MARATHA VIDYA PRASARAK SAMAJ'S
RAJARSHI SHAHU MAHARAJ POLYTECHNIC
NASHIK-422013

A

PROJECT REPORT ON

"LiveInn- PG Locator"

SUBMITTED BY

1.Krushna Ramesh Khairnar   **[315332]**
2. Yash Mahendra Pawar.   **[315347]**
3. Om Vilas Shimpi.   **[375353]**
4. Sushant Ashok Solase.   **[315354]**

UNDER THE GUIDANCE OF

Prof. S. V. Sarode.

DEPARTMENT OF COMPUTER TECHNOLOGY

MARATHA VIDYA PRASARAK SAMAJ'S
RAJARSHI SHAHU MAHARAJ POLYTECHNIC
NASHIK-422013

2021-2022

1

CERTIFICATE

This is to certify that the project report entitled "**Liveinn -pg locator App**" has been successfully completed by:

1. Krushna Ramesh Khairnar   [315332]

2. Yash Mahendra Pawar.   [315347]

3. Om Vilas Shimpi.   [375353]

4. Sushant Ashok Solase.   [315354]

as partial fulfillment of Diploma course in **Computer Technology** under the **Maharashtra State Board of Technical Education, Mumbai** during the academic year **2021-2022.**

The said work has been carried out under my guidance, assessed by us and we are satisfied that, the same is up to the standard envisaged for the level of the course.

Prof. S.V.Sarode	**Prof. P.D.Boraste**	**Prof. D.B.Uphade**
Project Guide	Head of Department	Principal

External Examiner

Institute
Seal

2

ACKNOWLEDGEMENT

With all respect and gratitude, I would like to thank all people who have helped me directly or indirectly for the completion of this Project. I express my heartily gratitude towards **P. D. Boraste** for guiding me to understand the work conceptually and also for his constant encouragement to complete this project on "NGO System". My association with him as a student has been extremely inspiring. I express my sincere thanks to him for kind help and guidance. I would like to give my sincere thanks to **Prof. P. D. Boraste,** project Co-coordinator& Head of Department of Computer Technology Department for providing necessary help, providing facilities and time to time valuable guidance. No words could be good enough to express my deep gratitude to our honorable respected Principal **Dr. D. B. Uphade**, and staff members of Computer Technology department of MarathaVidya Prasarak Samaj's, Rajarshi Shahu Maharaj Polytechnic Nashik, for providing all necessary facilities with their constant encouragement and support.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved **parents** for their blessings, my **friends** and **colleagues** for their help and wishes for the successful completion of this project.

1.Krushna Ramesh Khairnar   **[315332]**

2. Yash Mahendra Pawar   **[315347]**

3. Om Vilas Shimpi   **[375353]**

4. Sushant Ashok Solase   **[315354]**

INDEX

List of the figures

List of the Tables

NOMENCLATURE

D	Rotor diameter
P	Air density
Σbp	Permissible bending stress of pinion
Σbg	Permissible bending stress of gear
Zg	Number of gear teeth
Zp	Number of pinion
M	Module
Φ	Pressure angle
Ka	Application Factor
Km	Load distribution Factor
Nf	Factor of safety
Kr	Velocity factor
Q	Ratio factor for external gear pair
K	Load stress factor
Fr	Radial force
Fa	Axial force
C	Dynamic load
Co	Static load
X	Radial factor
Y	Thrust factor
Pe	Equivalent dynamic
L_{10}	Rating life of bearing
P	Rated power

ABSTRACT

.

