

**MAHATMA GANDHI MEMORIAL COLLEGE**

**UDUPI-576102**

**Accredited by NAAC with “A+” Grade (CGPA 3.36)**



**PROJECT REPORT ON 2022-2023**

**“PG-FINDER”**

**DEVELOPED BY**

**Reshma R Bhat**

**Reg.No:201161522136**

**Shreethu K S**

**Reg.No:201161522185**

**Sinchana C**

**Reg.No:201161522215**

**Under the guidance of**

**“Prof. Shilpa Shanubhag”**

**Department of computer science**

**Submitted To Mangalore University in partial fulfillment of the**

**Award of**

**Bachelor of Computer Application**

**MANGALORE UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE**

**MAHATMA GANDHI MEMORIAL COLLEGE**

## **DECLARATION**

We are the undersigned **Reshma R Bhat, Shreethu K S, Sinchana C** students of Bachelor of Computer Application, Mahatma Gandhi Memorial College, Kunjibettu, Udupi hereby declared by the project work, presented in this report is our own work and has been carried out under the guidance of project guide **Prof. Shilpa Shanubhag**, Mahatma Gandhi Memorial College, Kunjibettu, Udupi

As per our knowledge, this work has not been submitted previously to the university by any other student.

There is no doubt that in spite of our strenuous efforts error might remain in the project. Naturally; we take full responsibility for any lack of clarity, occasional erratum or inexactness that might occur.

We have not submitted the matter embodied in this project for the award of any other degree.

Date:

Place: Udupi.

**RESHMA R BHAT**

**SHREETHU K S**

**SINCHANA C**

## **AKNOWLEDGEMENT**

A successful project is a fruitful culmination of the efforts of many people. Some directly involved and others who have quietly encouraged and extended their invaluable support throughout its progress.

We express gratitude to our beloved parents for their valuable support.

We would also like to convey our heartfelt thanks to our **Management** for providing us with good infrastructure, laboratory facility, qualified and inspiring staff whose guidance was of great help in successful completion of this project.

We would like to thank our beloved principal **Prof. Laxminarayana Karanth**, for his support throughout the project work.

We would like to express our heartfelt gratitude to our Head of the Department of Computer Science **Dr. M. Vishwanath Pai** and our project guide **Prof. Shilpa Shanubhag**, for guidance and encouragement throughout the project work.

We would like to thank our classmates for their encouragement.

Finally, we would like to thank all teaching and technical staff of Computer Science Department of MGM for their assistance during various stages of project work.

**Thanking You**

**RESHMA R BHAT**

**SHREETHU K S**

**SINCHANA C**



# **Index**

## **1. Introduction**

- a. Introduction of the System**
  - i. Project Title**
  - ii. Category**
  - iii. Overview**
- b. Background**
  - i. Brief note on Existing System**
- c. Objectives of the System**
- d. Scope of the System**
- e. End Users**
- f. Software/Hardware used for the development**
- g. Software/Hardware required for the implementation**

## **2. SRS**

- a. Introduction**
- b. Overall Description**
  - i. Product perspective**
  - ii. Product Functions**
  - iii. User Characteristics**
  - iv. General Constraints**
  - v. Assumptions**
- c. Functional Requirements**
  - i. Modules**
  - ii. Module Description**

- 3. System Design**
  - a. Introduction**
  - b. Description of Programs**
    - i. Context Flow Diagram**
    - ii. Data Flow Diagram**
- 4. Database Design**
  - a. Introduction**
  - b. Database Diagrams**
  - c. Data dictionary**
- 5. Program Code Listing**
- 6. User Interfaces**
- 7. Testing**
  - a. Introduction**
  - b. Test Reports**
    - i. Unit Testing**
    - ii. Integration Testing**
    - iii. System Testing**
  - c. Test Plan**
  - d. Test Cases**
- 8. Limitations**
- 9. Scope For Enhancements**
- 10. Abbreviation and Acronyms**
- 11. Bibliography/References**

# **1. Introduction**

## **a. Introduction to the System:**

PG Finder is an Android based platform that helps students and young professionals find suitable paying guest (PG) accommodations in a particular area that meets their needs and budget. Users can easily search for PGs by selecting their preferred city, locality and budget range. The platform also offers various filters such as gender-specific accommodations, food preferences, and availability of Wi-Fi and other amenities. Users can view photos of the PGs, read reviews from other users, and contact the owners directly through the platform.

### **i. Project Title:**

PG finder

### **ii. Category:**

The PG Finder Application is an android application with Android Studio as front end for user interface and Firebase as back end to store and retrieve data.

### **iii. Overview:**

The PG Finder is a convenient and reliable app designed to simplify the process of finding suitable paying guests for individuals in need. The app offers a user-friendly interface, making it easy for users to navigate and explore a wide range of available PG options, filtering them based on their preferences such as location, amenities, pricing, and more. The app provides detailed information about each listing, including images, descriptions, and contact details of the PG owner or manager. users can easily browse through a wide range of available PG options, filtering them based on their preferences such as location, amenities, pricing, and more. The app provides detailed information about each listing, including images, descriptions, and contact details of the PG owner or manager. it also offers a seamless booking process, allowing users to directly contact the PG owners or managers to inquire about availability, negotiate terms, and book their desired accommodations. the app provides a secure payment gateway for hassle-free transactions, ensuring a smooth and convenient booking experience.

**b. Background:**

**Brief note on Existing System:**

**c. Objectives of the System:**

- To provide an easy-to-use interface that enables users to search for PGs based on their preferences.
- Offering a variety of search filters to help users narrow down their search based on their specific needs such as location, budget, amenities etc.
- To facilitate easy communication between potential renters and PG owners.
- To enable PG owners to easily list their available paying guest accommodations on the platform and attract potential renters.
- Allow users to provide feedback and ratings about their experiences with the PG they have stayed in.

**d. Scope of the System:**

The project “PG-FINDER Android Application” can attract a wide range of users, including students, professionals, and people relocating for various reasons. The PG finder application is ideal solution for individual who are looking for affordable, comfortable and convenient PG accommodation in new city or town.

**e. End Users:**

End users are the users who have registered themselves with their Email, Username and Password. These users can search the PG according to their needs.

**f. Software/Hardware used for the development:**

**Software:**

**Front End: Android studio**

Android Studio is the official **Integrated Development Environment (IDE)** for **Google’s Android operating system**, built on **JetBrains IntelliJ IDEA** software and designed specifically for android development. Android Studio provides more features that enhance our productivity while building Android apps. It has a flexible gradle-based build system and it also has a fast and feature-rich emulator for app testing.



Android Virtual Device (Emulator) is used to run and debug apps in the Android Studio. Android Studio contains all the Android tools to design, test, debug, and profile the application. The Android Studio uses Gradle to manage the project, a Build Automation Tool. It provides a unified environment where we can build apps for Android phones, tablets, etc. The official language for Android Development is Java. The app module contains the following folders:

- **manifests:** It contains the AndroidManifest.xml file. Most of the code is generated automatically in this file, we need not change anything. The AndroidManifest.xml file contains information about the package, including components of the application such as activities, services, broadcast receivers, content providers, etc.
- **java:** It contains the source code of java files including the JUnit test code.
- **res:** It contains all non-code resources, UI strings, XML Layouts, and bitmap images.

### **Back End: Firebase**

**Firebase** is a Google backed application development software that enables the developers to develop android apps. The advantage of using online databases is that there are less chances of data being lost. The insertion and retrieval of data makes use of Firebase. Firebase is a Backend-as-a-Service, and it is a real time database which is basically designed for mobile applications.

### **Authentication:**

**Firebase Authentication** aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end solution supporting email and password accounts, phone auth and Google, Twitter login and more. Firebase Authentication provides backend services, easy-to-use SDKs, and readymade UI libraries to authenticate users to the app. Firebase Authentication uses Firebase Dynamic Links to send the email link to the mobile devices.

### **Realtime Database:**

**Firebase Realtime Database** is a cloud-hosted database in which data is stored as JSON. The data is synchronised in real-time to every connected client. It is a NoSQL database from which we can store and sync the data

between our users in real-time. It uses data synchronisation instead of using HTTP requests, due to this any connected device receives the updates within milliseconds. It is capable of providing all offline and online services. It ships with mobile SDKs, which allow us to build our app without the need for servers. Real-time syncing makes it easy for our users to access their data from any device, be it a web or mobile.

**Hardware:**

**Processor:** Intel core i5.

**RAM:** 8GB or more.

**Hard Disk:** 256Gb or more.

**g. Software/Hardware used for the implementation:**

**Software:**

Our Android project, PG Finder, uses common software tools like the Android operating system and Android Studio. Hence on the software side, we expect the user to use an Android Operating System based smartphone with minimum of Android Version 7.0 (Nougat).

**Hardware:**

To run our Android app smoothly, use any Android-compatible smartphone or tablet. Make sure it has enough processing power, memory, and storage space. It should also support Wi-Fi, Bluetooth, and mobile data for seamless connectivity.

## 2. SRS

### a. Introduction:

A Software Requirement Specification (SRS) is the complete description about what the software will do and how the software is expected to perform it. It also describes the functionality that the software needs to fulfil all the user needs. It is a formal report that enables the users to review whether SRS is according to their requirements.

### b. Overall Description:

#### i. **Product Perspective:**

PG Finder operates as an independent platform, acting as a bridge between individuals seeking PG accommodations and property owners offering such facilities. While there are existing online platforms that list PG accommodations, PG Finder differentiates itself by providing a user-friendly interface, advanced search filters.

#### ii. **Product Functions:**

- **Search:** Users can search for PG accommodations based on specific criteria such as location, budget, amenities, and occupancy preferences.
- **Post:** Users can post their PG by adding details about all the facilities.
- **Listing Details:** Users can view detailed information about each listed PG, including photographs, room availability, facilities provided, and contact information of the PG owner.
- **Wishlist:** Users can save their favorite PG listings for future reference.
- **Contact and Booking:** Users can directly contact PG owners through the platform, inquire about availability, schedule visits, and potentially make bookings

#### iii. **User Characteristics:**

The primary target audience for PG Finder is college students and young working professionals seeking affordable and convenient PG accommodations. These users are tech-savvy and rely on digital platforms for their daily needs. They are often looking for PGs in specific areas near their educational institutions or workplaces, with preferences for amenities like Wi-Fi, food services, security, and cleanliness.

#### **iv. General Constraints:**

- **Data Availability:** PG Finder relies on accurate and up-to-date information provided by PG owners. However, the availability of data may vary, and it may not always be possible to maintain real-time updates.
- **Geographic Limitations:** Initially, PG Finder will focus on specific cities or regions, expanding its coverage gradually over time.
- **Legal Compliance:** PG Finder will ensure the privacy and security of user data .

#### **v. Assumptions:**

- **Availability of PG Listings:** The application assumes that there will be a sufficient number of PG accommodations listed by PG owners or PG providers to offer users a wide range of choices.
- **User Engagement:** It assumes that users will actively engage with the application and it helps to enhance the overall user experience and help others in their search for suitable PG's.
- **Compliance with Regulations:** The application assumes that property owners or PG providers comply with relevant regulations and legal requirements for listing their accommodations.

#### **c. Functional requirements:**

##### **i. Modules:**

1. Registration
  - Login
  - Sign up
  - Forgot password
2. User
  - Booking PG
  - Posting Pg
  - PG Enquiry
  - wish list
3. Search
  - Filter
  - Sort
4. Booking and Payment
  - Contact Owner
  - Submit Document for Verification

- Payment Gateway

## 5. PG Details

- Photos
- Location
- Facilities
- Rent

### ii. Module Description:

- **Registration**

The user would require to create an account on the app by providing their email address and secured password. Once the user has completed the signup process, they can login to the app using email and password.

The app would have “forgot password” feature, that would allow the user to reset the password if they forget it.

Once the user has completed the above steps they would be registered on the app and can start looking for PG.

- **User**

After registering the user can either start searching for the PG that meets his/her requirement or can post a new PG on the app if he is looking to sell his own PG. The user can contact the PG owner to enquire about the property details before booking the PG.

The user can add PG of his liking to Wishlist so that he can view it later easily.

- **Search**

The app could provide the user with the ability to search for PG’s based on filter option such as location, rent range, gender, room type(home/apartment). It also provides an option to list the properties based on the newest or oldest posted Pg.

- **Booking and Payment**

The user can contact the PG owner to enquire details and visit the property. If satisfied with the property he can then proceed with the booking by submitting required details and confirm booking through payment option.

- **PG details**

The app would provide detailed features of PG by providing photos, location, rent details etc. The app would provide list of amenities that the PG offers such as Wi-Fi, 24/7 power backup, water supply etc. The app would display rent and advance payment that the user would need to pay.

