**1. INTRODUCTION**

The actual problem is to create a application for the Consultant, is developed for creating an interactive job vacancy for candidates. This application is to be conceived in its current form as a dynamic site-requiring constant updates both from the seekers as well as the companies. On the whole the objective of the project is to enable jobseekers to place their resumes and companies to publish their vacancies. It enables jobseekers to post their resume, search for jobs, view personal job listings. It will provide various companies to place their vacancy profile on the site and also have an option to search candidate resumes. Apart from this there will be an admin module for the customer to make changes to the database content.

The users of this system are searching for job, registration their personal, educational, skills, project and resume details. This system is designed such a way that the users can easily interact with the system with minimum knowledge about the application

Firstchapter explains the exact definition of the problem and evolves out with the feasibility study of the application part.

Second chapter System Analysis which deals about the Hardware and Software Specifications, and Software Requirement Specification.

Third chapter describes the System Design, under this two levels of designs, they are

High level design (Data design, functional & interface design).

Low level design (Pseudo code & detail description of functions).

Next chapter fully deals about Testing and Implementation of the whole project.

Fifth chapter deals the Conclusion and Enhancements of the system.

Last chapter deals about the reference of this Project.

**2.SYSTEM REQUIREMENTS**

The development of this project deals with the following environment

2.1 Hardware requirements

2.2 Software requirements

**2.1 Hardware Requirements:**

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware, the size and the capacity requirements are also important.

The job explorer can be efficiently run on latest android versions like nougat(android 7.0) ,orieo (android 8.0) .

.

* processor -------- 1.7 ghz or above
* RAM Capacity -------- 1gb minimum.

**2.2 Software Requirements:**

As our project is an android application we will be designing this in the computer/laptop using android studios (2.3.3) and use database in order to maintain records of job seeker and job provider.

* Operating System -------- Android Versions Greater Than 5.0
* Database Server -------- Wamp Server.
* Database Connectivity -------- JDBC.
* Front end -------- JAVA,PHP.
* Back end -------- Database

**3.SYSTEM ANALYSIS**

**3.1 Existing system:**

Before creating this application , all jobseekers to send their resumes or information through postal mails or they use person to person contacts with each other. It will take long time to send their requirements through this type of communications.

Here there May error occurs in the process. The administration faces the problems to collect all the information from clients and consultants to analyze the requirement in the corresponding Clients. Administration has to send requirements information to different consultants and jobseekers.

**3.2 Proposed System:**

Here all job seekers send their resumes Or information through our application .It does not consume much of time .It is very easier to modify if any error occurs in the process. It is also very easier to administrator to collect information from clients and consultants.

**3.3 Users of the system:**

The users of this system are administrator, clients and company. This system is designed such a way that the users can easily interact with the system with minimum knowledge about the application and company rules.

**3.4 Scope:**

* To maintain job seeker and job provider records.
* Provide the categories of jobs.
* To maintain address and details of the company.
* To keep the application as simple as possible so that it becomes easy to use.

System Analysis therefore understands such problems and proposes a new system in which the above problems are rectified.

## Class-Diagram:

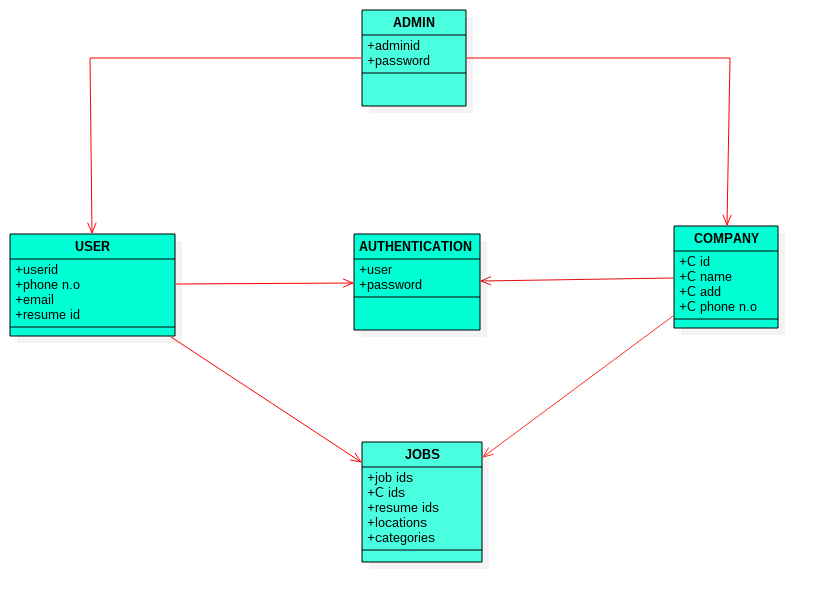


Fig 4.1:Represents classes that are used in the project.

The above class diagram contains five classes they are : Admin ,User

Authentication, Company, Jobs.