In the present system a customer can get only little information like address, contact number and food. Due to lack of information like price, exact location, security measures, the customer is unable to find the best PG of their choice. Often the customer may be misguided. The main objective of the project is to develop an application that provides PG's information regarding the location, facilities, food, price, maps, transportation facility, and safety measures. Any news, information, advertisements, displayed on this website reaches millions of potential customers.The proposed system is a android based application and maintains a centralized repository of all related information. The system allows one to easily access the relevant information and make necessary judgments regarding the PG's selection. User can take a look on different aspects of the information provided like location map, food, price, transportation facility and even security measures and select a best PG of their choice

Chapter: - 1

Introduction

1.1 Overview

In the present system a customer can get only little information like address, contact number and food. Due to lack of information like price, exact location, security measures, the customer is unable to find the best PG of their choice. Often the customer may be misguided. The main objective of the project is to develop an application that provides PG's information regarding the location, facilities, food, price, maps, transportation facility, and safety measures. Any news, information, advertisements, displayed on this website reaches millions of potential customers. The proposed system is a android based application and maintains a centralized repository of all related information. The system allows one to easily access the relevant information and make necessary judgments regarding the PG's selection. User can take a look on different aspects of the information provided like location map, food, price, transportation facility and even security measures and select a best PG of their choice.

The application is divided into 3 modules. User MODULE: After the registration the user can obtain the details of PG located in a place. A user can select the appropriate PG depending on information provided like security, address, and price. PG_OWNER MODULE: After the registration the pg owner has to submit the details of PG based on the verification process. The pg owner can provide information regarding location through address, cost, PG's description, facilities and even security details. The pg owner can also update the status of the PG like whether it is filled or available. ADMINISTRATOR MODULE: This module provides administrator related functionality. Administrator manages all information and has access rights to add, delete, edit and view the data related to PG, User etc.

Our Project "Live inn" is a new way to help people meet their needs for living. It will benefit the community of students, working people, and therefore the larger community as a whole. "Live inn", is an android based application that makes the common people (user) to search for the Paying Guest (PG) all over the city. The PG owners can advertise the PG details by registering into the website which is based on Admin side. Once the admin verifies the details and authenticate the login, the owners will be able to login and can also able to upload the PG details into the application. The users can view those details and make use of this information to search for the PGs according to their needs. The unnecessary need to go to the place physically and ask about the location pricing and others details is very tyring, but with our system this loop hole is filled as the user can search any pc according to the place or city and get all the details of the pg at a button click. Even sometimes the location of pg is very hard to find which in turn leads to frustration which can be skipped with this system.

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

1.2 Applying Software Engineering Approach

The goal of system design is to produce a model or representation that exhibit, commodity and delight. It provides information about the application domain for the software to be built. It fully describes the internal details of each software. Here are some advantages of incremental model: -

1. Each iteration passes through the requirements, design, coding and testing phases.
2. Software will be generated quickly during the software life cycle.
3. It is flexible and less expensive to change requirements and scope.
4. Customer can respond to each built and errors are easy to be identified.
5. Easier to test and debug during a smaller iteration.

Fig.1.1: Incremental Model

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology
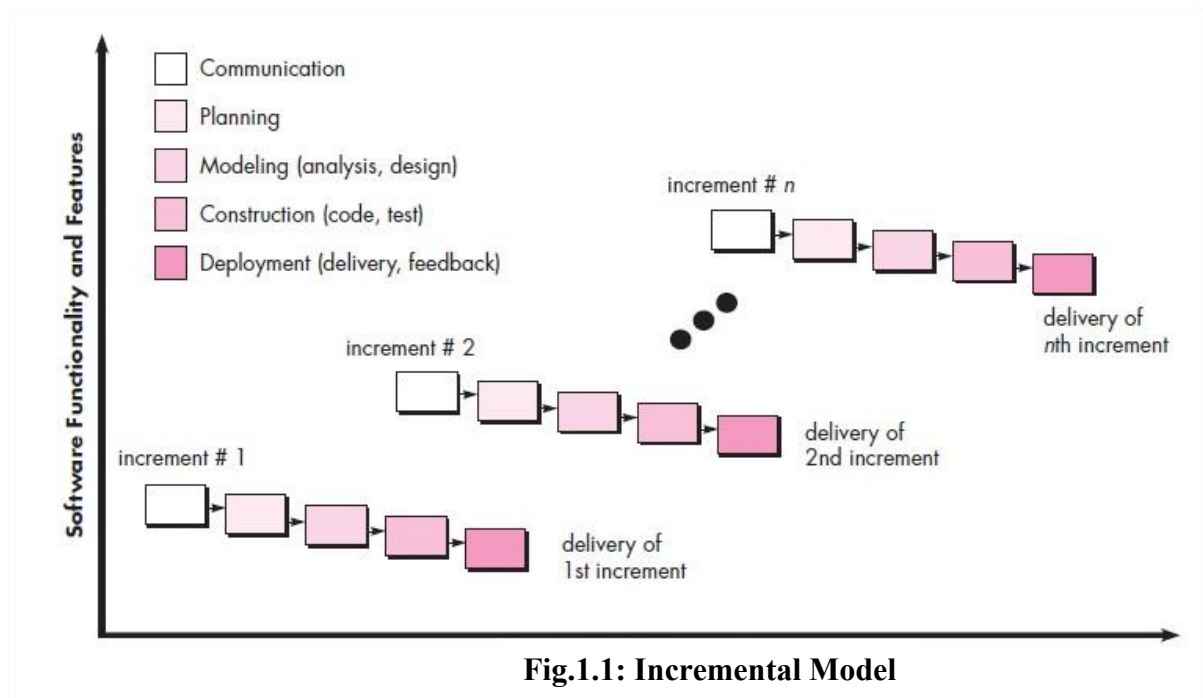
1.2.1 Communication