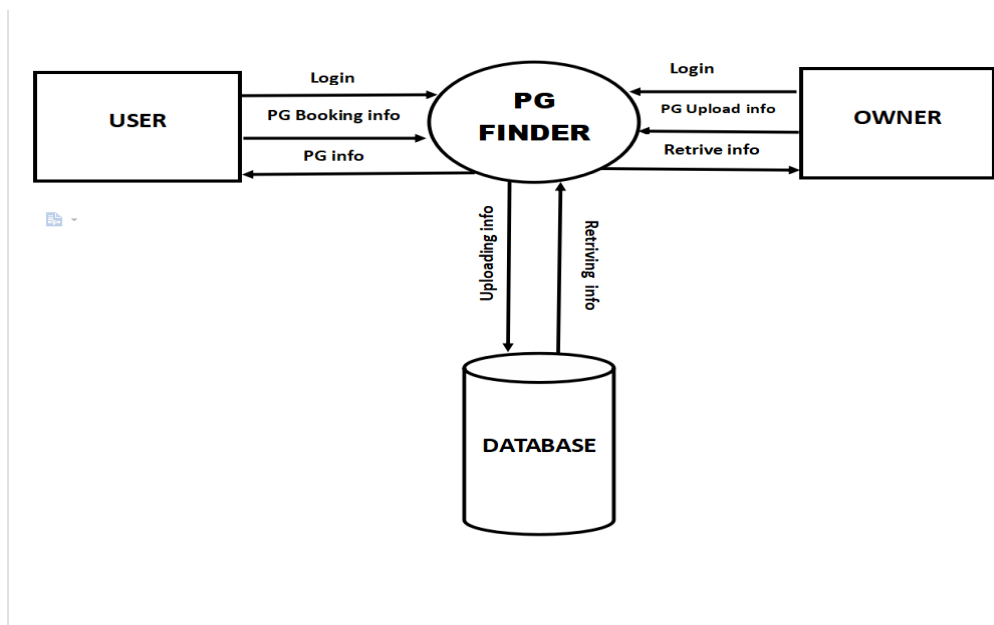
### **3. System Design**

#### **a. Introduction:**

System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organisation.

#### **b. Description of Programs:**

- i. **Context Flow Diagram:** CFD is one in which the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified. A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of the information between the system and external entities. The entire software system is shown as a single process. It is used to establish the context and boundaries of the system to be modelled and identify the relationship of the system with the external entities.




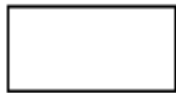

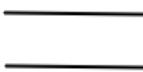
## ii. Data Flow Diagram:

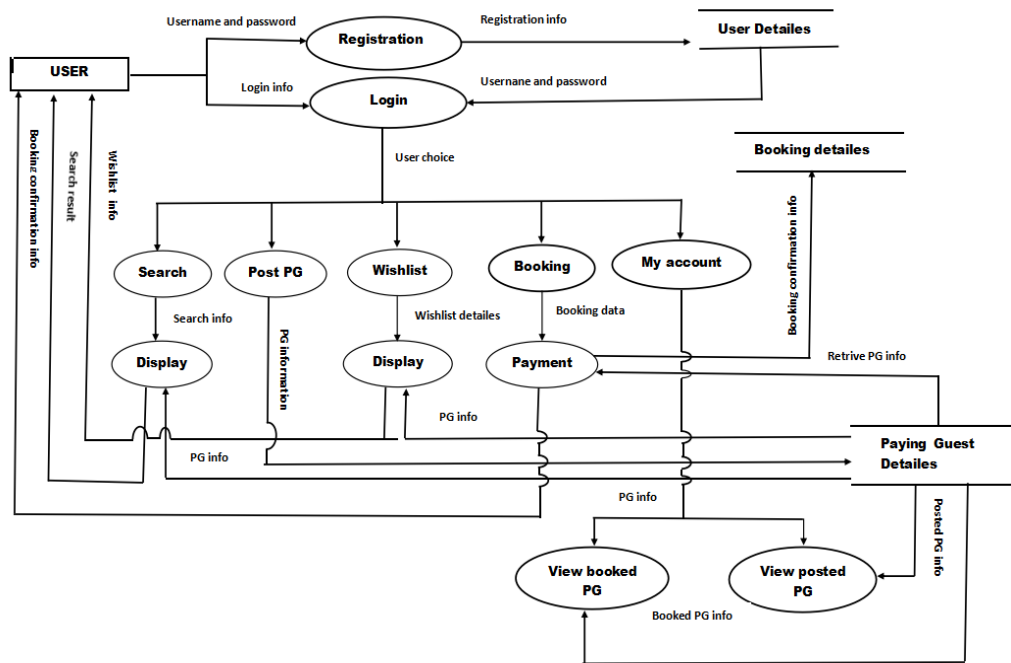
A data-flow diagram is a way of representing a flow through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data flow diagram has no control flow; there are no decision rules and no loops. Levels in DFD are numbered 0, 1, 2 and beyond.

The DFD server two purposes:

- To provide an indication of flow of data are transformed as they move through the system.
- To depict the function that transforms the data flow.

### Notations for DFD Diagram:

NOTATION	DISCRIPTION
	A circle represents a process applied to data or control or changes if in same way.
	A rectangle is used to represent an external entity that is a system element or another system that produce information by the Software or receiver information produced <u>By</u> the software.
	An arrow represents one or more data items or data objects.
	The store includes the database of the system.it is represented by two parallel lines.





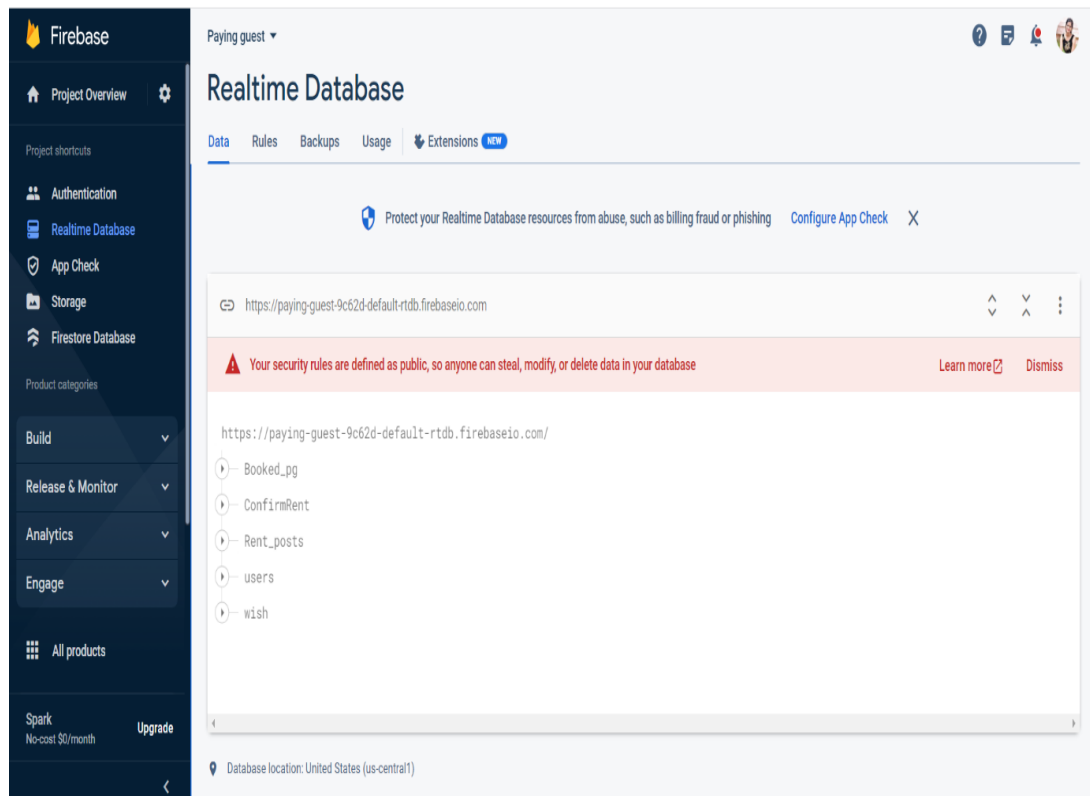
## 4. Database Design

### a. Introduction:

The data is stored as JSON objects, which is not a relational database and it doesn't store data in the form of tables, rows and columns. The database is in the form of a cloud hosted json tree. The node "users" containing ID's stores the data in form of key value pairs.

### b. Database Diagrams:

#### Realtime Database



## Authentication:

The screenshot shows the Firebase Authentication console. The left sidebar contains the Firebase logo and navigation links: Project Overview, Authentication (selected), Realtime Database, App Check, Storage, and Firestore Database. Below these are Product categories (Build, Release & Monitor, Analytics, Engage) and All products. The main content area is titled 'Authentication' and includes tabs for Users, Sign-in method, Templates, Usage, Settings, and Extensions. A search bar at the top allows searching by email address, phone number, or user UID. Below the search bar is a table of users.

Identifier	Providers	Created	Signed In	User UID
sinchana0613@gmail.com	Google	Jul 3, 2023	Jul 3, 2023	Msh7mQazFWRYmvaPBmMKT9q4r...
spandanagowda618@gmail...	Google	Jul 2, 2023	Jul 2, 2023	pQh5J96wLbXH56ybYqFdASyLj2
rohanbhat72@gmail.com	Google	Jun 29, 2023	Jul 3, 2023	tELDy2YWzqZbf8m4NdfbuDITN293
harshithnayak15@gmail.co...	Google	Jun 26, 2023	Jun 26, 2023	8xIsqEb0E4gcCYkmKlr434X2F12
shreyakundar84@gmail.com	Google	Jun 26, 2023	Jun 28, 2023	oowxreBtYvdZVmlGekWKV4eUkk...
rash1@gmail.com	Google	Jun 26, 2023	Jul 4, 2023	9JH0m4sDalQ1wUTAcDzW3E727...
shreethu@gmail.com	Google	Jun 25, 2023	Jul 4, 2023	cq98JdAJCqPyweeb1ysuN2DqtcC2
sinchana0613@gmail.com	Google	Jun 25, 2023	Jun 25, 2023	swgsfVNALbYSMu2MIRh33eMhB2
rohan.bhat72@gmail.com	Google	Jun 25, 2023	Jun 25, 2023	0vaR6z8D3SQE9Pwy86A4goKzY6...
sreethu.ks.santekatte@gm...	Google	Jun 24, 2023	Jun 24, 2023	MXoDpP17zbMm9AwnJiMu90S...

## Storage:

The screenshot shows the Firebase Storage console. The left sidebar is identical to the Authentication console. The main content area is titled 'Storage' and shows a list of files. The file list has columns for Name, Size, Type, and Last modified. A file named 'tELDy2YWzqZbf8m...' is selected, and its details are shown on the right.

Name	Size	Type	Last modified
8xIsqEb0E4gcCYkmKlr434X2F12.jpg	652.9 KB	image/jpeg	Jun 26, 2023
9JH0m4sDalQ1wUTAcDzW3E72743.jpg	460.43 KB	image/jpeg	Jun 26, 2023
LtnOVKvSPMNPanWjQ3hywTKB12.jpg	80.91 KB	image/jpeg	Jun 23, 2023
MXoDpP17zbMm9AwnJiMu90S4282.jpg	80.24 KB	image/jpeg	Jun 24, 2023
NhSaTtUol.qdJUV0kC0k2UdXa0R3.jpg	363.43 KB	image/jpeg	Jun 23, 2023
cq98JdAJCqPyweeb1ysuN2DqtcC2.jpg	317.89 KB	image/jpeg	Jun 29, 2023
oowxreBtYvdZVmlGekWKV4eUkk3.heic	1.46 MB	image/heic	Jun 27, 2023
phLz5TGBnzQZdV8qadFB1zE9uA2.jpg	3.43 MB	image/jpeg	Jun 23, 2023
swgsfVNALbYSMu2MIRh33eMhB2.jpg	625.63 KB	image/jpeg	Jun 25, 2023
tELDy2YWzqZbf8m4NdfbuDITN293.png	49.37 KB	image/png	Jun 30, 2023

File details for 'tELDy2YWzqZbf8m...':

- Name: tELDy2YWzqZbf8m4NdfbuDITN293.png
- Size: 50,552 bytes
- Type: image/png
- Created: Jun 30, 2023, 2:27:34PM
- Updated: Jun 30, 2023, 2:27:34PM
- File location: gs://paying-guest-9c62d.appspot.com/DisplayPics/tELDy2YWzqZbf8m4NdfbuDITN293.png
- Other metadata: Expandable section