**USERCASE DIAGRAM**

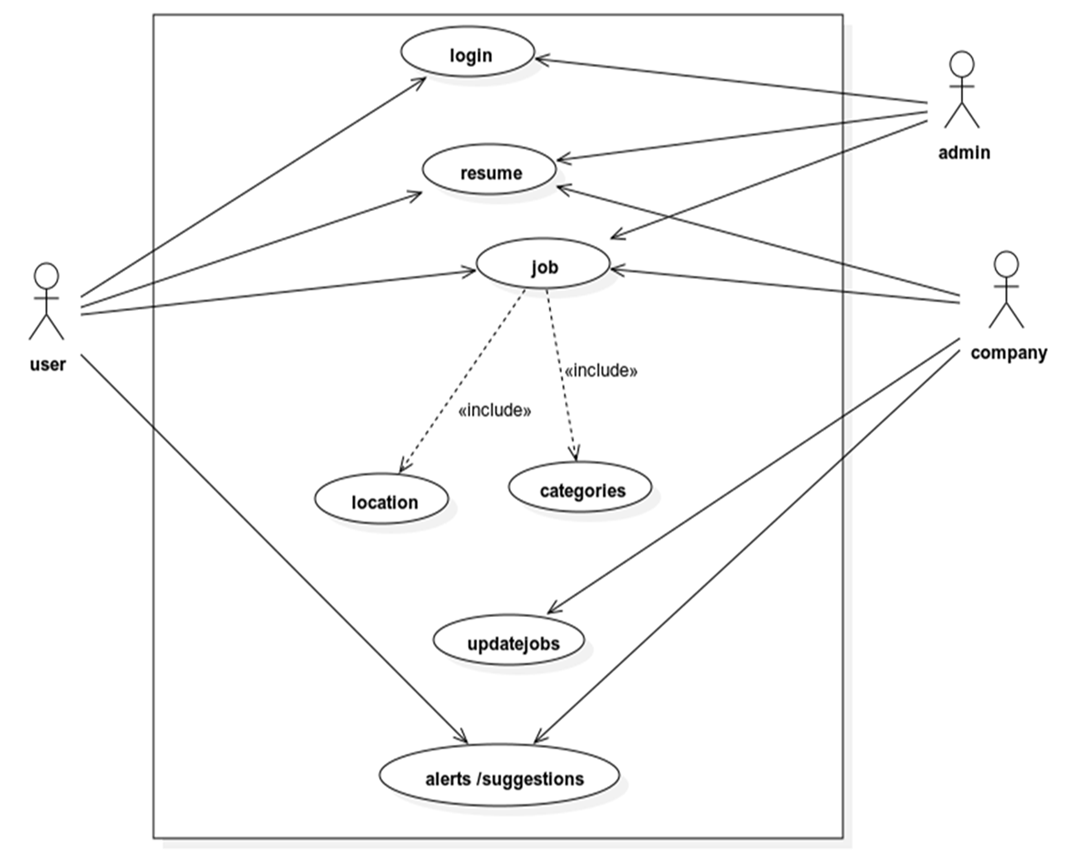


Fig4.2:Represents the flow of activity occur in 'JOB EXPLORER' between user, admin and company.

**ACTIVITY DIAGRAM**

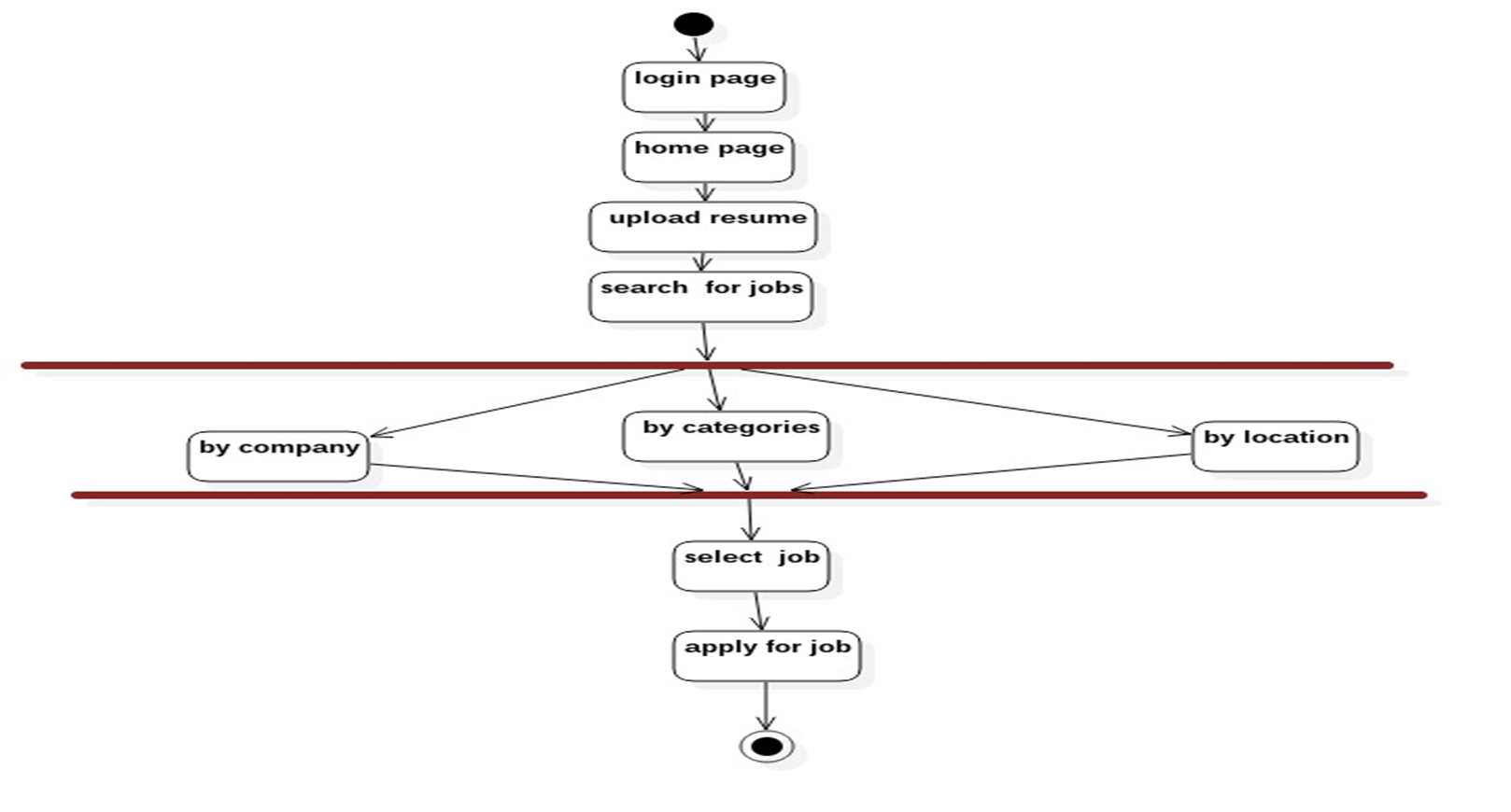
****

Fig 4.3:flow chart represents activities from login page to apply for job.

