Android?

Android is a software package and Linux based operating system for mobile devices such as tablet computers and smartphones. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

1.2.2 Different versions of Android OS

Table 1.1: Different versions of Android OS

Name	Version Numbers	Release Date	API
No official Name	1.0	Sept, 2008	1
No official Name	1.1	Feb, 2009	2
Cupcake	1.5	Apr, 2009	3
Donut	1.6	Sept, 2009	4
Éclair	2.0-2.1	Oct, 2009	5-7
Froyo	2.2-2.2.3	May, 2010	8
Gingerbread	2.3-2.3.7	Dec, 2010	9-10

KitKat	4.4-4.4.4	Oct, 2013	19-20
Lollipop	5.0-5.1.1	Nov, 2014	21-22
Marshmallow	6.0-6.0.1	Oct, 2015	23
Nougat	7.0-7.1.2	Aug, 2016	24-25
Oreo	8.0-8.1	Aug, 2017	26-27
Pie	9	Aug, 2018	28
Android 10	10	Sept, 2019	29
Android 11	11	Sept 2020	30

1.2.3 Android Development Architecture

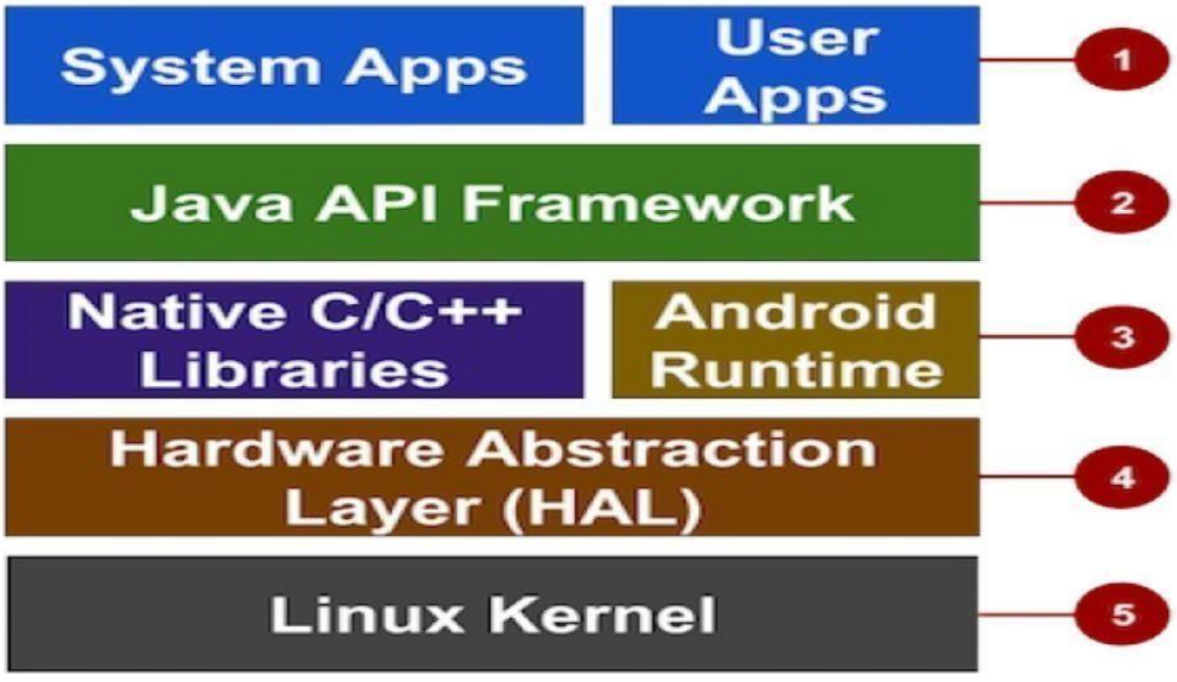


Fig 1.1: Android development architecture.

Apps: Your apps live at this level, along with core system apps for email, SMS messaging, calendars, Internet browsing, or contacts.

Java API Framework: All features of Android are available to developers through application programming interfaces (APIs) written in the Java language. You don't need to know the details of all of the APIs to learn how to develop Android apps, but you can learn more about the following APIs, which are useful for creating apps:

- **View System** used to build an app's UI, including lists, buttons, and menus.
- **Resource Manager** used to access to non-code resources such as localized strings, graphics, and layout files.
- **Notification Manager** used to display custom alerts in the status bar.
- **Activity Manager** that manages the lifecycle of apps.
- **Content Providers** that enable apps to access data from other apps.
- **All framework APIs** that Android system apps use.

Libraries and Android Runtime: Each app runs in its own process and with its own instance of the Android Runtime, which enables multiple virtual machines on low-memory devices. Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features that the Java API framework uses. Many core Android system components and services are built from native code that require native libraries written in C and C++. These native libraries are available to apps through the Java API framework.

Hardware Abstraction Layer (HAL): This layer provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.

Linux Kernel: The foundation of the Android platform is the Linux kernel. The above layers rely on the Linux kernel for underlying functionalities such as threading and low-level memory management.

1.2.4 Installing Android Studio

System Requirements:

- Microsoft® Windows® 7/8/10, Mac or Linux (64-bit).
- 4 GB RAM minimum, 8 GB RAM recommended.
- GB of available disk space minimum, 4 GB Recommended.
- 1280 x 800 minimum screen resolution.