## c. Data Dictionary

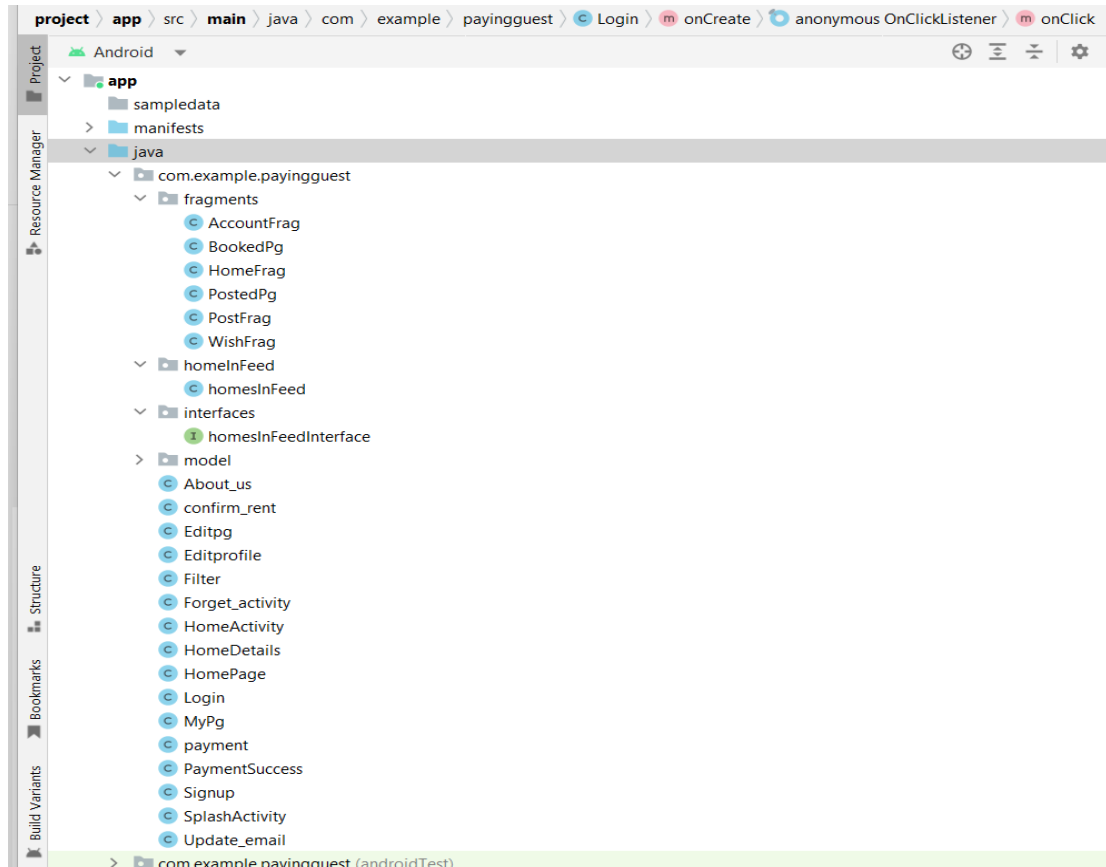
**1.Login:**  $^(?=.*[0-9])(?=. *[a-z])(?=. *[A-Z]) (?=.*[ @# $ % ^ & * & ++ ])(?=\$+)$ . \{6, \} \$$

**2.Forgot Password:**  $^(?=.*[0-9])(?=. *[a-z])(?=. *[A-Z]) (?=.*[ @# $ % ^ & * & ++ ])(?=\$+)$ . \{6, \} \$$

**3.Update Password:**  $^(?=.*[0-9])(?=. *[a-z])(?=. *[A-Z]) (?=.*[ @# $ % ^ & * & ++ ])(?=\$+)$ . \{6, \} \$$

## 5. Program Code Listing

### Project Structure:



### Source code

#### LOGIN:

#### Java file

```
package com.example.payingguest;
```

```
import androidx.annotation.NonNull;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.appcompat.widget.AppCompatButton;

import android.annotation.SuppressLint;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Patterns;

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.OnFailureListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

public class Login extends AppCompatActivity {

    EditText usermail, userpass;

    TextView txt, forgot;
```

```

AppCompatActivity btn;

private FirebaseAuth mAuth;

ProgressDialog progress;

@SuppressLint("MissingInflatedId")

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_login);

    usermail = findViewById(R.id.mail);

    userpass = findViewById(R.id.pass);

    btn = findViewById(R.id.loginbtn);

    mAuth=FirebaseAuth.getInstance();

    txt = findViewById(R.id.no_acc);

    forgot=findViewById(R.id.forgottext);

    progress=new ProgressDialog(this);

    txt.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) {

            Intent i = new Intent(Login.this, Signup.class);

```

```

        startActivity(i);

        finish();} });

forgot.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

    startActivity(new Intent(Login.this,Forget_activity.class))});

btn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

    if(!usermail.getText().toString().trim().isEmpty()
&&emailChecker(usermail.getText().toString().trim())) {

        if(!usermail.getText().toString().isEmpty()) {

            progress.setTitle("Login");

            progress.setMessage("Logging in");

            progress.setCanceledOnTouchOutside(false);

            progress.show();
loginUser(usermail.getText().toString().trim(),userpass.getText().toString().trim());}

            else{

                Toast.makeText(Login.this,"Entervalid
password",Toast.LENGTH_SHORT).show();}

```

```

        }else{

            Toast.makeText(Login.this,"Entervolid
email",Toast.LENGTH_SHORT).show();

        } } } });

    boolean emailChecker(String email) {

        return Patterns.EMAIL_ADDRESS.matcher(email).matches();

    }

    void loginUser(String email,String password){

        mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new
        OnCompleteListener<AuthResult>() {

            @Override

            public void onComplete(@NonNull Task<AuthResult> task) {

                if(task.isSuccessful()){

                    progress.dismiss();

                    Toast.makeText(Login.this,"Login
successful",Toast.LENGTH_SHORT).show();

                    startActivity(new Intent(Login.this,HomeActivity.class));

                    finish();}

                else{

                    progress.dismiss();
                    Toast.makeText(Login.this,"Failed",Toast.LENGTH_SHORT).show();}
            }
        }
    }

```

```

    }).addOnFailureListener(new OnFailureListener() {

        @Override

        public void onFailure(@NonNull Exception e) {

            progress.dismiss();

            Toast.makeText(Login.this, "Fail..", Toast.LENGTH_SHORT).show();

        }
    });
}
}
}
}

```

### **Design Code**

```

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout 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="#c5e2ea"

    tools:context=".Login">

    <TextView

        android:id="@+id/textView3"

```



```
android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginStart="20dp"

android:layout_marginTop="10dp"

android:layout_marginEnd="20dp"

android:fontFamily="@font/fl"

android:text="Welcome"

android:textColor="#3b9193"

android:textSize="34sp"

android:textStyle="italic" />
```

```
<ImageView
```

```
    android:id="@+id/imageView4"

    android:layout_width="401dp"

    android:layout_height="272dp"

    android:layout_alignParentTop="true"

    android:layout_alignParentBottom="true"

    android:layout_marginStart="10dp"

    android:layout_marginTop="17dp"

    android:layout_marginEnd="10dp"
```

```
android:layout_marginBottom="442dp"

android:contentDescription="@string/todo2"

app:srcCompat="@drawable/log" />
```

```
<LinearLayout
```

```
    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:layout_marginLeft="20dp"

    android:layout_marginTop="250dp"

    android:layout_marginRight="20dp"

    android:layout_marginBottom="5dp"

    android:background="@drawable/layout_bg"

    android:orientation="vertical"

    android:visibility="visible">
```

```
<TextView
```

```
    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_marginStart="20dp"

    android:layout_marginTop="20dp"

    android:layout_marginEnd="40dp"
```

```
android:fontFamily="@font/arbutus"
```

```
android:text="Login"
```

```
android:textColor="#3aa8c1"
```

```
android:textSize="25sp" />
```

```
<EditText
```

```
android:id="@+id/mail"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
```

```
android:layout_margin="20dp"
```

```
android:layout_marginStart="100dp"
```

```
android:layout_marginTop="70dp"
```

```
android:background="@drawable/text_bg"
```

```
android:drawableLeft="@drawable/baseline_email"
```

```
android:drawableTint="#1191A1"
```

```
android:hint=" Email"
```

```
android:padding="15dp"
```

```
android:textColor="#00babe"
```

```
android:textColorHint="#00babe"
```

```
android:textSize="20sp" />
```

<EditText

```
    android:id="@+id/pass"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_margin="20dp"

    android:layout_marginStart="100dp"

    android:layout_marginTop="70dp"

    android:background="@drawable/text_bg"

    android:drawableLeft="@drawable/baseline_security"

    android:drawableTint="#1191A1"

    android:hint=" Password"

    android:inputType="textPassword"

    android:password="true"

    android:padding="15dp"

    android:textColor="#00babe"

    android:textColorHint="#00babe"

    android:textSize="20sp" />
```

<TextView

```
    android:id="@+id/forgottext"
```

```
android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginStart="200dp"

android:layout_marginEnd="10dp"

android:text="Forgot password?"

android:textColor="@color/black"

android:textSize="20sp"

android:textStyle="bold|italic" />
```

```
<androidx.appcompat.widget.AppCompatButton
```

```
    android:id="@+id/loginbtn"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_marginStart="20dp"

    android:layout_marginTop="15dp"

    android:layout_marginEnd="20dp"

    android:background="@drawable/btn_bg"

    android:padding="15dp"

    android:text="Login"

    android:textSize="24sp"
```

```

        android:textStyle="italic" />

<TextView

    android:id="@+id/no_acc"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_margin="15dp"

    android:text="@string/don_t_have_an_account_sign_up"

    android:textAlignment="center"

    android:textColor="@color/black"

    android:textSize="18sp"

    android:textStyle="bold" />

</LinearLayout>

</RelativeLayout>

```

### **Sign up:**

#### **Java File**

```

package com.example.payingguest;

import android.app.ProgressDialog;

import android.content.Intent;

```

```
import android.os.Bundle;

import android.util.Patterns;

import android.view.View;

import android.widget.EditText;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.AppCompatButton;

import com.example.payingguest.model.User;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import java.util.regex.Matcher;
```

```

import java.util.regex.Pattern;

public class Signup extends AppCompatActivity {

    private EditText name, mail, pass, cpass;

    private TextView msg;

    private AppCompatActivity b;

    private FirebaseAuth mAuth;

    private DatabaseReference mRef;

    private String uname,upass,ucpass,umail;

    StringpasswordPattern="^(?=.*[0-9])(?=.*[a-
        z])(?=.*[AZ])(?=.*[@#$%^&+=])(?=\S+$).{6,}$";

    ProgressDialog dialog;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_signup);

        name = findViewById(R.id.name);

        mail = findViewById(R.id.mail);

        pass = findViewById(R.id.pass);

        cpass = findViewById(R.id.repass);

```



```

b = findViewById(R.id.signbtn);

msg = findViewById(R.id.have_acc);

mAuth = FirebaseAuth.getInstance();

mRef= FirebaseDatabase.getInstance().getReference();

dialog=new ProgressDialog(this);

msg.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        Intent i = new Intent(Signup.this, Login.class);

        startActivity(i);

        finish();

    } });

b.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        uname=name.getText().toString();

        umail=mail.getText().toString();

        upass=pass.getText().toString();

        ucpass=cpass.getText().toString();

```

```
if (uname.isEmpty()) {

    name.setError("Enter name");

} else if (umail.isEmpty()) {

    mail.setError("Enter valid email");

} else if (!upass.matches(passwordPattern)) {

    pass.setError("Password should contain atleast 6 characters with 1 digit,1 lowercase letter,1 uppercase letter");

} else if (!upass.equals(ucpass)) {

    cpass.setError("Password does not match");

} else {

    if (emailChecker(umail)) {

        dialog.setTitle("Sign Up");

        dialog.setMessage("Signing up");

        dialog.setCanceledOnTouchOutside(false);

        dialog.show();

        createUser(umail,upass,uname);

    }else{

        mail.setError("Enter valid email");

    }

}}}
```

```

boolean emailChecker(String email) {

    return Patterns.EMAIL_ADDRESS.matcher(email).matches();}

void createUser(String email, String pword,String name) {

 mAuth.createUserWithEmailAndPassword(email,pword).addOnCompleteListener(new
    OnCompleteListener<AuthResult>() {

        @Override

        public void onComplete(@NonNull Task<AuthResult> task) {

            User user=new User(name,email);

            if(task.isSuccessful()){

                mRef.child("users").push().setValue(user);

                dialog.dismiss();

                startActivity(new Intent(Signup.this,HomeActivity.class));

                finish();

                Toast.makeText(Signup.this,"Usercreated",Toast.LENGTH_SHORT).show();

            }else{

                dialog.dismiss();
Toast.makeText(Signup.this,""+task.getException(),Toast.LENGTH_SHORT).show();

            }}

        }).addOnFailureListener(new OnFailureListener() {

            @Override

```

```

        public void onFailure(@NonNull Exception e) {

            dialog.dismiss();

            Toast.makeText(Signup.this,"Fail",Toast.LENGTH_SHORT).show();

        });}}

```

### **Design Code**

```

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout 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="#c5e2ea"

    tools:context=".Signup">

    <ImageView

        android:id="@+id/imageView3"

        android:layout_width="420dp"

        android:layout_height="284dp"

        android:layout_alignParentStart="true"

        android:layout_alignParentEnd="true"