**SEQUENCE DIAGRAM**

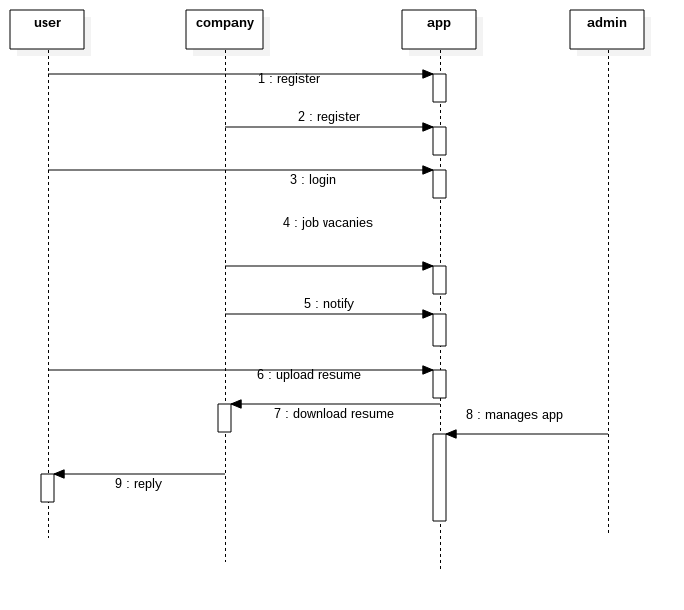


Fig 4.4:Show how classes are interacted with each other in what order.

## 5.sample code:

**# for front end:**

Main activity

import android.content.Intent;;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import java.util.HashMap;

import info.androidhive.loginandregistration.R;

import info.androidhive.loginandregistration.helper.SQLiteHandler;

import info.androidhive.loginandregistration.helper.SessionManager;

public class MainActivity extends AppCompatActivity

{

private TextView txtName;

private TextView txtEmail;

private Button btnLogout;

private Button btnjob;

private SQLiteHandler db;

private SessionManager session;

@Override protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

txtName = (TextView) findViewById(R.id.name);

txtEmail = (TextView) findViewById(R.id.email);

btnLogout = (Button) findViewById(R.id.btnLogout);

btnjob= (Button) findViewById(R.id.btnjob);

db = new SQLiteHandler(getApplicationContext());

// session manager

session = new SessionManager(getApplicationContext());

if (!session.isLoggedIn())

{

logoutUser();

}

//Fetching user details from SQLite

HashMap<String, String> user = db.getUserDetails();

String name = user.get("name");

String email = user.get("email");

// Displaying the user details on the screen

txtName.setText(name);

txtEmail.setText(email)

;

// Logout button click event

btnLogout.setOnClickListener(new View.OnClickListener(){@Override

public void onClick(View v) {

logoutUser();}

});

}

/\*\*\* Logging out the user. Will set isLoggedIn flag to false in shared

\* preferences Clears the user data from sqlite users table\* \*/

private void logoutUser() session.setLogin(false);

db.deleteUsers();

// Launching the login activity

Intent intent = new Intent(MainActivity.this, LoginActivity.class);

startActivity(intent);

finish();

}

public void onButtonClick(View view) {

if (view.getId()==R.id.btnjob) {

Intent intent = new Intent (MainActivity.this,category.class);

startActivity(intent);

}

}

}

**#job search :**

package info.androidhive.loginandregistration.activity;

import android.os.Bundle;

import android.support.design.widget.FloatingActionButton;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.DefaultItemAnimator;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.Toolbar;

import android.view.View;

import info.androidhive.loginandregistration.m\_MySQL.Downloader;

import info.androidhive.loginandregistration.R;

public class jobsearch extends AppCompatActivity {

final static String urlAddress="http://192.168.0.4/android\_login\_api/se.php";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.jobsearch);

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

final RecyclerView rv= (RecyclerView) findViewById(R.id.rv);

rv.setLayoutManager(new LinearLayoutManager(this));

rv.setItemAnimator(new DefaultItemAnimator());

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

new Downloader(jobsearch.this,urlAddress,rv).execute();

}

});

}

}

nt(MainActivity.this,category.class);

startActivity(intent);

}}}

**#login activity**

package info.androidhive.loginandregistration.activity;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import com.android.volley.Request.Method;

import com.android.volley.Response;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.StringRequest;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.HashMap;

import java.util.Map;

import info.androidhive.loginandregistration.R;

import info.androidhive.loginandregistration.app.AppConfig;

import info.androidhive.loginandregistration.app.AppController;

import info.androidhive.loginandregistration.helper.SQLiteHandler;

import info.androidhive.loginandregistration.helper.SessionManager;

public class LoginActivity extends Activity {

private static final String TAG = RegisterActivity.class.getSimpleName();

private Button btnLogin;

private Button btnLinkToRegister;

private EditText inputEmail;

private EditText inputPassword;

private ProgressDialog pDialog;

private SessionManager session;

private SQLiteHandler db;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

inputEmail = (EditText) findViewById(R.id.email);

inputPassword = (EditText) findViewById(R.id.password);

btnLogin = (Button) findViewById(R.id.btnLogin);

btnLinkToRegister = (Button) findViewById(R.id.btnLinkToRegisterScreen);

// Progress dialog

pDialog = new ProgressDialog(this);

pDialog.setCancelable(false);

// SQLite database handler

db = new SQLiteHandler(getApplicationContext());

// Session manager

session = new SessionManager(getApplicationContext());

// Check if user is already logged in or not

if (session.isLoggedIn()) {

// User is already logged in. Take him to main activity

Intent intent = new Intent(LoginActivity.this, MainActivity.class);

startActivity(intent);

finish();