Windows

To install Android Studio on Windows, proceed as follows:

- Download ***android-studio-ide-201.7199119-windows.exe*** file from the https://developer.android.com/studio
- Double-click .exe file to launch it.
- Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Mac

To install Android Studio on your Mac, proceed as follows:

- Launch the Android Studio DMG file.
- Drag and drop Android Studio into the Applications folder, then launch Android Studio.
- Select whether you want to import previous Android Studio settings, then click OK.
- The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Linux

To install Android Studio on Linux, proceed as follows:

- Unpack the .zip file you downloaded to an appropriate location for your applications, such as within /usr/local/ for your user profile, or /opt/ for shared users.
- If you're using a 64-bit version of Linux, make sure you first install the required libraries for 64-bit machines.
- To launch Android Studio, open a terminal, navigate to the android-studio/bin/ directory, and execute studio.sh.
- Select whether you want to import previous Android Studio settings or not, then click OK.
- The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

1.2.5 Android Studio Window Panes

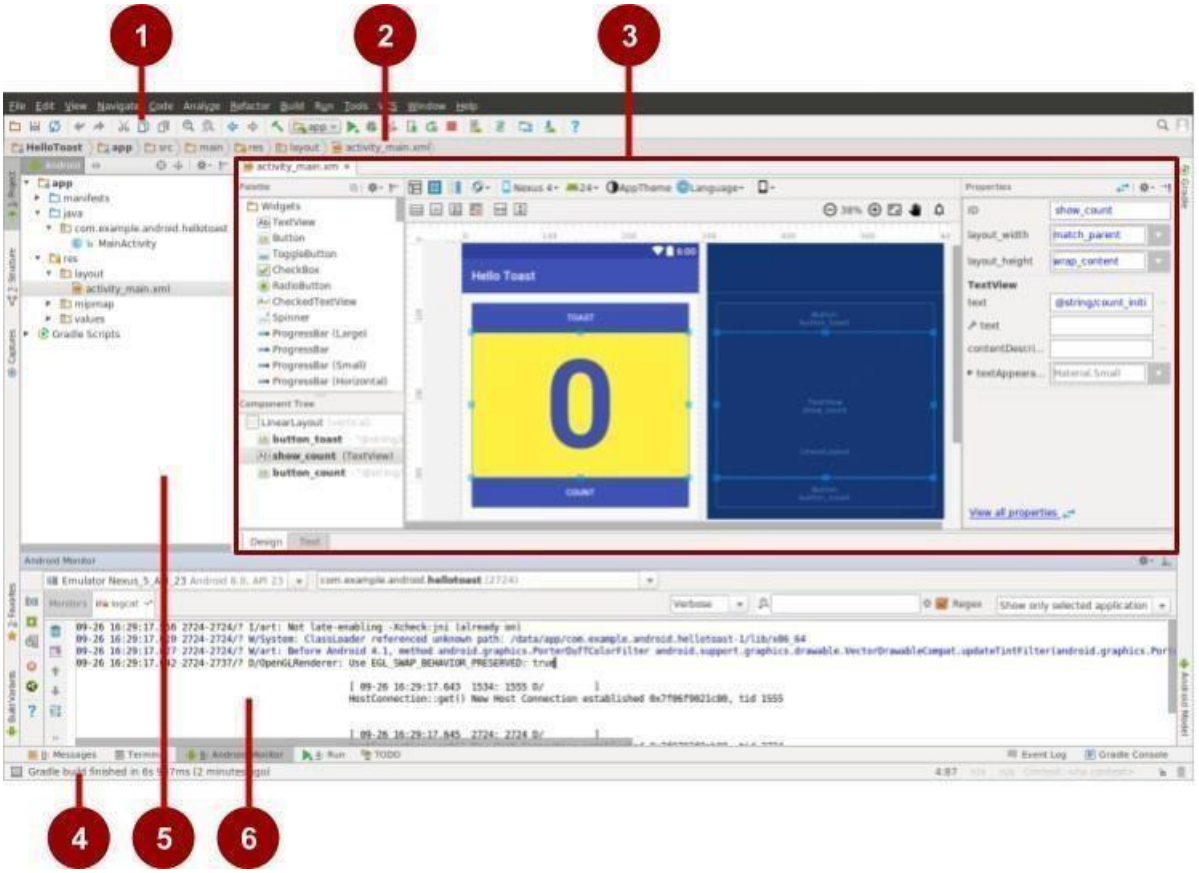


Fig 1.2: Android studio window panes.

- **The Toolbar.** The toolbar carries out a wide range of actions, including running the Android app and launching Android tools.
- **The Navigation Bar**. The navigation bar allows navigation through the project and open files for editing. It provides a more compact view of the project structure.
- **The Editor Pane**. This pane shows the contents of a selected file in the project. For example, after selecting a layout (as shown in the figure), this pane shows the layout editor with tools to edit the layout. After selecting a Java code file, this pane shows the code with tools for editing the code.
- **The Status Bar**. The status bar displays the status of the project and Android Studio itself, as well as any warnings or messages. You can watch the build progress in the status bar.
- **The Project Pane**. The project pane shows the project files and project hierarchy.

- **The Monitor Pane**. The monitor pane offers access to the TODO list for managing tasks, the Android Monitor for monitoring app execution (shown in the figure), the logcat for viewing log messages, and the Terminal application for performing Terminal activities.

1.2.6 Viewing the Android Manifest

Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file. The app must declare all its components in this file, which must be at the root of the app project directory. To view this file, expand the manifests folder in the Project: Android view, and double-click the file (AndroidManifest.xml). Its contents appear in the editing pane as shown in the figure below.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello World"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Fig 1.3: Android manifest file.

1.2.7 Viewing and editing Java code

Components are written in Java and listed within module folders in the java folder in the Project: Android view. Each module name begins with the domain name (such as com.example.android) and includes the app name.