```

```

        android:layout_alignParentBottom="true"

        android:layout_marginStart="-9dp"

        android:layout_marginTop="1dp"

        android:layout_marginEnd="0dp"

        android:layout_marginBottom="445dp"

        app:srcCompat="@drawable/sign" />
<LinearLayout

        android:layout_width="match_parent"

        android:layout_height="match_parent"

        android:layout_marginLeft="20dp"

        android:layout_marginTop="255dp"

        android:layout_marginRight="20dp"

        android:layout_marginBottom="5dp"

        android:background="@drawable/layout_bg"

        android:orientation="vertical">

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_marginStart="20dp"

```

```
android:layout_marginTop="20dp"

android:layout_marginEnd="40dp"

android:fontFamily="@font/arbutus"

android:text="SignUp"

android:textColor="#3aa8c1"

android:textSize="25sp" />
```

```
<EditText
```

```
    android:id="@+id/name"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_marginStart="20dp"

    android:layout_marginTop="20dp"

    android:layout_marginEnd="20dp"

    android:background="@drawable/text_bg"

    android:drawableLeft="@drawable/baseline_person"

    android:drawableTint="#1191A1"

    android:hint="  Name"

    android:padding="15dp"

    android:textColor="#00babe"
```

```

        android:textColorHint="#00babe"

        android:textSize="20sp" />
<EditText

        android:id="@+id/mail"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_marginStart="20dp"

        android:layout_marginTop="12dp"

        android:layout_marginEnd="20dp"

        android:layout_marginBottom="1dp"

        android:background="@drawable/text_bg"

        android:drawableLeft="@drawable/baseline_email"

        android:drawableTint="#1191A1"

        android:hint=" Email"

        android:padding="15dp"

        android:textColor="#00babe"

        android:textColorHint="#00babe"

        android:textSize="20sp" />
<EditText

```

```
android:id="@+id/pass"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:layout_marginStart="20dp"

android:layout_marginTop="12dp"

android:layout_marginEnd="20dp"

android:background="@drawable/text_bg"

android:drawableLeft="@drawable/baseline_security"

android:drawableTint="#1191A1"

android:hint=" Password"

android:inputType="textPassword"

android:padding="15dp"

android:password="true"

android:textColor="#00babe"

android:textColorHint="#00babe"

android:textSize="20sp"

android:visibility="visible" />
```

```
<EditText
```

```
android:id="@+id/repass"
```



```

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_marginStart="20dp"

        android:layout_marginTop="12dp"

        android:layout_marginEnd="20dp"

        android:background="@drawable/text_bg"

        android:drawableLeft="@drawable/baseline_security"

        android:drawableTint="#1191A1"

        android:hint="Confirm Password"

        android:password="true"

        android:inputType="textPassword"

        android:padding="15dp"

        android:textColor="#00babe"

        android:textColorHint="#00babe"

        android:textSize="20sp" />
<androidx.appcompat.widget.AppCompatButton

        android:id="@+id/signbtn"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

```

```
android:layout_marginStart="20dp"

android:layout_marginTop="10dp"

android:layout_marginEnd="20dp"

android:background="@drawable/btn_bg"

android:padding="10dp"

android:text="SignUp"

android:textSize="24sp"

android:textStyle="italic" />
```

```
<TextView
```

```
    android:id="@+id/have_acc"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:layout_marginStart="20dp"

    android:layout_marginTop="5dp"

    android:layout_marginEnd="20dp"

    android:text="@string/already_have_an_account_login"

    android:textAlignment="center"

    android:textColor="@color/black"

    android:textSize="18sp"
```

```
        android:textStyle="bold" />

    </LinearLayout>

</RelativeLayout>
```

## **FORGOT PASSWORD**

### **Java file**

```
package com.example.payingguest;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.AppCompatButton;


import android.annotation.SuppressLint;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ProgressBar;
```

```

import android.widget.Toast;

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

public class Forget_activity extends AppCompatActivity {

    private EditText em;

    private AppCompatActivity bt;

    private FirebaseAuth auth;

    ProgressDialog dg;

    private String fmail;

    @SuppressWarnings("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_forget);

```

```

em=findViewById(R.id.mail);

bt=findViewById(R.id.button);

dg=new ProgressDialog(this);

auth=FirebaseAuth.getInstance();

bt.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        fmail=em.getText().toString();

        if(fmail.isEmpty())

            { em.setError("Enter email");}

        else{

            dg.setMessage("Loading");

            dg.setCanceledOnTouchOutside(false);

            dg.show();

            forgetpass(); } } });

private void forgetpass() {

    auth.sendPasswordResetEmail(fmail).addOnCompleteListener(new

        OnCompleteListener<Void>() {

            @Override

```

```

public void onComplete(@NonNull Task<Void> task) {

    if(task.isSuccessful()){

        dg.dismiss();

        Toast.makeText(Forget_activity.this,"mailhasbeensent
successfully",Toast.LENGTH_SHORT).show();

        startActivity( new Intent(Forget_activity.this,Login.class));

        finish();

    }else{dg.dismiss();
    Toast.makeText(Forget_activity.this,""+task.getException().getMessage(),Toast.
    LENGTH_SHORT).show();}}

}).addOnFailureListener(new OnFailureListener() {

@Override public void onFailure(@NonNull Exception e) {

    Toast.makeText(Forget_activity.this,"Failed",Toast.LENGTH_SHORT).show();

    }});}}

```

### **Design Code**

```

<?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/fpp"

tools:context=".Forgot_activity">

<TextView

    android:id="@+id/textView4"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:fontFamily="@font/fl"

    android:padding="20dp"

    android:text="Forgot Password"

    android:textAlignment="center"

    android:textColor="@color/MainBlue"

    android:textSize="40sp"

    android:textStyle="italic"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHorizontal_bias="0.0"

    app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent"

app:layout_constraintVertical_bias="0.128" />

<EditText

    android:id="@+id/mail"

    android:layout_width="360dp"

    android:layout_height="69dp"

    android:background="@drawable/text_bg"

    android:drawableLeft="@drawable/baseline_email"

    android:drawableTint="#24B3B3"

    android:ems="10"

    android:hint="    Email"

    android:inputType="textPersonName"

    android:padding="20dp"

    android:textColor="@color/MainBlue"

    android:textColorHint="@color/MainBlue"

    android:textSize="20dp"

    android:textStyle="italic"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintEnd_toEndOf="parent"
```



```
app:layout_constraintHorizontal_bias="0.686"
```

```
app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent"
```

```
app:layout_constraintVertical_bias="0.439" />
```

```
<androidx.appcompat.widget.AppCompatButton
```

```
android:id="@+id/button"
```

```
android:layout_width="290dp"
```

```
android:layout_height="61dp"
```

```
android:background="@drawable/btn_bg"
```

```
android:text="Reset"
```

```
android:textSize="24dp"
```

```
android:textStyle="bold|italic"
```

```
app:layout_constraintBottom_toBottomOf="parent"
```

```
app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintHorizontal_bias="0.495"
```

```
app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toTopOf="parent"
```

```
app:layout_constraintVertical_bias="0.61" />
```

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

### **Post fragment**

```
package com.example.payingguest.fragments;

import static android.app.Activity.RESULT_OK;

import android.annotation.SuppressLint;

import android.app.ProgressDialog;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.widget.AppCompatButton;

import androidx.fragment.app.Fragment;

import android.provider.MediaStore;

import android.util.Patterns;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;
```

```
import android.widget.ArrayAdapter;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.Spinner;

import android.widget.Toast;

import com.example.payingguest.HomeActivity;

import com.example.payingguest.Login;

import com.example.payingguest.R;

import com.google.android.gms.tasks.Continuation;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;
```

```

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.UploadTask;

import com.google.type.DateTime;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.HashMap;

public class PostFrag extends Fragment {

    RadioGroup rgroomtype,rgsharetype,rgfood,rgguest,rggender;

    RadioButton radioroomtype,radioshare,radiofood,radioguest,radiogender;

    CheckBox
    breakf,luch,dinr,none,bed,pillw,wf,cupbrd,frdg,laund,park,table,keeping,water,tv,stove;

    EditText rent,depo,city,loc,description,contact;

    String mobilepattern="[0-9]{10}";

    String[] cities={"Udupi","Bhramavar","Karkala","Kundapur"};

    String spincity;

    AppCompatButton post;

```

```
ImageView homeImg;
```

```
Spinner spin;
```

```
    private String downloadUri;
```

```
Stringroom="",share="",food="",guest="",gender="",mrent="",sdep="",edcity="",edloc=
"",eddesc="",edcontact="";
```

```
String meals,amenities;
```

```
    String saveCurrentDate, saveCurrentTime;
```

```
    private String randomKey;
```

```
    private ProgressDialog pd;
```

```
    private DatabaseReference postDataRef;
```

```
    private FirebaseAuth auth;
```

```
    FirebaseAuth mAuth;
```

```
    FirebaseUser user;
```

```
    private DatabaseReference databaseReference;
```

```
    private static final int galleryPic = 1;
```

```
    private Uri ImageUri ;
```

```
    private StorageReference homepic;
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
```

```

        super.onCreate(savedInstanceState);}

@SuppressLint("MissingInflatedId")

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

                           Bundle savedInstanceState) {

    View view=inflater.inflate(R.layout.fragment_post, container, false);

    mAuth=FirebaseAuth.getInstance();

    auth= FirebaseAuth.getInstance();

    user=FirebaseAuth.getInstance().getCurrentUser();

    databaseReference=FirebaseDatabase.getInstance().getReference().child("users");

    homepic= FirebaseStorage.getInstance().getReference().child("home_pictures");

    postDataRef = FirebaseDatabase.getInstance().getReference().child("Rent_posts");

    rgroomtype=(RadioGroup)view.findViewById(R.id.rgroom);

    rgsharetype=(RadioGroup)view.findViewById(R.id.rgshare);

    rgfood=(RadioGroup)view.findViewById(R.id.rgfood);

    rgguest=(RadioGroup)view.findViewById(R.id.rgguest);

    rggender=(RadioGroup) view.findViewById(R.id.rggender);

    spin=view.findViewById(R.id.spinnercity);

    breakf=view.findViewById(R.id.breakfst);

```

```
lunch=view.findViewById(R.id.lunch);

dinner=view.findViewById(R.id.dinner);

none=view.findViewById(R.id.None);

bed=view.findViewById(R.id.bed);

pillw=view.findViewById(R.id.Pillow);

wf=view.findViewById(R.id.Wifi);

cupbrd=view.findViewById(R.id.Cupbrd);

frdg=view.findViewById(R.id.Fridge);

laund=view.findViewById(R.id.Laundry);

park=view.findViewById(R.id.Parking);

table=view.findViewById(R.id.table);

keeping=view.findViewById(R.id.keeping);

water=view.findViewById(R.id.water);

tv=view.findViewById(R.id.tv);

stove=view.findViewById(R.id.Stove);

loc=view.findViewById(R.id.etloc);

post=view.findViewById(R.id.post);

homeImg=view.findViewById(R.id.homeImage);

rent=view.findViewById(R.id.mrent);
```

```

depo=view.findViewById(R.id.sdepo);

description=view.findViewById(R.id.etdesc);

contact=view.findViewById(R.id.etcontact);

pd = new ProgressDialog(getContext());

ArrayAdapteradapter=new ArrayAdapter<>(getActivity(),

R.layout.spinnerdisplay,R.id.spinnerDisplay,cities);

spin.setAdapter(adapter);

homeImg.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {openGallery(); } });

    post.setOnClickListener(new View.OnClickListener() {

        @Override

        public void onClick(View v) { collectData();});

    return view;}

    private void openGallery() {

        Intentintent=newIntent(Intent.ACTION_PICK,MediaStore.Images.Media.EXTE
            RNAL_CONTENT_URI);

        startActivityForResult(intent, galleryPic);}

        @Override

```



```

    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {

        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == galleryPic && resultCode == RESULT_OK && data != null
&&    data.getData() != null) {

            ImageUri = data.getData();

            homeImg.setImageURI(ImageUri);}}

        public String getCurrentUser() {

            user=auth.getCurrentUser();

            if(user!=null){

return user.getEmail();

            }else{return null;}}

        public void collectData(){

            int genderselect=rggender.getCheckedRadioButtonId();

            radiogender= rggender.findViewById(genderselect);

            int roomselected=rgroomtype.getCheckedRadioButtonId();

            radiatoroomtype=rgroomtype.findViewById(roomselected);

            int shareselect=rgsharetype.getCheckedRadioButtonId();

            radioshare=rgsharetype.findViewById(shareselect);

            int foodselect=rgfood.getCheckedRadioButtonId();