}

// Login button Click Event

btnLogin.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) {

String email = inputEmail.getText().toString().trim();

String password = inputPassword.getText().toString().trim();

// Check for empty data in the form

if (!email.isEmpty() && !password.isEmpty()) {

// login user

checkLogin(email, password);

} else {

// Prompt user to enter credentials

Toast.makeText(getApplicationContext(),

"Please enter the credentials!", Toast.LENGTH\_LONG)

.show();

}

}

});

// Link to Register Screen

btnLinkToRegister.setOnClickListener(new View.OnClickListener()

{

public void onClick(View view) {

Intent i = new Intent(getApplicationContext(),

RegisterActivity.class);

startActivity(i);

finish();

}

});

}

/\*\*

\* function to verify login details in mysql db

\* \*/

private void checkLogin(final String email, final String password) {

// Tag used to cancel the request

String tag\_string\_req = "req\_login";

pDialog.setMessage("Logging in ...");

showDialog();

StringRequest strReq = new StringRequest(Method.POST,

AppConfig.URL\_LOGIN, new Response.Listener<String>() {

@Override

public void onResponse(String response) {

Log.d(TAG, "Login Response: " + response.toString());

hideDialog();

try {

JSONObject jObj = new JSONObject(response);

boolean error = jObj.getBoolean("error");

// Check for error node in json

if (!error) {

// user successfully logged in

// Create login session

session.setLogin(true);

// Now store the user in SQLite

String uid = jObj.getString("uid");

JSONObject user = jObj.getJSONObject("user");

String name = user.getString("name");

String email = user.getString("email");

String created\_at = user

.getString("created\_at");

// Inserting row in users table

db.addUser(name, email, uid, created\_at);

// Launch main activity

Intent intent = new Intent(LoginActivity.this,

MainActivity.class);

startActivity(intent);

finish();

} else {

// Error in login. Get the error message

String errorMsg = jObj.getString("error\_msg");

Toast.makeText(getApplicationContext(),

errorMsg, Toast.LENGTH\_LONG).show();

}

} catch (JSONException e) {

// JSON error

e.printStackTrace();

Toast.makeText(getApplicationContext(), "Json error: " + e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

}, new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

Log.e(TAG, "Login Error: " + error.getMessage());

Toast.makeText(getApplicationContext(),

error.getMessage(), Toast.LENGTH\_LONG).show();

hideDialog();

}

}) {

@Override

protected Map<String, String> getParams() {

// Posting parameters to login url

Map<String, String> params = new HashMap<String, String>();

params.put("email", email);

params.put("password", password);

return params;

}

};

// Adding request to request queue

AppController.getInstance().addToRequestQueue(strReq, tag\_string\_req);

}

private void showDialog() {

if (!pDialog.isShowing())

pDialog.show();

}0private void hideDialog() {

if (pDialog.isShowing())

pDialog.dismiss(}

}

**# back end:**

<?php

$host='127.0.0.1';

$username='root';

$pwd='';

$db="android\_api";

$con=mysqli\_connect($host,$username,$pwd,$db) or die('Unable to connect');

if(mysqli\_connect\_error($con))

{

echo "Failed to Connect to Database ".mysqli\_connect\_error();}

$sql="SELECT \* FROM jobs";

$result=mysqli\_query($con,$sql);

if($result)

{while($row=mysqli\_fetch\_array($result))

{$data[]=$row; }

print(json\_encode($data));

}

mysqli\_close($con);

?>

**#job categories:**

package info.androidhive.loginandregistration.activity;

import android.os.Bundle;

import android.support.design.widget.FloatingActionButton;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.DefaultItemAnimator;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.Toolbar;

import android.view.View;

import info.androidhive.loginandregistration.m\_MySQL.Downloader;

import info.androidhive.loginandregistration.R;

public class jobsearch extends AppCompatActivity {

final static String urlAddress="http://192.168.0.4/android\_login\_api/se.php";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.jobsearch);

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

final RecyclerView rv= (RecyclerView) findViewById(R.id.rv);

rv.setLayoutManager(new LinearLayoutManager(this));

rv.setItemAnimator(new DefaultItemAnimator());

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

new Downloader(jobsearch.this,urlAddress,rv).execute();

}

});

}}

## 6.SYSTEM TESTING:

User Registration :

To use the app user should register first and then login There are three fields for every user in the registration page .Every field should be filled by the user. When user completes his/her registration then they can login.

User Login:

User should login with the registered email and password. If the user credentials are right you can login.

Log Out:

User can logout by clicking on logout button.

Categories:

User can see the different categories of jobs by clicking on the job categories button. They can select any category and know the jobs available in that. They can then select any of the job and know location ,company, qualifications..etc.