The following example shows an activity component:

- Click the module folder to expand it and show the MainActivity file for the activity written in Java (the MainActivity class).
- Double-click MainActivity to see the source file in the editing pane, as shown in the figure below.

```
package com.example.android.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Fig 1.4: MainActivity.java file.

1.2.8 Viewing and editing layouts

Layout resources are written in XML and listed within the layout folder in the res folder in the Project: Android view. Click res > layout and then double-click activity_main.xml to see the layout file in the editing pane. Android Studio shows the Design view of the layout, as shown in the figure below. This view provides a Palette pane of user interface elements, and a grid showing the screen layout.

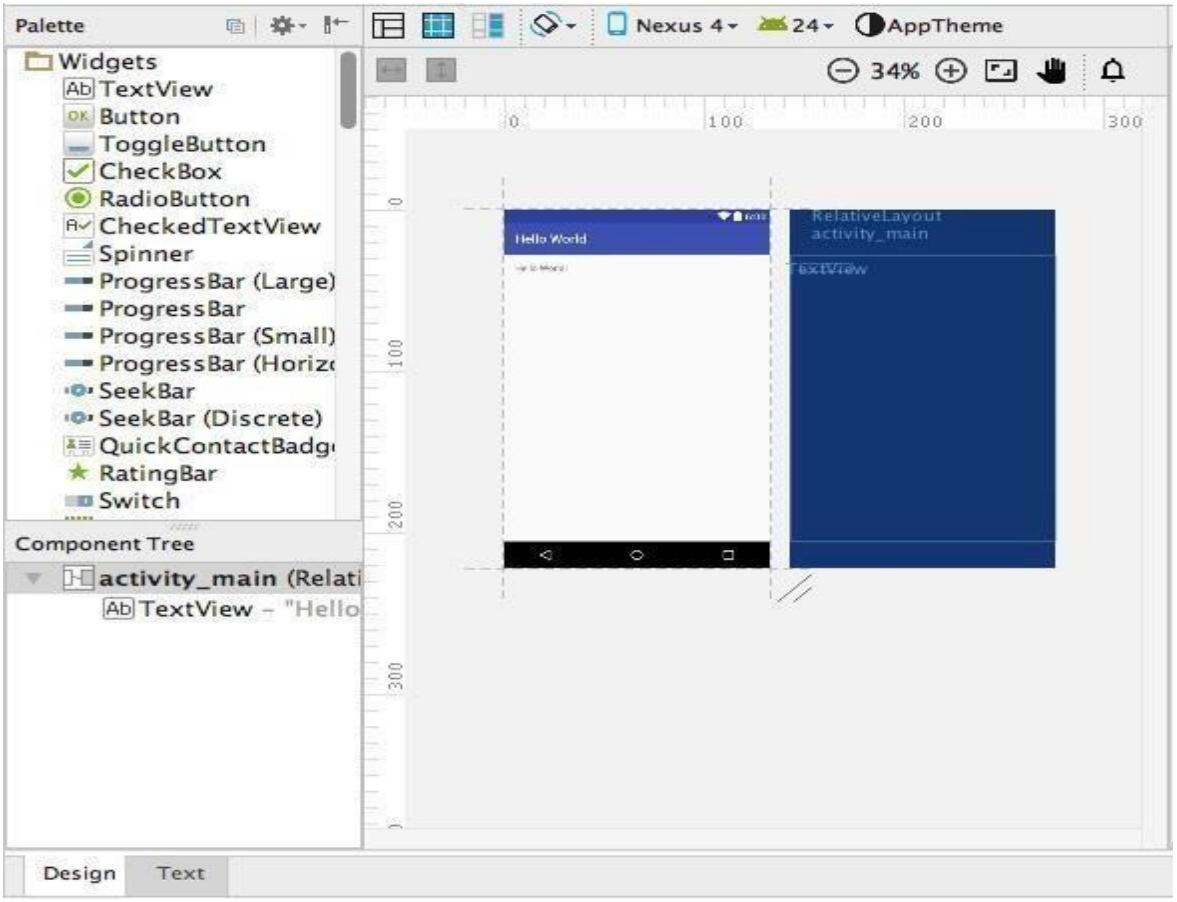


Fig 1.5: Design view of layout.

1.3 Introduction to Project

Job Seeker Android App is mainly developed for user to get the live status status of all the job vacancy in one app. Android app helps the user toknow about the vacant update regularly with advanced feature and functionality Job Seeker Android App Source Code gives accurate industrial trends and demand in their user hand. Our Android Job Seeker Application is not only for the people who are looking for job opportunity but also useful for employers to advertise Job vacancies in their organization. The user can register their account with their valid mail id and password and other details and they can login to their account using Email id and password. Also they can save their interested job for future reference.