```

```

radiofood=rgfood.findViewById(foodselect);

int guestselect=rgguest.getCheckedRadioButtonId();

spincity=spin.getSelectedItem().toString();

radioguest=rgguest.findViewById(guestselect);

room= (String) radiatorroomtype.getText().toString();

share= (String) radioshare.getText().toString();

food= (String) radiofood.getText().toString();

guest= (String) radioguest.getText().toString();

gender= (String) radiogender.getText().toString();

mrent= String.valueOf(rent.getText());

sdep= String.valueOf(depo.getText());

edloc= String.valueOf(loc.getText());

eddsc= String.valueOf(description.getText());

edcontact= String.valueOf(contact.getText());

StringBuilder mealselected=new StringBuilder();

StringBuilder facilityselected=new StringBuilder();

if(breakf.isChecked()){

mealselected.append(breakf.getText()).append(",");}

if(lrch.isChecked()){

```

```

mealselected.append(lnch.getText()).append(",");}

if(dinr.isChecked()){

mealselected.append(dinr.getText()).append(",");}

if(none.isChecked()){ mealselected.append(none.getText()).append(",");}

else { mealselected.append("");}

    meals=mealselected.toString();

    if(bed.isChecked()){

facilityselected.append(bed.getText()).append(",");

    }if(pillw.isChecked()){

facilityselected.append(pillw.getText()).append(",");

}if(wf.isChecked()){

    facilityselected.append(wf.getText()).append(",");

}if(laund.isChecked()

{ facilityselected.append(laund.getText()).append(",");

    }if(cupbrd.isChecked()

{facilityselected.append(cupbrd.getText()).append(",");

    }if(park.isChecked()

{ facilityselected.append(park.getText()).append(",");

    }if(keeping.isChecked()

```

```

        { facilityselected.append(keeping.getText()).append(",");

            }if(tv.isChecked())

        { facilityselected.append(tv.getText()).append(",");

            }if(table.isChecked())

        { facilityselected.append(table.getText()).append(",");

        }if(frdg.isChecked())

        { facilityselected.append(frdg.getText()).append(",");

        }if(water.isChecked())

        { facilityselected.append(water.getText()).append(",");

        }if(stove.isChecked())

        {    facilityselected.append(stove.getText()).append(",");

            }else { facilityselected.append("");}

amenities=facilityselected.toString();

if(ImageUri==null){

Toast.makeText(getActivity(),"Pleaseselectimage",Toast.LENGTH_SHORT).show();}

else if (mrent.isEmpty()) { rent.setError("Enter rent");}

else if (sdep.isEmpty()) { depo.setError("Enter deposit");}

else if (edloc.isEmpty()) { loc.setError("enter the locality");}

```

```

        elseif(!edcontact.matches(mobilepattern)){contact.setError("Entervailidphonenu
        mber");

    } else{ storeImageData();}}

    private void storeImageData() {

pd.setMessage("Posting");

pd.show();

Calendar calendar= Calendar.getInstance();

SimpleDateFormat currentDate= new SimpleDateFormat("MM dd, yyyy");

saveCurrentDate= currentDate.format(calendar.getTime());

SimpleDateFormat currentTime= new SimpleDateFormat("HH: mm: ss a");

saveCurrentTime= currentTime.format(calendar.getTime());

randomKey= saveCurrentDate+ saveCurrentTime;

if(randomKey==null){

    saveCurrentDate= currentDate.format(calendar.getTime());

    saveCurrentTime= currentTime.format(calendar.getTime());

    randomKey= saveCurrentDate+ saveCurrentTime;

}    finalStorageReferencefile=homepic.child(ImageUri.getLastPathSegment()+
    randomKey + ".jpg");

    final UploadTask uploadTask= file.putFile(ImageUri);

uploadTask.addOnFailureListener(new OnFailureListener() {

```

@Override

```
public void onFailure(@NonNull Exception e){
```

```
    String message = e.toString();
```

```
    Toast.makeText(getActivity(), "Failed to upload image: " + message,  
    Toast.LENGTH_SHORT).show();}
```

```
}).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
```

@Override

```
public void onSuccess(UploadTask.TaskSnapshot taskSnapshot){
```

```
    Toast.makeText(getActivity(), "Image uploaded Successfully",  
    Toast.LENGTH_SHORT).show();
```

```
    Task<Uri> urlTask = uploadTask.continueWithTask(new  
    Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
```

@Override

```
    PublicTask<Uri>then(@NonNull Task<UploadTask.TaskSnapshot> task)  
    throws Exception
```

```
{ if (!task.isSuccessful()){ throw task.getException();}
```

```
    downloadUri = file.getDownloadUrl().toString();
```

```
    return file.getDownloadUrl();}
```

```
}).addOnCompleteListener(new OnCompleteListener<Uri>() {
```

@Override

```
public void onComplete(@NonNull Task<Uri> task) {
```

```

        if(task.isSuccessful()){

            downloadUri=task.getResult().toString();
            Toast.makeText(getActivity(),"Done",Toast.LENGTH_SHORT).show();

            UpdateDatabase();

        } } ); } } ); }

private void UpdateDatabase() {

//databaseReference = FirebaseDatabase.getInstance().getReference("Rent_posts");

String userid=getCurrentUser();

HashMap<String, Object> map= new HashMap<>();

map.put("pId",randomKey);

map.put("date",saveCurrentDate);

map.put("time",saveCurrentTime);

map.put("image",downloadUri);

map.put("roomType",room);

map.put("Meals",meals);

map.put("Amenities",amenities);

map.put("gender",gender);

map.put("sharetype",share);

map.put("monthlyRent",mrent);

```

```

map.put("securityDeposit",sdep);

map.put("city",spincity);

map.put("locality",edloc);

map.put("description",eddesc);

map.put("contactNo",edcontact);

map.put("foodPreference",food);

map.put("guestsAllowed",guest);

map.put("user",userid);

map.put("Publisher", FirebaseAuth.getInstance().getCurrentUser().getUid());

postDataRef.push().setValue(map).addOnCompleteListener(new
    OnCompleteListener<Void>() {

        @Override

        public void onComplete(@NonNull Task<Void> task) {

            if(task.isSuccessful()){

                pd.dismiss();
                Toast.makeText(getActivity(),"Posted",Toast.LENGTH_SHORT).show();

                Intent j=new Intent(getActivity(), HomeActivity.class);

                getActivity().startActivity(j);}

            else{

                pd.dismiss();

```



```
String msg=task.getException().toString();

Toast.makeText(getActivity(),"Error"+msg,Toast.LENGTH_SHORT).show();

}} });}}
```

### **Posted fragment**

```
package com.example.payingguest.fragments;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.fragment.app.Fragment;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.MenuInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.PopupMenu;

import android.widget.TextView;

import com.example.payingguest.R;

import com.example.payingguest.adapter.BookAdapter;

import com.example.payingguest.adapter.PostAdapter;
```

```

import com.example.payingguest.model.BookModel;

import com.example.payingguest.model.homesInFeedModel;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class PostedPg extends Fragment {

    PostAdapter postAdapter;

    TextView t;

    RecyclerView recyclerView;

    DatabaseReference data;

    ArrayList<homesInFeedModel> list;

    FirebaseAuth auth;

    FirebaseUser user;

    public PostedPg() {}

```

@Override

```
public void onCreate(Bundle savedInstanceState) {  
  
    super.onCreate(savedInstanceState);}
```

@Override

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
  
    Bundle savedInstanceState) {
```

```
    // Inflate the layout for this fragment
```

```
    View v= inflater.inflate(R.layout.fragment_posted_pg, container, false);
```

```
    recyclerView=v.findViewById(R.id.postrecycler);
```

```
    t=v.findViewById(R.id.txtpost);
```

```
    t.setVisibility(v.INVISIBLE);
```

```
    auth=FirebaseAuth.getInstance();
```

```
    user=FirebaseAuth.getInstance().getCurrentUser();
```

```
    data= FirebaseDatabase.getInstance().getReference("Rent_posts");
```

```
    recyclerView.setHasFixedSize(true);
```

```
    recyclerView.setLayoutManager(new
```

```
        LinearLayoutManager(getActivity()));
```

```
    list=new ArrayList<>();
```

```
    postAdapter=new PostAdapter(getActivity(),list);
```

```

recyclerView.setAdapter(postAdapter);

String userid=getCurrentUser();

data.orderByChild("user").equalTo(userid).addValueEventListener(new
 ValueEventListener() {

    @Override

    public void onDataChange(@NonNull DataSnapshot snapshot) {

        for (DataSnapshot dataSnapshot:snapshot.getChildren()){

            homesInFeedModel pg=dataSnapshot.getValue(homesInFeedModel.class);

            list.add(pg);}

        postAdapter.notifyDataSetChanged();

    @Override

    public void onCancelled(@NonNull DatabaseError error) {} });

return v;}

public String getCurrentUser() {

    user=auth.getCurrentUser();

    if(user!=null){

        return user.getEmail();

    }else{return null;}}}

```

### **Booked PG**

```
package com.example.payingguest.fragments;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.fragment.app.Fragment;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import com.example.payingguest.R;

import com.example.payingguest.adapter.BookAdapter;

import com.example.payingguest.model.BookModel;

import com.example.payingguest.model.homesInFeedModel;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;
```

```

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class BookedPg extends Fragment {

    BookAdapter bookAdapter;

    TextView t;

    RecyclerView recyclerView;

    DatabaseReference data;

    ArrayList<BookModel> list;

    FirebaseAuth auth;

    FirebaseUser user;

    public BookedPg() {}

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);}

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                               Bundle savedInstanceState) {

```

```

View v= inflater.inflate(R.layout.fragment_booked_pg, container, false);

recyclerView=v.findViewById(R.id.bookrecycler);

t=v.findViewById(R.id.txtbooked);

t.setVisibility(v.INVISIBLE);

auth=FirebaseAuth.getInstance();

user=FirebaseAuth.getInstance().getCurrentUser();

data= FirebaseDatabase.getInstance().getReference("Booked_pg");

recyclerView.setHasFixedSize(true);

recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));

list=new ArrayList<>();

bookAdapter=new BookAdapter(getActivity(),list);

recyclerView.setAdapter(bookAdapter);

Stringuserid=getCurrentUser();
data.orderByChild("user").equalTo(userid).addValueEventListener(new
 ValueEventListener() {

    @Override

    public void onDataChange(@NonNull DataSnapshot snapshot) {

        for (DataSnapshot dataSnapshot:snapshot.getChildren()){

            BookModel pg=dataSnapshot.getValue(BookModel.class);

            list.add(pg);}

```

```

        bookAdapter.notifyDataSetChanged();

        @Override

        public void onCancelled(@NonNull DatabaseError error) {}

        return v;}

        public String getCurrentUser() {

        user=auth.getCurrentUser();

        if(user!=null){

            return user.getEmail();

        }else{ return null;}});

```

### **Booked adapter**

```

package com.example.payingguest.adapter;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

```



```

import com.example.payingguest.R;

import com.example.payingguest.model.BookModel;

import com.example.payingguest.model.homesInFeedModel;

import com.squareup.picasso.Picasso;

import java.util.ArrayList;

public class BookAdapter extends RecyclerView.Adapter<BookAdapter.MyViewHolder> {

    Context context;

    ArrayList<BookModel> list;

    public BookAdapter(Context context, ArrayList<BookModel> list) {

        this.context = context;

        this.list = list;
    }

    @NonNull

    @Override

    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

        View vi= LayoutInflater.from(context).inflate(R.layout.booked_design,parent,false);

        return new MyViewHolder(vi);}

    @Override

    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {

```

```

BookModel pg = list.get(position);

holder.brent.setText(pg.getRent());

holder.btype.setText(pg.getRoomtype());

holder.date.setText(pg.getBookedDate());

holder.bamount.setText(pg.getAmountpaid());

Picasso.get().load(pg.getImage()).into(holder.bpic);}

@Override

public int getItemCount() {

    return list.size();}

public static class MyViewHolder extends RecyclerView.ViewHolder{

    TextView brent,btype,date,bamount;

    ImageView bpic;

    public MyViewHolder(@NonNull View itemView) {

        super(itemView);

        brent=itemView.findViewById(R.id.brent);

        btype=itemView.findViewById(R.id.btype);

        date=itemView.findViewById(R.id.bdate);

        bamount=itemView.findViewById(R.id.bamt);

        bpic=itemView.findViewById(R.id.bookPic);} }

```

```
}
```

### **Confirm rent**

```
package com.example.payingguest;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.appcompat.widget.AppCompatButton;

import android.annotation.SuppressLint;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import com.example.payingguest.model.homesInFeedModel;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;
```

```

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;

import java.util.Objects;

public class confirm_rent extends AppCompatActivity {

    EditText mname,address,contno;

    AppCompatButton confirm;

    String nam,addrss,numb;

    String mpattern="[0-9]{10}";

    DatabaseReference pgref;

    String pId;

    FirebaseAuth auth;

    FirebaseUser user;

    @SuppressWarnings("MissingInflatedId")