7.Output Screens

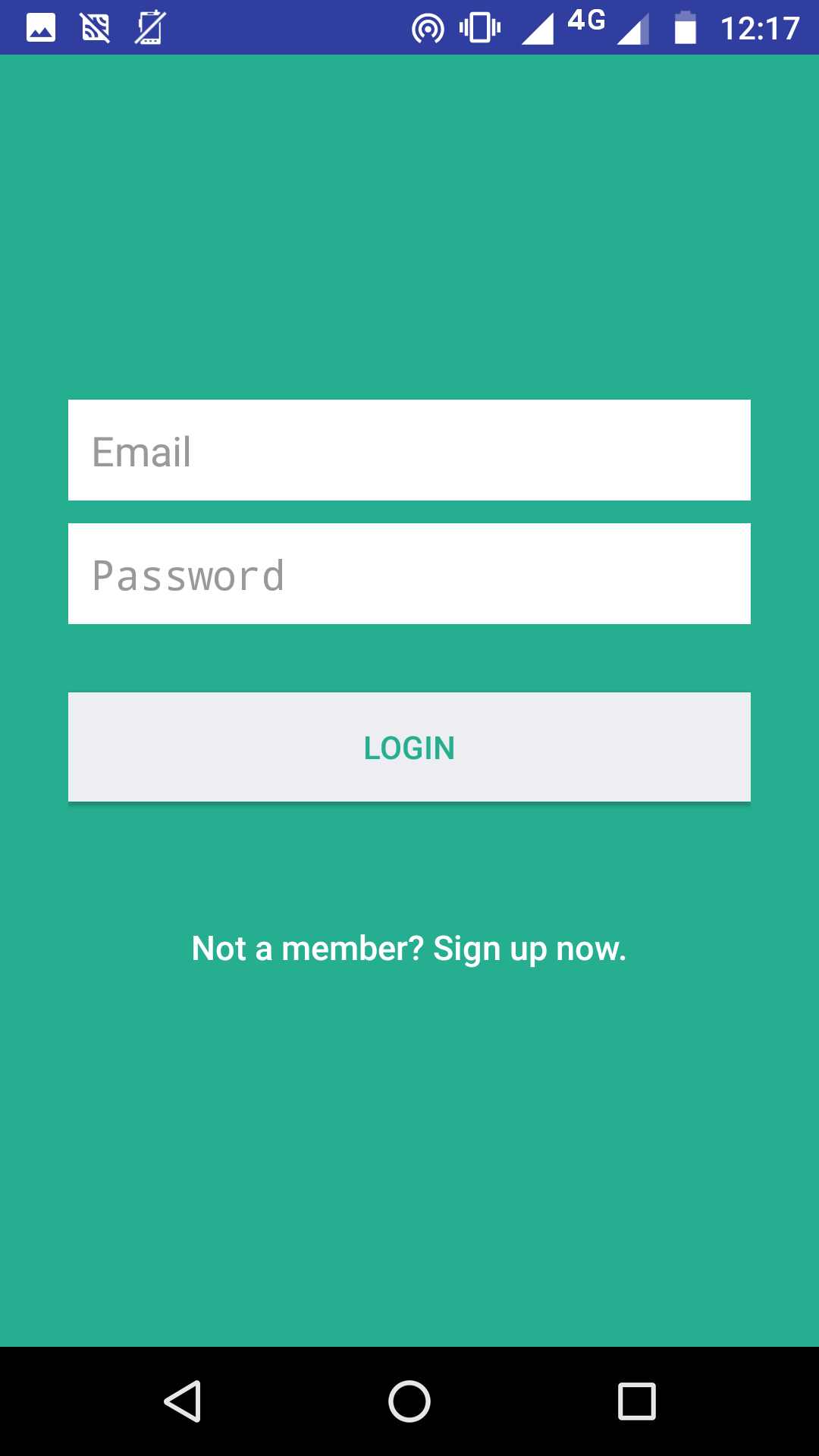
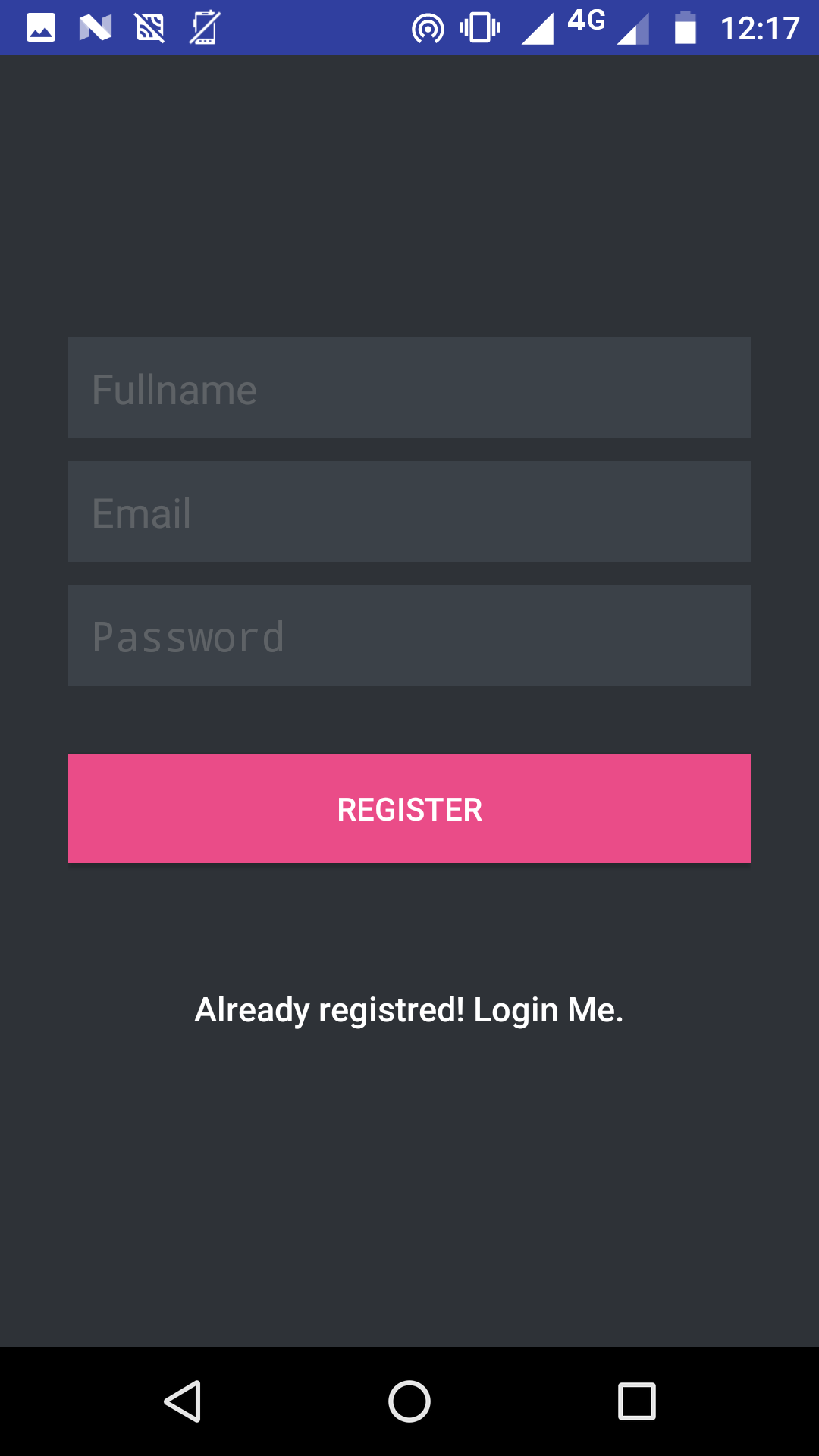


Fig7.1: Represents login page where user can login.In this page a new user can also login by sign up option as he is a new user.