1.3.1 Problem Statement

1. Develop an Android application that helps the user to know the Job vacancies in various field.
2. The app should allow users to input their details and job interest.
3. Also employers can advertise the Job vacancies to attract the employees.
4. Also person can apply to those job which are advertised by employers based on their interest.

1.3.2 Motivation and Objectives

Job seeker aim to maintain real-time information on current job trends and job vacancies, so to enhance employability . Job Seeker project strives to secure employment that aligns with individual's Career goals and interest.

1.3.3 Project Applications

1. Easy to use and helpful in real time situations.
2. It is easy to access as it is made available for all the android users to use it effectively.
3. The users can save time when they want are searching for job opportunity by having all the vacancy data available in their hand.
4. Employers can hire efficient candidates by posting job description.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

- **Processor**:X86 Compatible processor with 1.7Ghz clock
- **Speed RAM**:4GB or greater
- **Hard disk**:400 GB min
- **Monitor**: VGA/SVGA
- **Keyboard**:104 keys standard
- **Mouse**:2/3 button Optical/mechanical

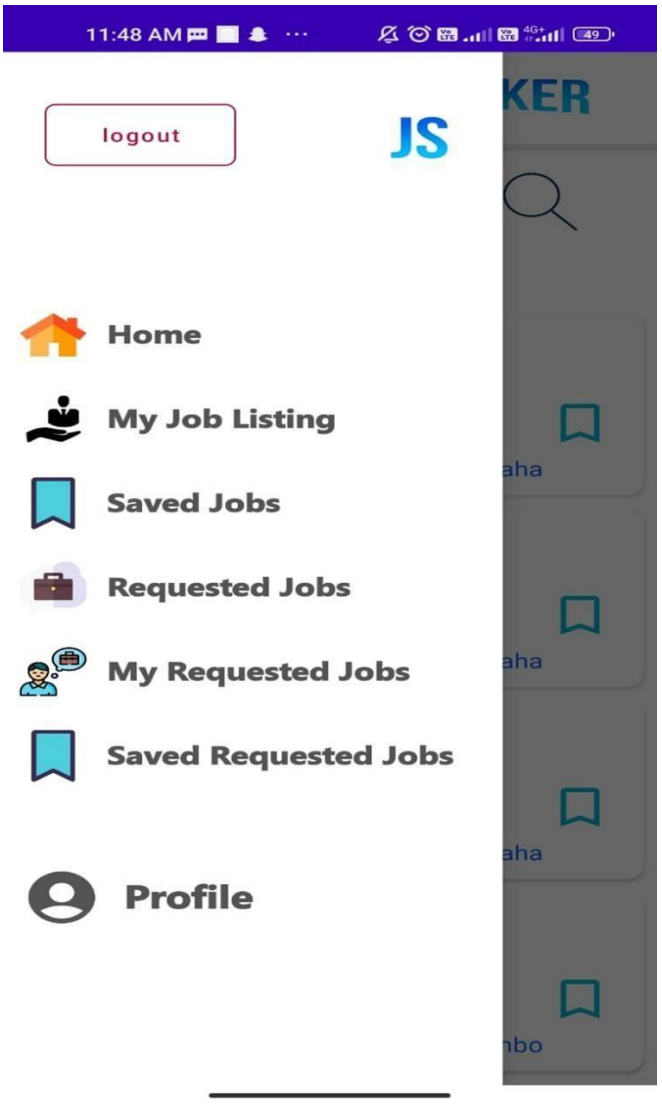
2.2 Software Requirements

- **Operating system**: windows 11
- **Software used**: Android Studio
- **Front end**: XML **Backend**: JAVA

CHAPTER 3

DESIGN

3.1 User Interface Design



3.2 ER Diagram /Data Flow Diagram

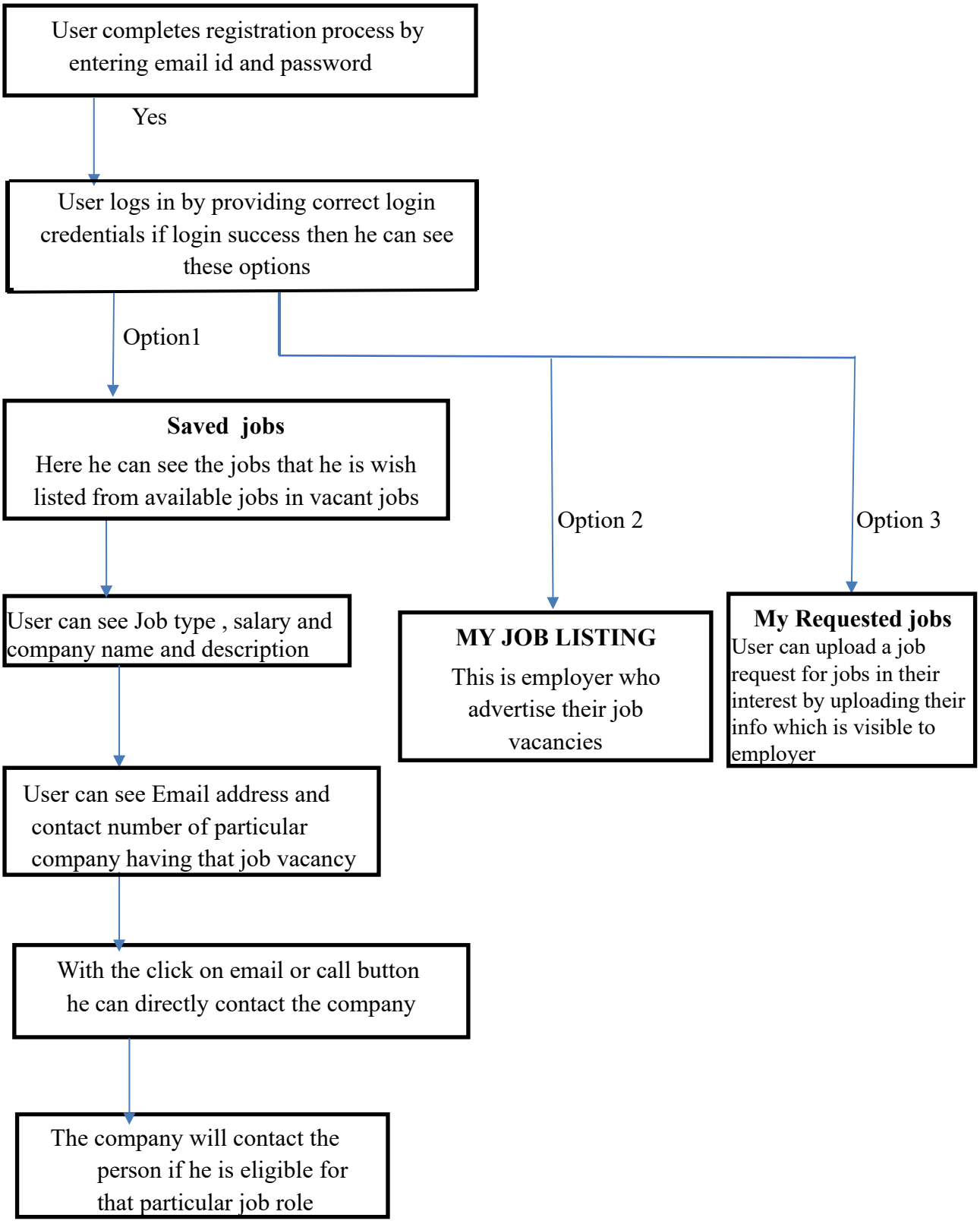


Fig 3.2: Data Flow Diagram for Job Seeker

CHAPTER 4

IMPLEMENTATION

4.1 Source Code

4.1.1 Java code