    @Override

    protected void onCreate(Bundle savedInstanceState) {

```

```

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_confirm_rent);

rname=findViewById(R.id.rname);

address=findViewById(R.id.raddress);

contno=findViewById(R.id.rcontact);

confirm=findViewById(R.id.confirm);

auth=FirebaseAuth.getInstance();

user=FirebaseAuth.getInstance().getCurrentUser();

pId=getIntent().getStringExtra("pId");

confirm.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {

        collect(); }); }

private void collect() {

    nam=rname.getText().toString();

    addrss=address.getText().toString();

    numb=contno.getText().toString();

    if(nam.isEmpty()){

        rname.setError("Enter your name");

```

```

    } else if (addrss.isEmpty()) {

        address.setError("Please provide your address");

    } else if (!numb.matches(mpattern)) {

        contno.setError("Enter valid phone number");

    } else { storedata(); } }

private void storedata() {

    DatabaseReferencerentRef=FirebaseDatabase.getInstance().getReference().child
    ("ConfirmRent");

    HashMap<String,Object> map= new HashMap<>();

    map.put("RenterName",nam);

    map.put("ContactNo",numb);

    map.put("Address",addrss);
    rentRef.child(auth.getCurrentUser().getUid()).setValue(map).addOnCompleteListener
    er(new OnCompleteListener<Void>() {

        @Override

        public void onComplete(@NonNull Task<Void> task) {

            if(task.isSuccessful()){

                pgref=FirebaseDatabase.getInstance().getReference().child("Rent_posts");

                pgref.addListenerForSingleValueEvent(new ValueEventListener() {

                    @Override

```

```

        public void onDataChange(@NonNull DataSnapshot snapshot) {

            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                homesInFeedModel pg=dataSnapshot.getValue(homesInFeedModel.class);

                if (dataSnapshot.child("pId").getValue() != null) {

                    if(Objects.requireNonNull(dataSnapshot.child("pId").getValue()).toString().equals(pId)) {

                        Intent intent = new Intent(confirm_rent.this, payment.class);

                        intent.putExtra("pId", pg.getId());

                        intent.putExtra("rent",pg.getMonthlyRent());

                        intent.putExtra("type",pg.getRoomType());

                        intent.putExtra("amount",pg.getSecurityDeposit());

                        intent.putExtra("bookimage",pg.getImage());

                        startActivity(intent);

                        finish();

                        Break;}}}}

        @Override

        public void onCancelled(@NonNull DatabaseError error) {

            Toast.makeText(confirm_rent.this, "Error retrieving pg",
            Toast.LENGTH_SHORT).show();}

    } else {

```

```

        Toast.makeText(confirm_rent.this, "Error storing data",
        Toast.LENGTH_SHORT).show();}}

    }).addOnFailureListener(new OnFailureListener() {

        @Override

        public void onFailure(@NonNull Exception e) {

            Toast.makeText(confirm_rent.this,"error",Toast.LENGTH_SHORT).show();

            }});}

    public String getCurrentUser() {

        user=auth.getCurrentUser();

        if(user!=null){

            return user.getEmail();