On clicking sign up now registration page will be opened.



.

Fig 7.2:Represents registration page where new user can register .

Here on clicking register button login page will be opened .

If the details that you enter in registration page are matched with login page details then only next page will be opened.

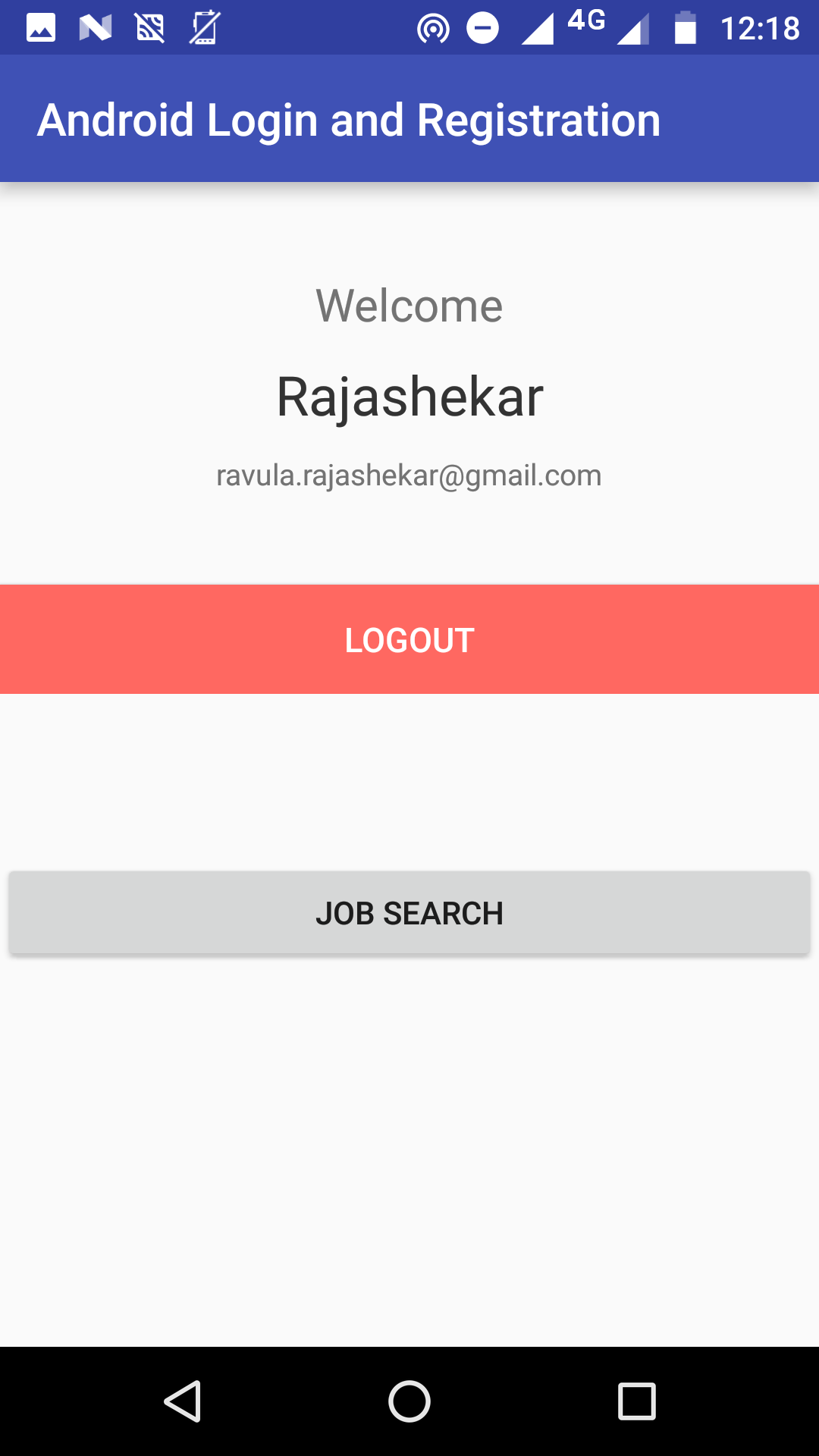


fig 7.3: Represents both for the user to search the job or to exit from app.