```
package com.example.mysaved;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.squareup.picasso.Picasso;


public MyReqAdapter(@NonNull View itemView) {
   super(itemView);
   reqjobtitle = itemView.findViewById(R.id.d_tv_reqjobname2);
   reqgender = itemView.findViewById(R.id.d_tv_reqjobgender2);
   reqage = itemView.findViewById(R.id.d_tv_reqjobage2);
   imageView = itemView.findViewById(R.id.d_img_reqprofile2);
   remove = itemView.findViewById(R.id.d_img_delete2);
   reqjob_card = itemView.findViewById(R.id.d_reqjobcard2);
   remove.setOnCheckedChangeListener(this);
```

```
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if (FirebaseAuth.getInstance().getCurrentUser() == null){
                Toast.makeText(context, "Please Login First", Toast.LENGTH_SHORT).show();
                context.startActivity(new Intent(context,LoginActivity.class));
                return;
            }
            DatabaseReference dbremove = FirebaseDatabase.getInstance().getReference("user")
                    .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                    .child("SaveRequestedjobs");
            int position = getAbsoluteAdapterPosition();
            ReqJobList reqJobList = list.get(position);
            String reqjob = reqJobList.reqid + reqJobList.reqjobid;
            if (b){
                dbremove.child(reqjob).setValue(null);
                compoundButton.setChecked(false);
            }
        }
        @Override
        public void onClick(View view) {
            int position = getAbsoluteAdapterPosition();
            ReqJobList reqJobList = list.get(position);
            String userid = reqJobList.reqid;
            String jobid = reqJobList.reqjobid;
            Intent intent = new Intent(view.getContext(), ViewReqJob.class);
            intent.putExtra("user_id", userid);
            intent.putExtra("ReqId", jobid);
            view.getContext().startActivity(intent);
        }
    }
}
package com.example.mysaved;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
android.view.inputmethod.EditorInfo;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.Toast;
com.google.firebase.database.DataSnapshot;
com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class Homepage_new extends AppCompatActivity {
Button nav_logout, nav_login;
DrawerLayout drawerLayout;
    TextView nav_home_txt, nav_savedreqjobs_txt, nav_myjobs_txt,
nav_requestedjobs_txt, nav_myreqjobs_txt, nav_savedjobs_txt, nav_profile_txt;
FirebaseAuth fAuth;
String userID;
private RecyclerView recyclerView; private
DatabaseReference root; private
ViewHolder_Homepage viewHolder_homepage;
private ArrayList<HomeList> list;
//  private CheckBox save;
private SearchView search;
private ProgressBar load;
```

```
@Override protected void onCreate(Bundle
savedInstanceState) {
super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_homepage_new);

   drawerLayout = findViewById(R.id.drawer_layout);

   nav_login = findViewById(R.id.btn_nav_login); nav_logout
   = findViewById(R.id.btn_nav_logout);




   nav_myjobs_txt = findViewById(R.id.tv_nav_myjobs); nav_savedjobs_txt
   = findViewById(R.id.tv_nav_savedjobs); nav_requestedjobs_txt =
   findViewById(R.id.tv_nav_requestedjobs); nav_myreqjobs_txt =
   findViewById(R.id.tv_nav_myreqjobs); nav_savedreqjobs_txt =
   findViewById(R.id.tv_nav_savedreqjobs); nav_profile_txt =
   findViewById(R.id.tv_nav_profile);

   //get current instance of firebase authentication fAuth
   = FirebaseAuth.getInstance();

   //check if user is already logged in if
   (fAuth.getCurrentUser() != null) { userID =
   fAuth.getCurrentUser().getUid();
   nav_login.setVisibility(View.GONE);
   nav_logout.setVisibility(View.VISIBLE);
   } else { nav_logout.setVisibility(View.GONE);
      nav_login.setVisibility(View.VISIBLE);

   }
```

```
load = findViewById(R.id.d_progressBar4);
recyclerView = findViewById(R.id.d_recycle); recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(new LinearLayoutManager(Homepage_new.this));
root = FirebaseDatabase.getInstance().getReference();

root.child("create_job").addValueEventListener(new ValueEventListener() {
   @Override public void onDataChange(@NonNull DataSnapshot
   snapshot) { list.clear();
      for (DataSnapshot dataSnapshot : snapshot.getChildren()) { String
         id = dataSnapshot.getKey();
         for (DataSnapshot ds : dataSnapshot.getChildren()) {
            String jobid = ds.getKey();
            String date = ds.child("date").getValue(String.class);
            String name = ds.child("name").getValue(String.class);
            String title = ds.child("title").getValue(String.class);
            String salary1 = ds.child("salary1").getValue(String.class);
            String job_type = ds.child("job_type").getValue(String.class);
            String description = ds.child("description").getValue(String.class);
            String email1 = ds.child("email1").getValue(String.class);
            String phone1 = ds.child("phone1").getValue(String.class);
            String district = ds.child("district").getValue(String.class);
            String img = ds.child("img").getValue(String.class);

            HomeList home = new HomeList(id, jobid,date,name, title, salary1, job_type,
 email1, phone1, district, img);
            list.add(home);
            load.setVisibility(View.INVISIBLE); }
//             Collections.reverse(list);
            Collections.sort(list, new Comparator<HomeList>() {
```

```
@Override public int compare(HomeList homeList, HomeList
t1) { return homeList.date.compareToIgnoreCase(t1.date); }
viewHolder_homepage = new ViewHolder_Homepage(Homepage_new.this, list);
recyclerView.setAdapter(viewHolder_homepage);
viewHolder_homepage.notifyDataSetChanged();
}
}