        }else{ return null; }}}

    }

```

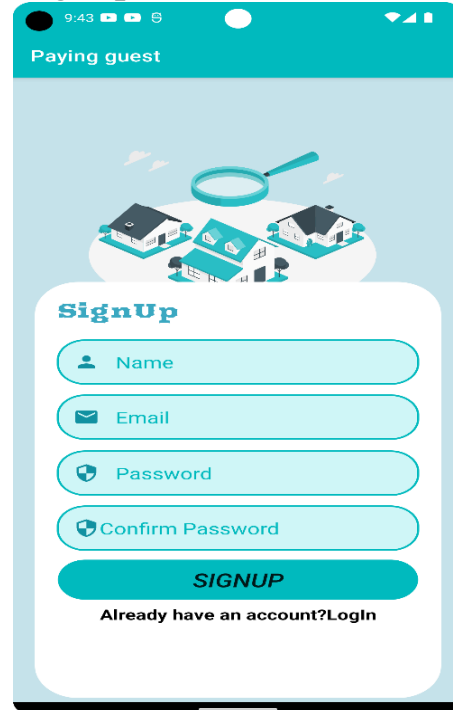


## 6. User Interfaces

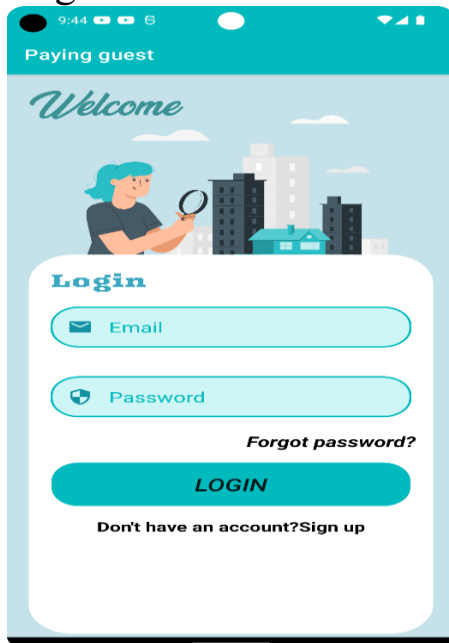
Splashscreen.xml



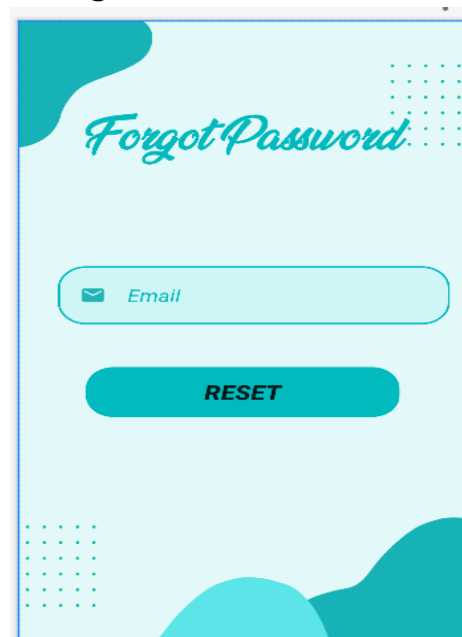
Signup.xml



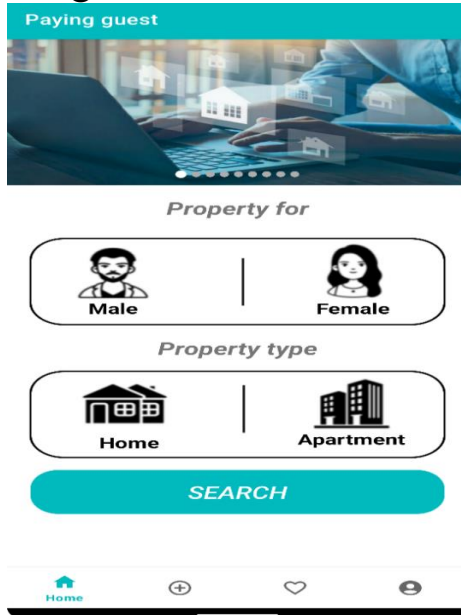
Login.xml



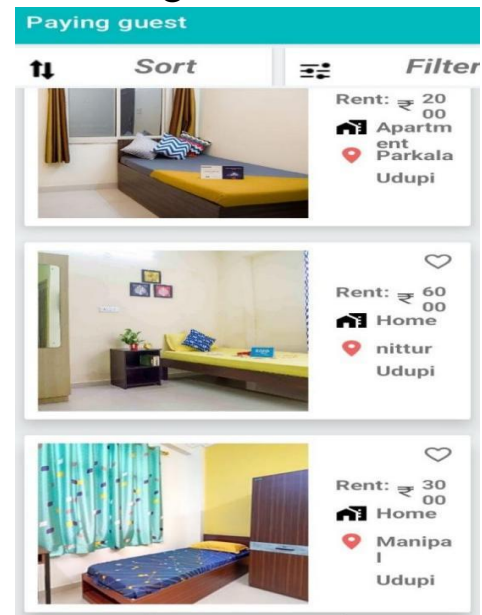
Forgot.xml



## FragmentHome.xml



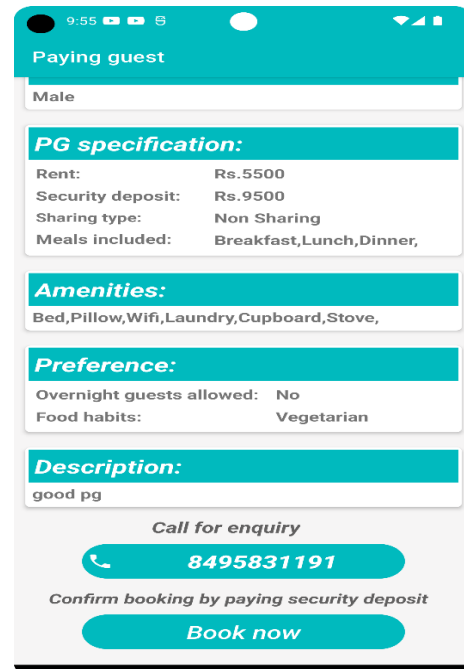
## HomePage.xml



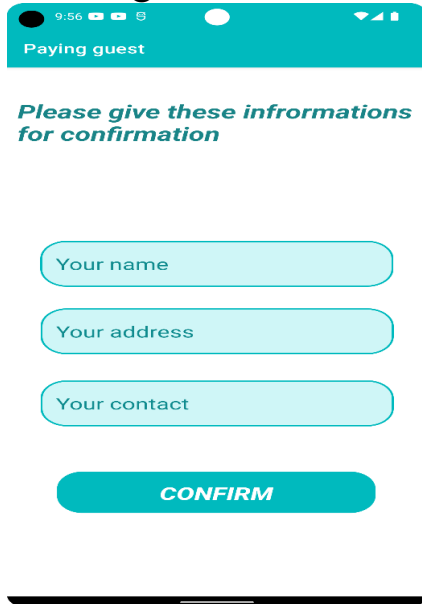
## Homedetails.xml



## Homedetails.xml

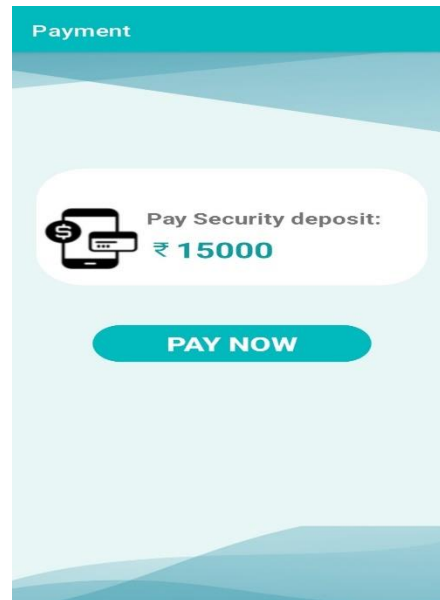


## Booking.xml



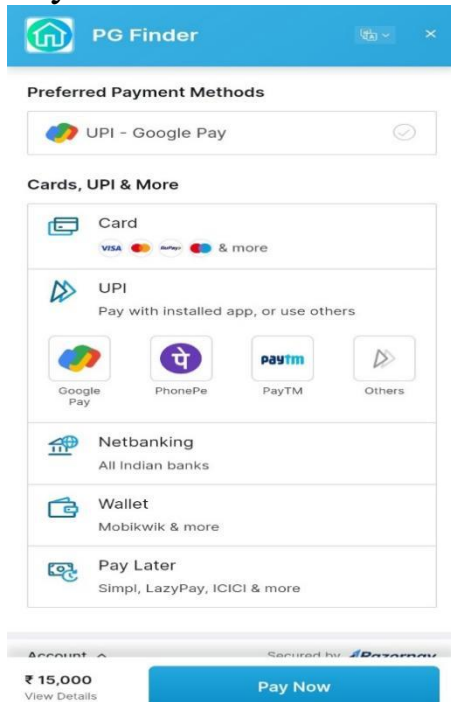
A mobile app interface for booking confirmation. At the top, a teal header bar contains the time '9:56' and the text 'Paying guest'. Below the header, a teal bar displays the text 'Please give these informations for confirmation'. The main area features three light blue rounded rectangular input fields labeled 'Your name', 'Your address', and 'Your contact'. Below these fields is a teal button with the text 'CONFIRM' in white. At the bottom, a black horizontal bar represents a mobile home indicator.

## Payment.xml



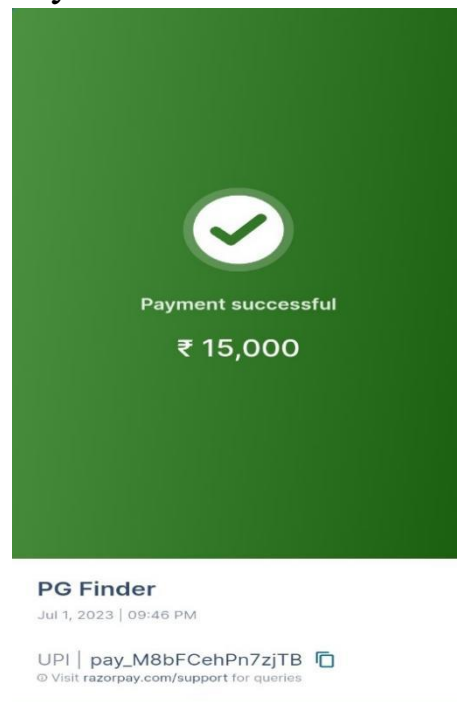
A mobile app interface for payment. The top has a teal header bar with the text 'Payment'. Below the header, a light blue background features a white rounded rectangular box containing a smartphone icon with a dollar sign and the text 'Pay Security deposit: ₹ 15000'. Below this box is a teal button with the text 'PAY NOW' in white.

## PaymentDetails.xml



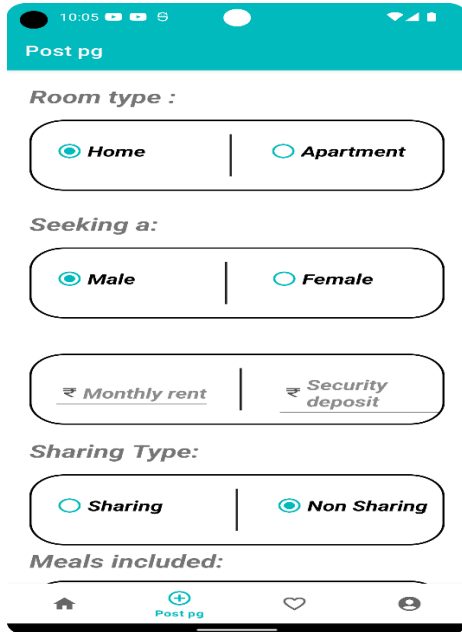
A mobile app interface for payment details. The top has a blue header bar with the 'PG Finder' logo and a close button. Below the header, the text 'Preferred Payment Methods' is displayed. Under this text, there is a white box with the 'UPI - Google Pay' option selected, indicated by a checkmark. Below this, the text 'Cards, UPI & More' is displayed. Under this text, there are several payment options: 'Card' (with VISA, Mastercard, and others), 'UPI' (with a note to 'Pay with installed app, or use others'), 'Netbanking' (with a note to 'All Indian banks'), 'Wallet' (with a note to 'Mobikwik & more'), and 'Pay Later' (with a note to 'Simpli, LazyPay, ICICI & more'). At the bottom, there is a white bar with the text 'Account' and 'Secured by Razorpay'. Below this bar, there is a white box with the text '₹ 15,000' and a 'View Details' link, and a blue button with the text 'Pay Now'.

## PaymentSuccesfull.xml



A mobile app interface for payment success. The top has a green header bar. Below the header, a green background features a white circle with a green checkmark. Below the checkmark, the text 'Payment successful' is displayed. Below this text, the text '₹ 15,000' is displayed. At the bottom, there is a white bar with the text 'PG Finder', the date and time 'Jul 1, 2023 | 09:46 PM', and the text 'UPI | pay\_M8bFCehPn7zjTB' with a QR code icon. Below this bar, there is a white box with the text '© Visit razorpay.com/support for queries'.

## PostPG.xml



Post pg

Room type :

☒ Home | ☐ Apartment

Seeking a:

☒ Male | ☐ Female

₹ Monthly rent |  ₹ Security deposit

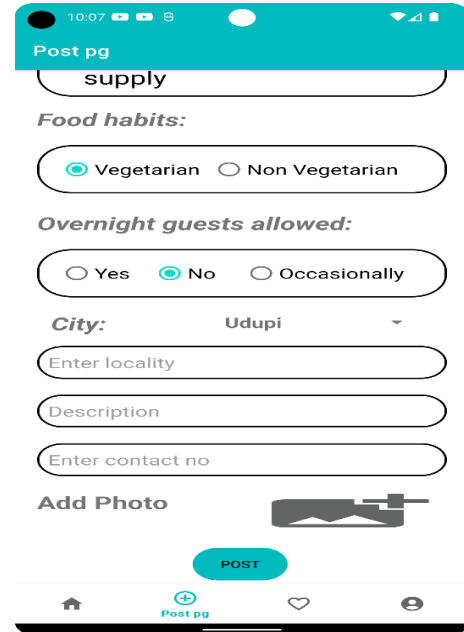
Sharing Type:

☐ Sharing | ☒ Non Sharing

Meals included:

Home | Post pg | Heart | Profile

## + PostPG.xml



Post pg

supply

Food habits:

☒ Vegetarian | ☐ Non Vegetarian

Overnight guests allowed:

☐ Yes | ☒ No | ☐ Occasionally

City: Udupi

Enter locality

Description

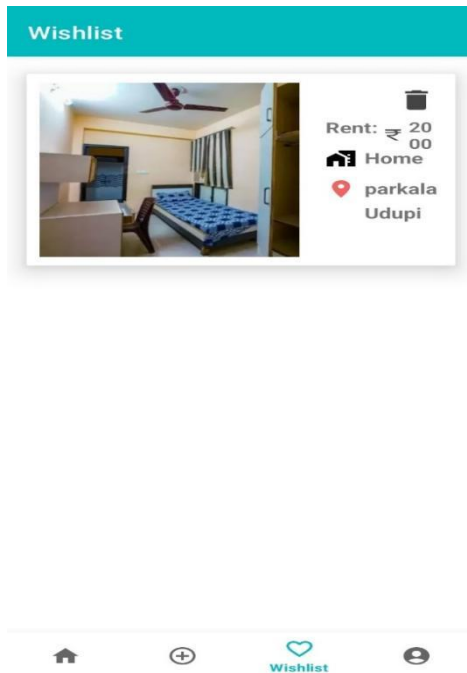
Enter contact no

Add Photo


POST

Home | Post pg | Heart | Profile

## Wishlist.xml



Wishlist



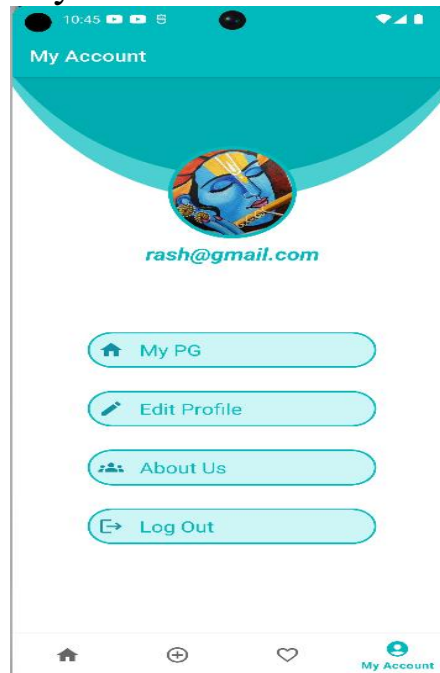
Rent: ₹ 20 00

Home


parkala Udupi

Home | Post pg | Heart | Profile

## MyAccount.xml



My Account



rash@gmail.com

My PG

Edit Profile

About Us

Log Out


Home | Post pg | Heart | My Account

## BookedPg.xml

Paying guest

Posted Pg

Booked Pg



Rent: ₹ 2000

Home

Booked on: 07 01, 2023

Amount paid: 15000

## EditProfile.xml

Update User Details

CHANGE PASSWORD

UPDATE PROFILEPICTURE

UPDATE EMAIL

## Changepassword.xml

Change Password

Current Password

Password

AUTENTICATE

New Password

Password

CANCEL


CHANGE

## UpdateProfile.xml

Edit profile

Update User Details

Upload Profile Image



CANCEL

UPLOAD

## Updateemail.xml

**Update your Email Address**  
You can update your email address after you authenticate your profile by entering your password.

✉ Current Email

Password\*

enter your password

AUTHENTICATE

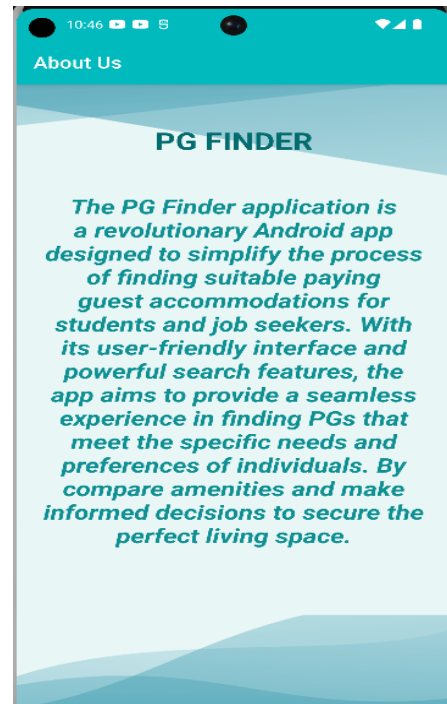
Enter your new e mail

✉ New Email\*

enter your new email address

UPDATE CANCEL

## AboutUs.xml



## 7. Testing

### a. Introduction:

Testing and implementation is the process, which tells the reality of efficiency and the flexibility of the system design. Reliability means how much the user is expecting from the system. Flexibility tells how much the user is comfortable and hopes for additional facilities with the system.

It is vital to the success of the system. System testing makes the logical assumption that if all parts of the system are correct, the goal will be successively achieved. It is a critical element of software quality assurance and represents the ultimate review of specification,

design and coding. Testing presents an interesting anomaly of the software. The testing phase involves testing a system using various test data.

The first test of the system is to see whether it produces the correct output. When the software is tested the actual output is tested with the expected output. If there is discrepancy the sequence of instruction must be traced to determine the problem. Breaking the program down into self-contained portions, each of which can be checked at certain key points, facilitates the process.