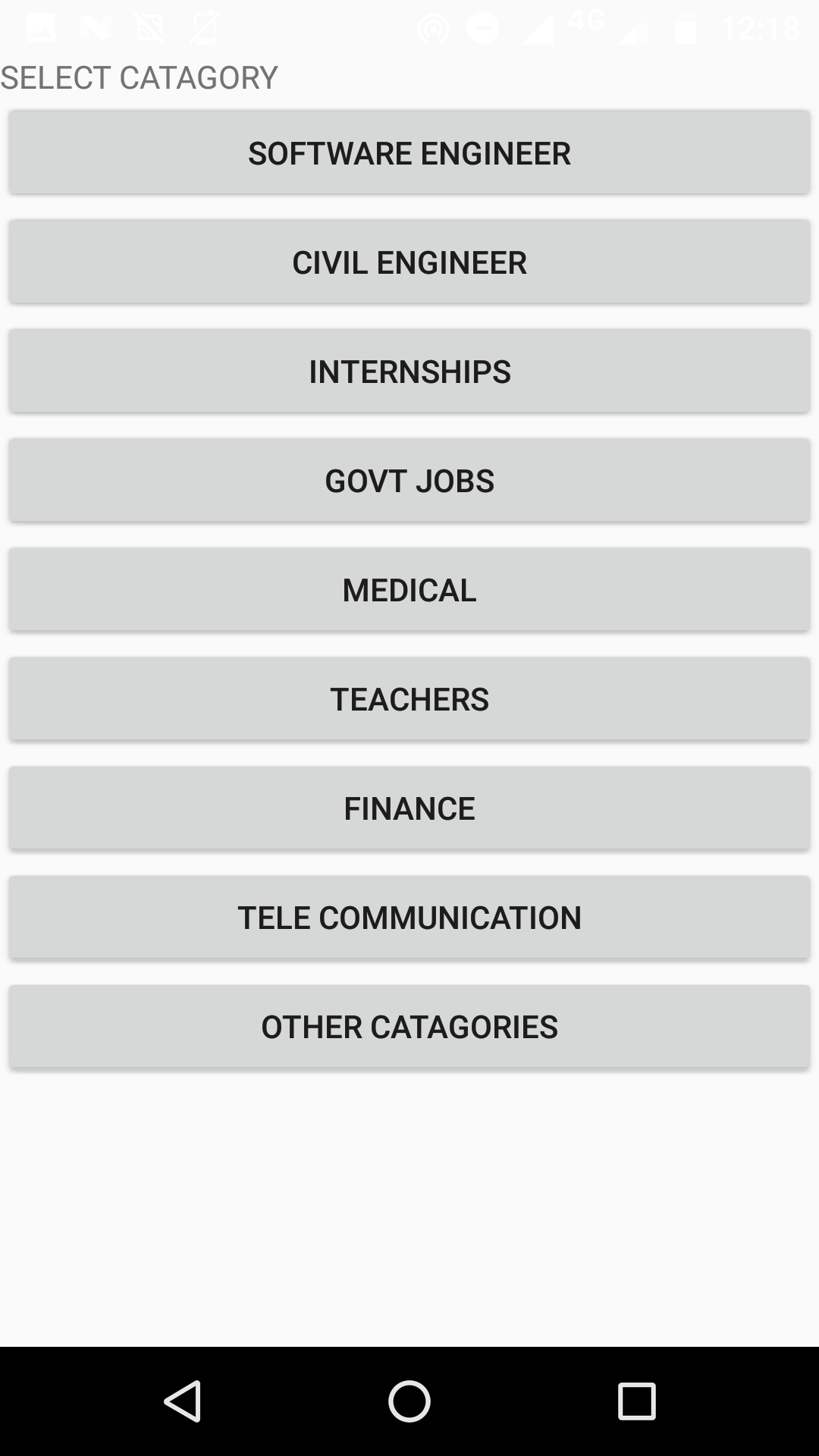


Fig 7.4:represents different categories of jobs.

Where on clicking on each category required jobs will be appeared.



fig 7.5: Represents the internships that are available.