@Override
public void onCancelled(@NonNull DatabaseError error) {
System.out.println("The read failed: " + error.getCode());
search = (SearchView) findViewById(R.id.d_search);
search.setQueryHint("Search Jobs");
search.setImeOptions(EditorInfo.IME_ACTION_DONE);

@Override public boolean
onQueryTextSubmit(String s) { return false;
}


@Override public boolean
onQueryTextChange(String s) {
viewHolder_homepage.getFilter().filter(s);
System.out.println(s); return false;
   } });
if(!isNetworkAvailable(
))
{ new AlertDialog.Builder(this)
.setIcon(android.R.drawable.ic_dialog_alert)
.setTitle("Internet Connection Alert")
.setMessage("Please Check Your Internet Connection")
.setPositiveButton("Close", null).show();
```

```
        } else if(isNetworkAvailable())
        {
                Toast.makeText(Homepage_new.this, "Welcome", Toast.LENGTH_LONG).show();
      }

}

        public boolean isNetworkAvailable() {
        ConnectivityManager connectivityManager =(ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
        if (connectivityManager != null) {
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        NetworkCapabilities capabilities =
        connectivityManager.getNetworkCapabilities(connectivityManager.getActiveNetwork()); if
        (capabilities != null) {
        if (capabilities.hasTransport(NetworkCapabilities.TRANSPORT_CELLULAR)) {
         return true;
        } else if (capabilities.hasTransport(NetworkCapabilities.TRANSPORT_WIFI)) {
         return true;
        } else if (capabilities.hasTransport(NetworkCapabilities.TRANSPORT_ETHERNET)) {
          return true;
        }
        }
                }
        }

    return false;

}

public void ClickMenu(View view) {
    //open drawer openDrawer(drawerLayout);
}
```

```
    public static void openDrawer(DrawerLayout drawerLayout) {
        //open drawer layer
        drawerLayout.openDrawer(GravityCompat.START);

    }

    //navigation drawer button functions public
void login(View view) {
        Intent i = new Intent(getApplicationContext(), UserloginActivity.class)
        .putExtra("from", "main"); startActivity(i);

    }

    public void logout(View view) { FirebaseAuth.getInstance().signOut();
        nav_logout.setVisibility(View.GONE);
        nav_login.setVisibility(View.VISIBLE);

    }

    //side_nav buttons public void navClickHome(View view) {
    nav_home_txt.setTextColor(ContextCompat.getColor(this,R.color.black));
    nav_myjobs_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
    nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
    nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
    nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
    nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
    nav_profile_txt.setTextColor(ContextCompat.getColor(this,R.color.nav));
        startActivity(new Intent(getApplicationContext(),Homepage_new.class));


    }
```

```
    public void navClickMyjobs(View view) {
        nav_home_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.black));
        nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_profile_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));

        startActivity(new Intent(getApplicationContext(), MyJobListings.class));


    }


    public void navClickSavedjobs(View view) {
        nav_home_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.black));
        nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_profile_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
startActivity(new Intent(getApplicationContext(), New_Saved_Jobs.class));

    }


    public void navClickRequestedjobs(View view) {
        nav_home_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.black));
        nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
```

```
        nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_profile_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));

         startActivity(new Intent(getApplicationContext(), NewReqJob_Homepage.class));
    }
    public void navClickMyreqjobs(View view) {
        nav_home_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.black));
        nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_profile_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        }
    }

    public void navClickSavedjobs(View view) {
        nav_home_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.black));
        nav_requestedjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_myreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_savedreqjobs_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
        nav_profile_txt.setTextColor(ContextCompat.getColor(this, R.color.nav));
startActivity(new Intent(getApplicationContext(), New_Saved_Jobs.class));
            }