The best program is worthless if it does not meet needs. The first step in system testing is to prepare a plan that will test all aspects of the system in a way that promotes its credibility among potential users. The development of software systems involves a series of production activities where opportunities for injunction of human error are enormous. Error may occur at the very imperfectly specified as well as later design and development stages.

#### **b. Test Reports:**

##### **i. Unit Testing:**

Unit testing focuses on verification efforts on the smallest unit of software design module. Using the unit test plans, prepared in the design phase of the system as guide, important control paths are tested to uncover errors within the boundary of the module. The interfaces of each of the modules under consideration are tested. Boundary condition was checked. All independent phases were exercised to ensure the error handling path was tested. Each unit was thoroughly tested to check if it might fail in any possible situation. This testing was carried out during the programming itself. At the end of the testing phase, each unit was found to be working satisfactorily, as regarded to the expected output from the module

### **ii. Integration Testing:**

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together and tested in small segments, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole.

### **iii. System Testing:**

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The testing phase involves the testing of a system using various test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested. Those test data, error were found and corrected by using some testing steps. Thus a series testing is performed on the system before it is ready for implementation.

### **c. Test Plan:**

<b>TEST CASE</b>	<b>OBJECTIVES</b>
<b>1</b>	<b>Test for signing in to the application</b>
<b>2</b>	<b>Test for login (username &amp; Password)</b>
<b>3</b>	<b>Test for resetting the forgotten password</b>
<b>4</b>	<b>Test for updating the email</b>
<b>5</b>	<b>Test for changing the password</b>



#### **d. Test Cases:**

##### **Test Case 1:**

Test Objective: Test for signing in to the application.

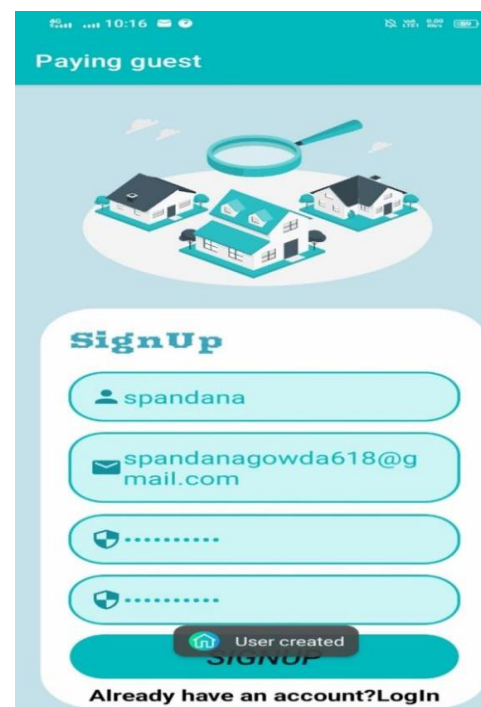
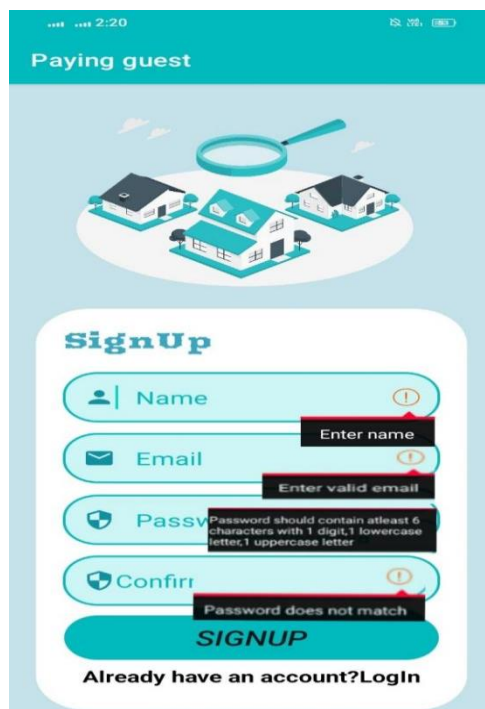
Test Data : valid-All required fields are entered.

Invalid-some fields are incorrect are blank.

Output : valid-allows all records to be added to the database.

Invalid-The user is prompt with an error message.

Result : All the records will be stored in database and user created Successfully.



## Test Case 2:

Test Objective: Test for login (username and password)

Test Data : Valid-correct username and password and click on login button.

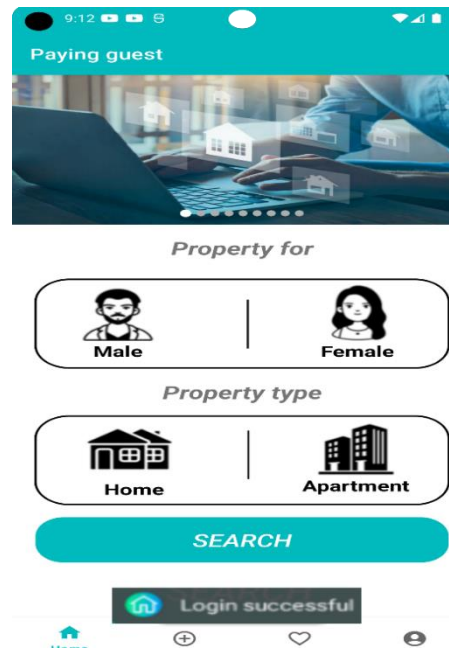
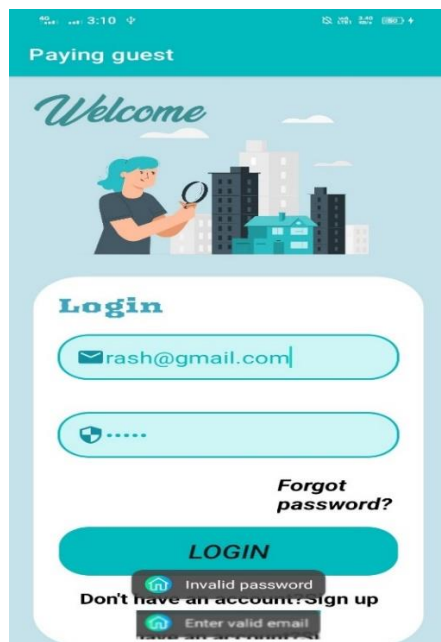
Invalid-incorrect username and password or blank fields.

Output : Valid-User is successfully logged in and he/she can view all the Services.

Invalid-Shows the error message.

Result :Valid-The user logs successfully and allowed to enter the Application.

Invalid-the users are prompt with the error message and restricted from entering the application.



### Test Case 3:

Test Objective: Test for Resetting the forgotten password.

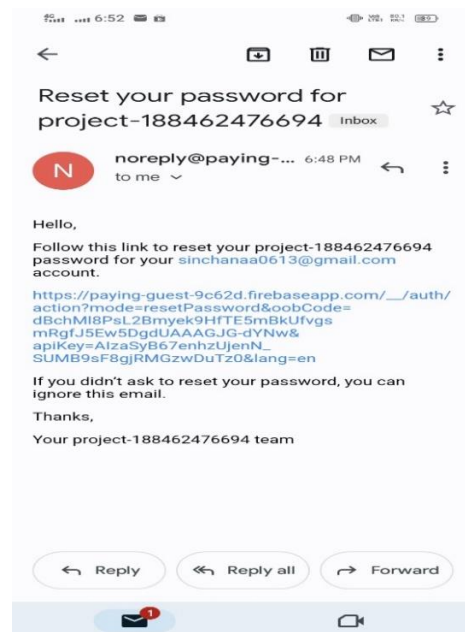
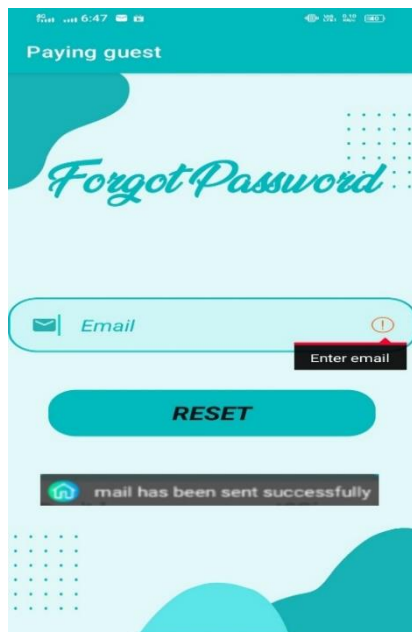
Test Data :Valid-valid user Email address.

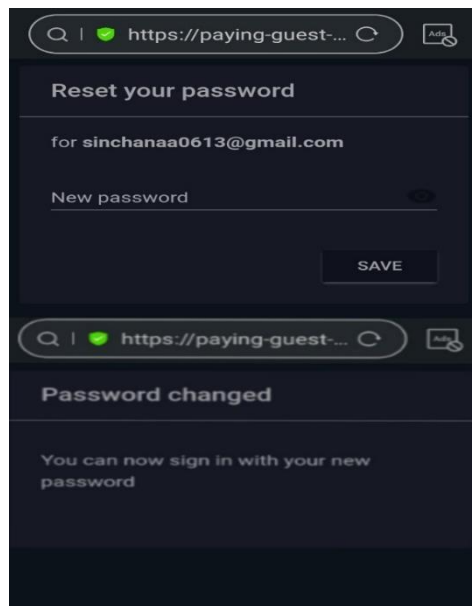
Invalid-Invalid user Email address or blank field.

Output : valid-Email will be sent to valid email address.

Invalid-user will be prompt with the error message.

Result :user will able to rest the password and can login with new password.





#### **Test Case 4:**

Test Objective: Test for updating the email address.

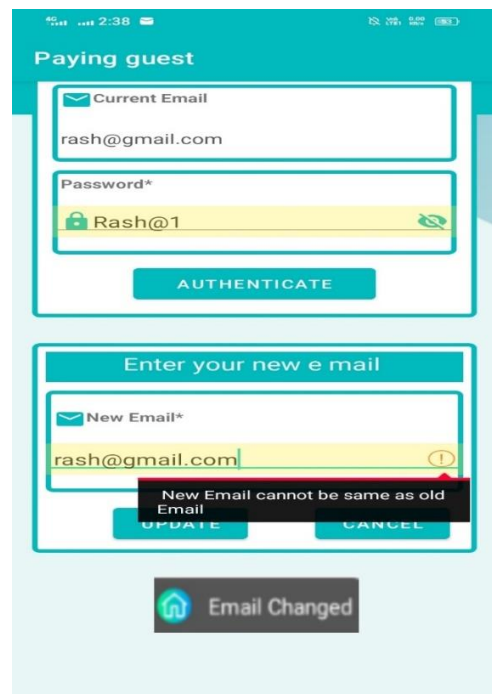
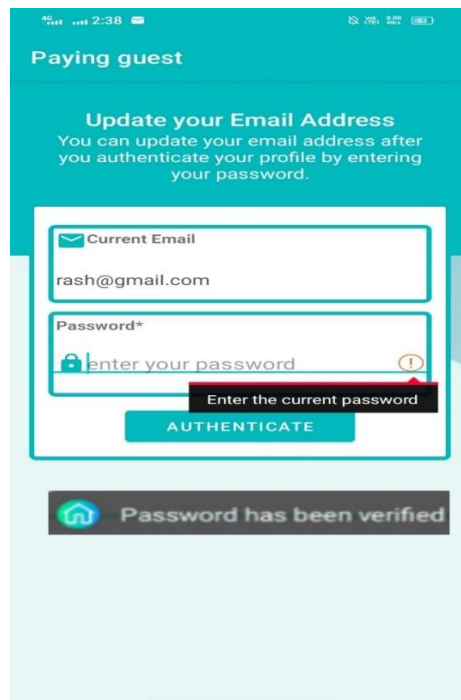
Test Data : valid-All required fields are entered.

Invalid-some fields are incorrect are blank.

Output : valid-allows to enter the new email address and update.

Invalid-The user is prompt with an error message.

Result : All the records will be stored in database and email updated Successfully.



### Test Case 5:

Test Objective: Test for changing the current password.

Test Data : valid-current password.

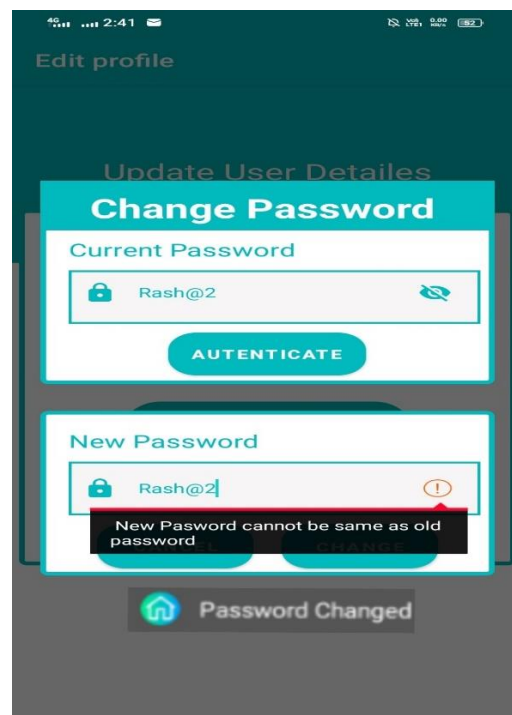
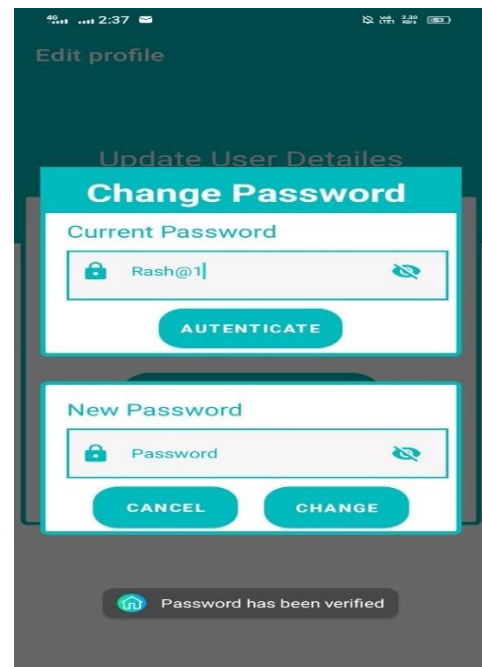
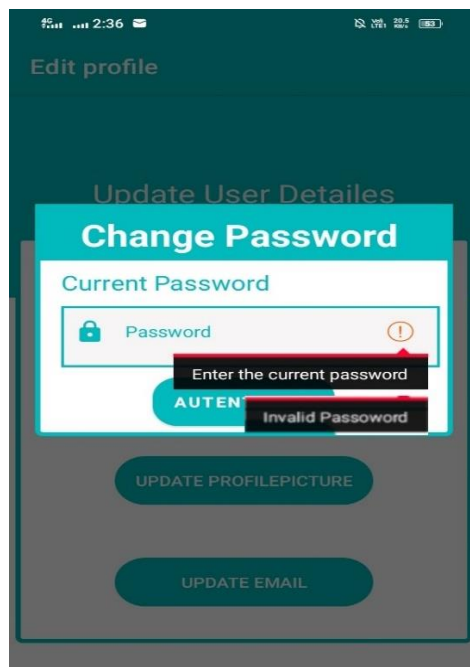
Invalid-incorrect password.

Output : valid-Verifies for existing user with the entered password and allows to enter new password.

Invalid-restricts the user to enter the new password.

Result : Valid-password will be changed successfully.

Invalid-user will be prompted with the error message and restricted from changing the password.



## **8. Limitation**

- No Cross platform support.
- The application requires Internet Connection.
- The application does not give notification when a PG is booked

## **9. Scope for Enhancements**

- This application can be developed for Cross Platform.
- The application can have notifications when a PG is booked.

## **10. Abbreviation and acronyms**

### **Database Reference**

A Firebase Reference represents a particular location in your database and can be used for reading or writing data to that Database location.

### **Data Snapshot**

A DataSnapshot instance contains the data from a Firebase Database location. Any time you read data from the database, you receive the data as a DataSnapshot.

**CFD:** Context Flow Diagram

**DFD:** Data Flow Diagram

**SDK:** Software Development Kit

**API:** Application programming interface

**JDK:** Java Development Kit

**IDK:** Integrated Development Environment

## **11. Bibliography**

The content for this project has been taken from the following sources:

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.razorpay.com](http://www.razorpay.com)

[www.youtube.com](http://www.youtube.com)

**THANK YOU**