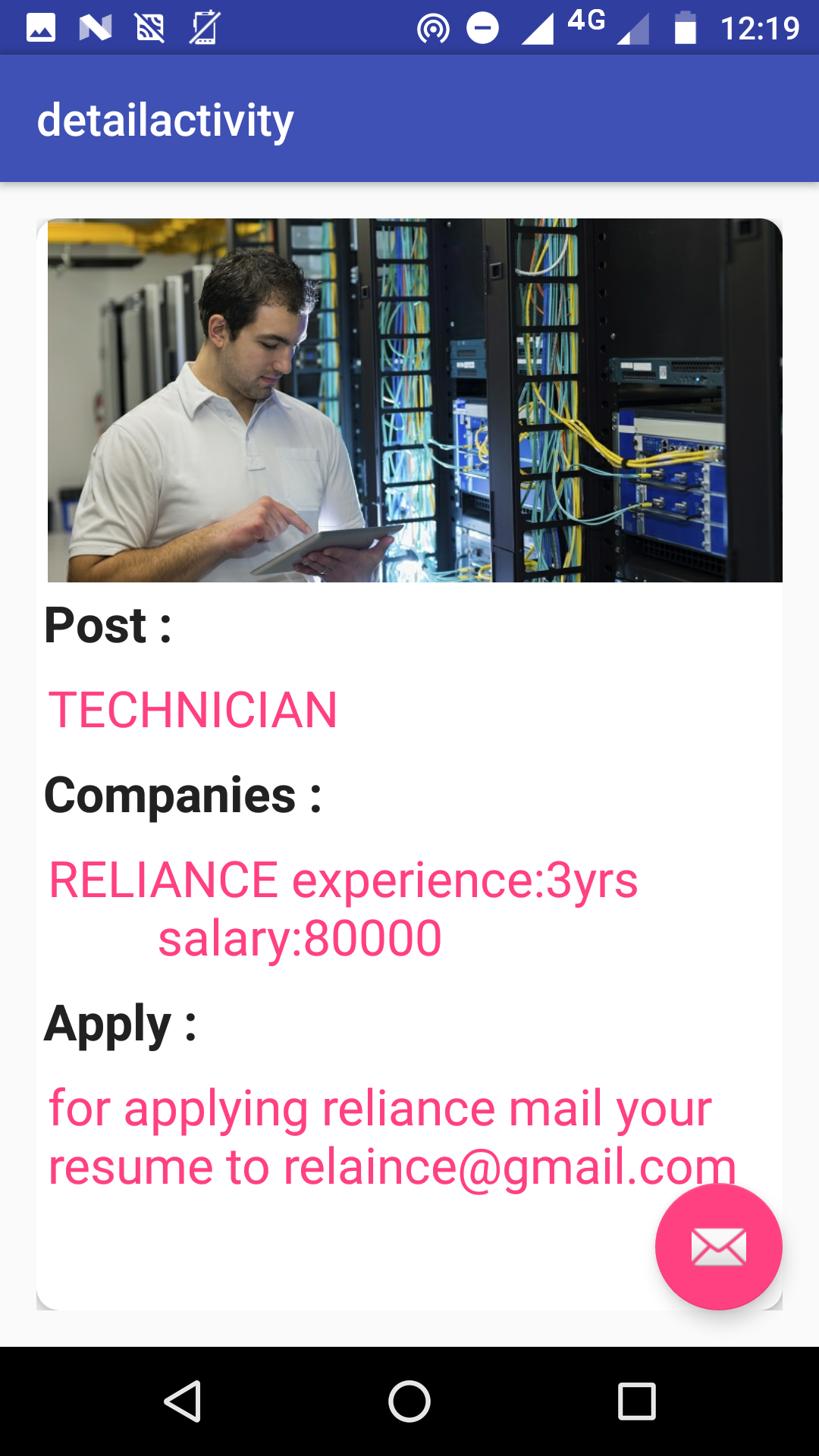


Fig 7.6: represents a job of technician by reliance company .

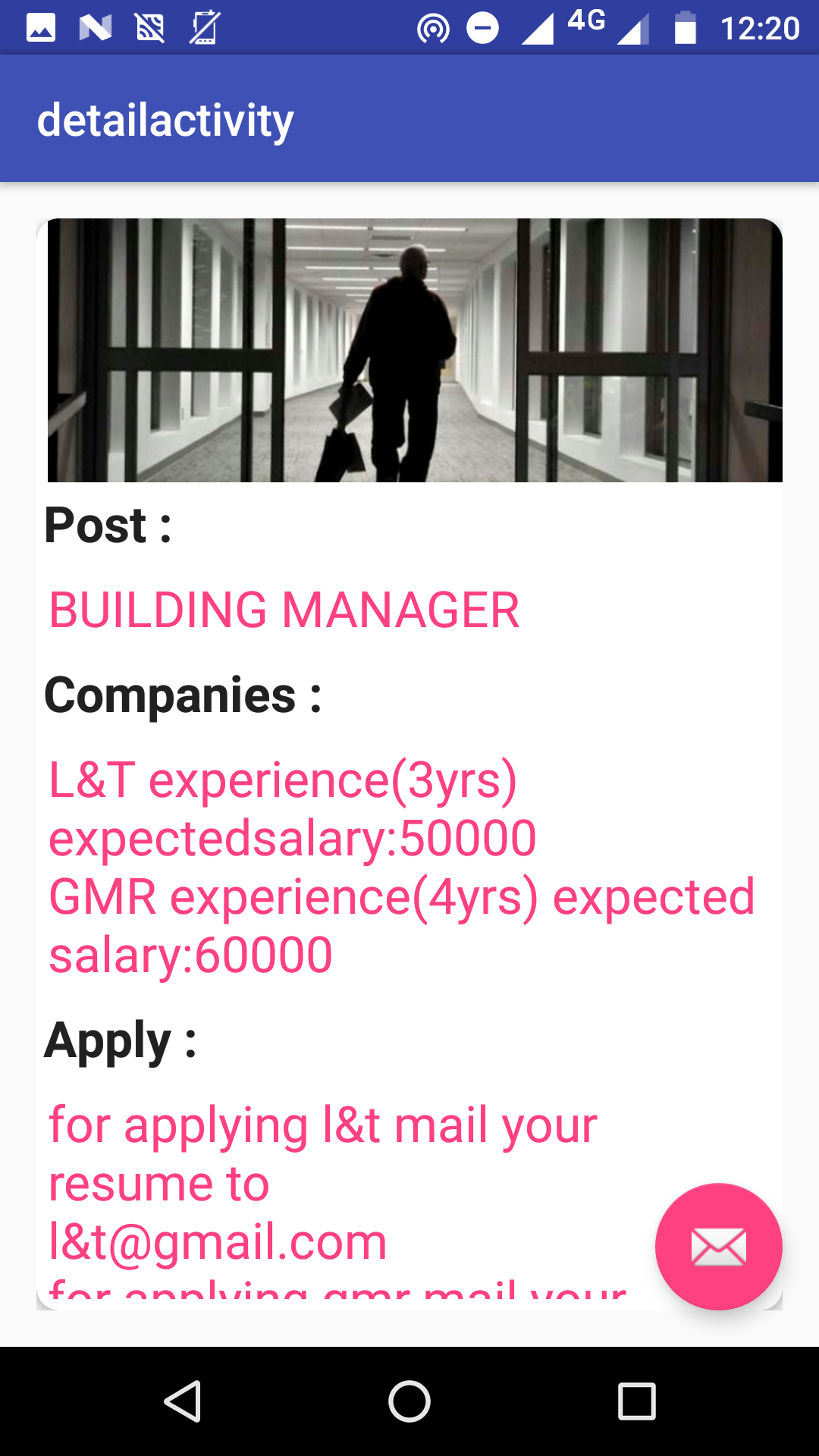


Fig 7.7: represents a post of job as Bank Manager in L&T Company

**Conclusion**

This application has been developed successfully incorporate all the requirements. Appropriate care has taken during database design maintain database integrity and to avoid redundancy of data. This application was developed in such a way that any further modifications needed can be easily done. User feels freely while using this app. In this all technical complexities are hidden. This app is more user friendly.

There is always a room for improvement in any software package, however good and efficient it may be. The important thing is that the application should be flexible enough for further modifications. Considering this important factor, the app is designed in such a way that the provisions are given for further enhancements. At present this app provides all the information using static pages and reservation forms.

In future we can enhance our project by providing options like. Include many sites information.

### References

1.www.youtube.com

2.www.wikipedia.org

3.www.androidhive.com