        }

    }
```

CHAPTER 5

RESULT

Snap shot

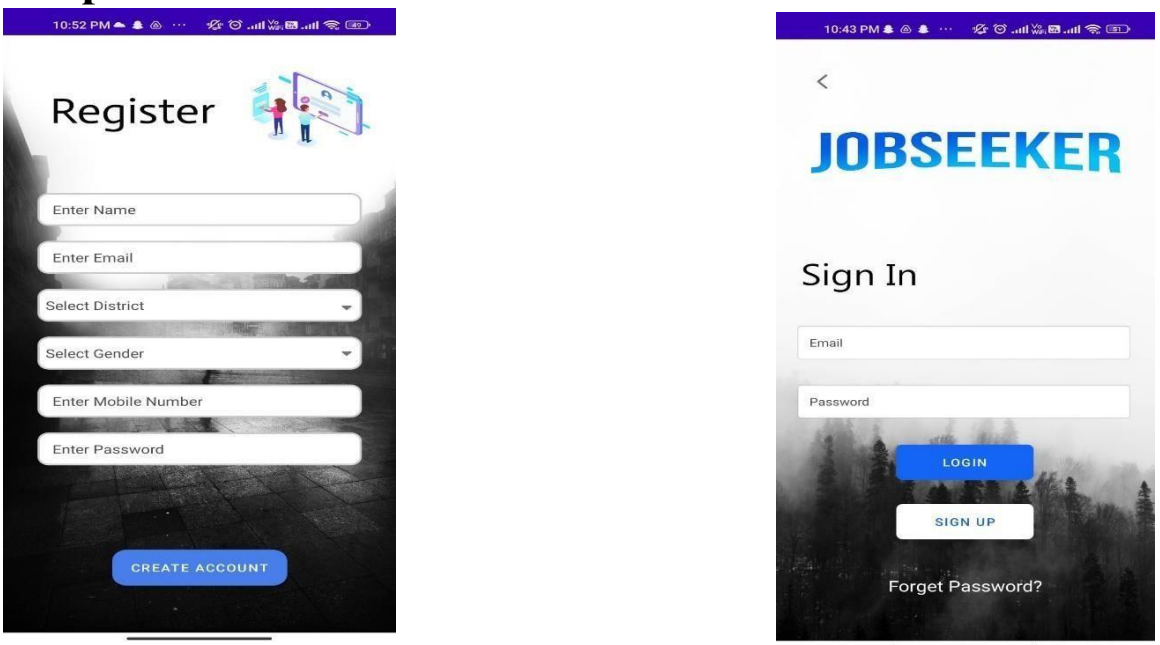


Fig 5.1: Registration and Login Process

The Registration process where user can enter Name, Maid id and password and other required details to create account and registered person login to their account using provided mail id and password

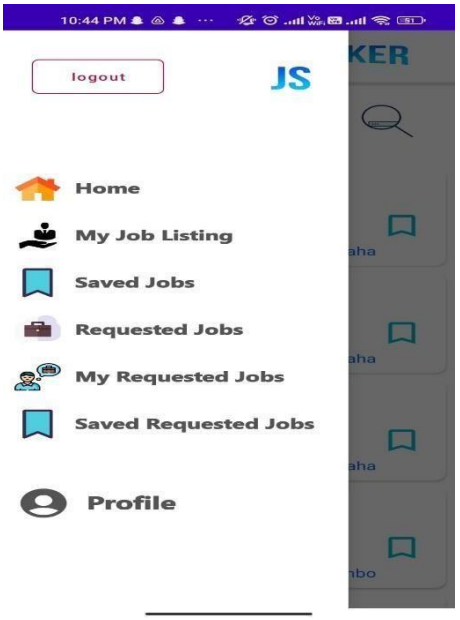


Fig 5.2: Available options

Once the user logs in they can see these page with these options

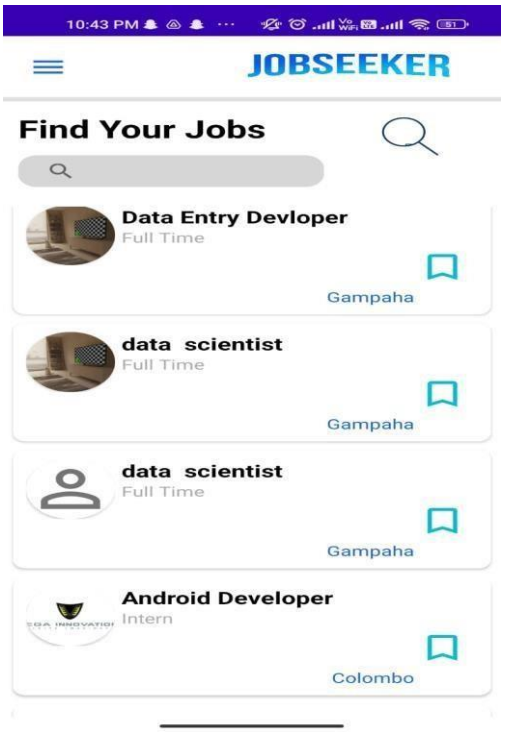


Fig 5.3: Home page

Here user can see all the job listings created by employers for hiring

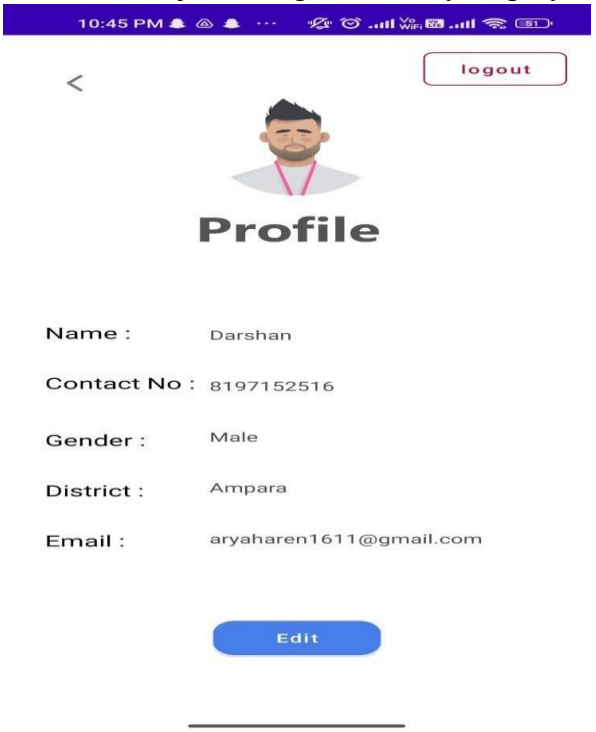


Fig 5.4: Profile Page

Here the User can see their profile details with edit and logout option

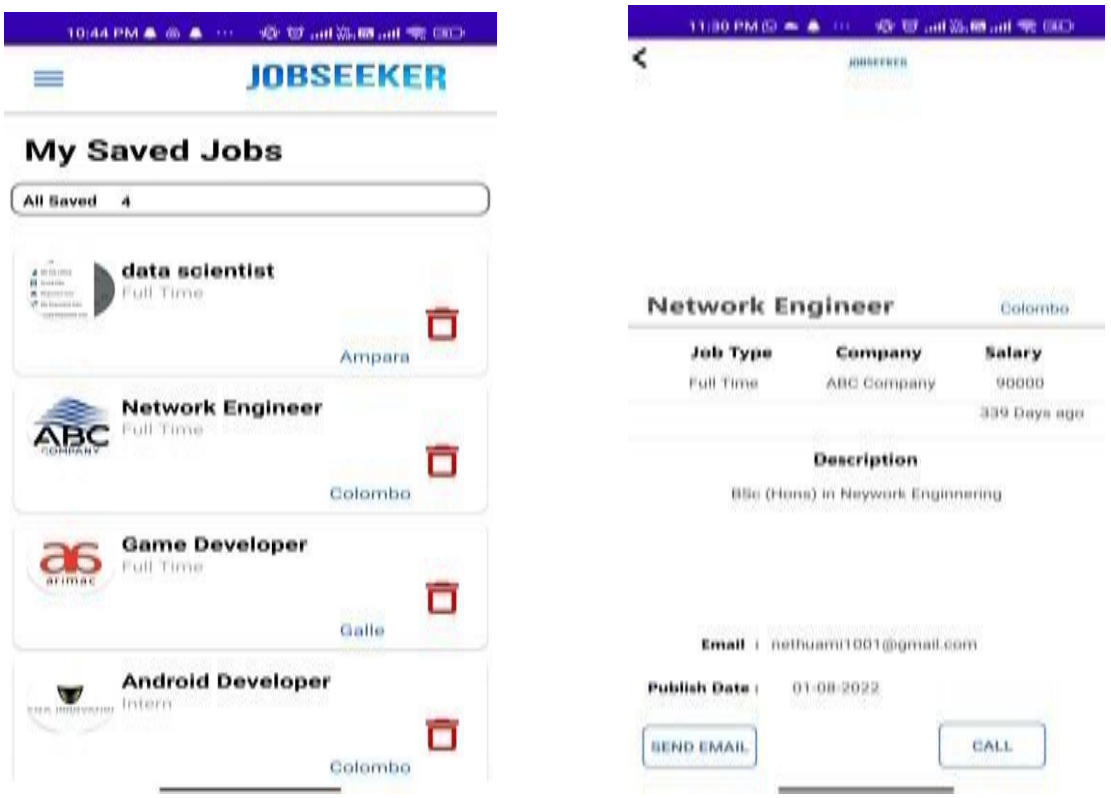
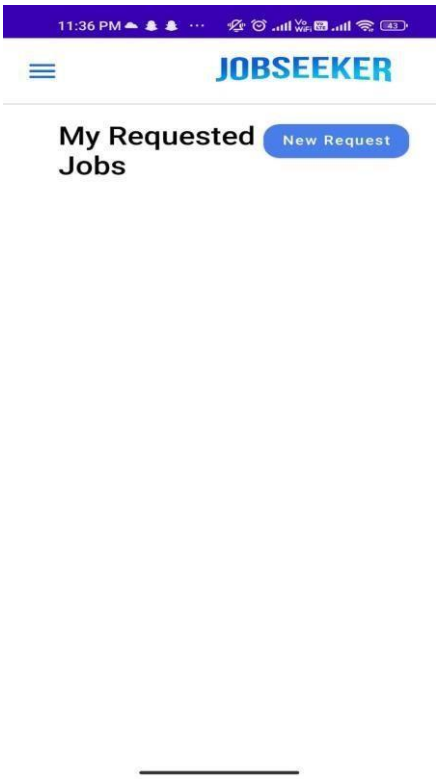


Fig 5.5: My Saved Jobs

Here in this page user can save jobs which they found useful and planning to apply for it later they can save it for future reference and apply



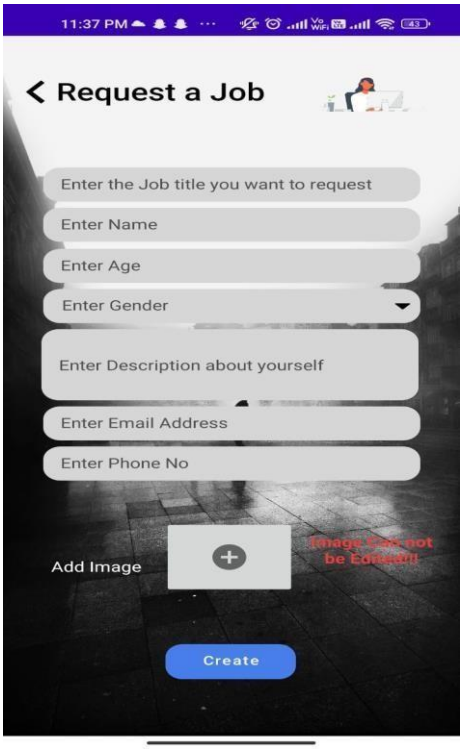


Fig 5.6: My Requested Jobs

Here user can post a job request. they can market themself by adding their educational qualifications, experience in the field and other qualifications to the job request

CONCLUSION & FUTURE ENHANCEMENT

Conclusion

` In conclusion, the development of our job seeking mobile app has been a significant milestone in addressing the challenges faced by job seekers. By offering a user-friendly interface, extensive job listings, and personalized recommendations, our app provides a comprehensive platform for individuals to explore career opportunities. The incorporation of advanced features such as resume building, skill assessments, and real-time notifications enhances the overall user experience. Through rigorous testing and feedback analysis, we have refined the app to ensure optimal performance and reliability. Moving forward, our job seeking mobile app holds great potential in revolutionizing the job market by connecting job seekers with relevant opportunities efficiently and effectively.

.

Future Enhancement

1. We will try improve our user interface design
2. Try to add authentication method such that no fraud happens
3. Implement secure data handling practices, such as encryption, to protect user data, especially when handling sensitive information such as currency conversion rates or user preferences.

Book References

[1] Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017.

[2] Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197.

[3] Dawn Griffiths and David Griffiths, "Head First Android Development", 1st Edition,O'relly.

[4] SPD Publishers, 2015. ISBN-13: 978-9352131341.

[5] Bill Phillips, Chris Stewart and Kristin Marsicano, "Android Programming: The Big Nerd Ranch Guide", 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054.

Web References

[1] Android Developers https://developer.android.com/

[2] Android Tutorial https://www.tutorialspoint.com/android/index.htm

[3] Android Tutorial https://www.w3schools.in/category/android-tutorial/

[4] Java Tutorial https://www.javatpoint.com/java-tutorial

[5] Java Tutorial https://www.w3schools.com/java/

Table of Contents