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

1.2.2 Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry.

1.2.3 Designing and Modeling

Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS.

1.2.4 Construction

This step is also known as programming phase. An estimate says that 50% of whole software development process should be tested. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user's end.

1.2.5 Deployment

Once the product is tested and ready to be deployed it is released formally in the appropriate market. The product may first be released in a limited segment and tested in the real business environment (UAT).

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Chapter: - 2
Literature Review

As compared to the other applications like Zocalo, Pep rooms, Zelo where the app focuses in giving the static information about the PGs, but not the dynamic i.e., there is no in-depth details about the PG's all over the city. Our app, "Live inn" is an informative application which is providing the information based on our city. It will provide the information to those people who are staying in our city as a paying guest. Mainly it will give a list of available houses for rent in their required budget. This application gives detailed information about all the PGs around the city and also, they are all sorted according to the area, so that search can be made easy. Thus, allowing the users to search it according to their need, there are no specific applications on this domain. But there are static apps which does not give a detailed description about this domain. When the users are viewing the different houses, instead of directly booking the house they can added to their favorite. And then from this sorted list of houses users can book the house later on. In our application, we are also going to provide various facilities like Tiffin service, Laundry service, Gym etc. In the Tiffin service, users will be given a choice to choose from different kind of menus like Punjabi thali, Guajarati thali etc.

Users will be also given the facility to take advantage of nearby laundry service, gym etc. "Live inn" is a new way to help people meet their needs for living. It will benefit the community of students, working people, and therefore the larger community as a whole. "Live inn", is an android based application that makes the common people (user) to search for the Paying Guest (PG) all over the city. The PG owners can advertise the PG details by registering into the website which is based on Admin side. Once the admin verifies the details and authenticate the login, the owners will be able to login and can also able to upload the PG details into the application. The users can view those details and make use of this information to search for the PGs according to their needs. This application helps the users to view the PG details, the facilities in the PG and so on. The users can also search the PG's information based on some parameters such as colleges nearby, rent, Wi-Fi etc... This application will be useful to anyone who is in search of a place to stay. This system mainly focuses on the PG owners as well as the common users who are in search of PGs who come from different places to study as well as to work

Rationale of Study

As the name specifies "PG Locator" is an app developed for managing various activities in the hostel. For the past few years, the number of educational institutions is increasing rapidly. Thereby the number of hostels is also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software's are not usually used in this context. This particular project deals with the problems on finding and providing information on a hostel and avoids the problems which occur when carried manually.

Objectives of Study:

a. To understand the evolution of Hostel Finding systems.

b. To describe the conceptual framework of Pg. Locators.

c. To examine the barriers of this technology in India.
d. To predict the future of this technology.

Research Methodology

I have reviewed the academic Literature to gain insight into "Hotel Management and Locator". So, various articles, journals, books, websites etc. have been used to study the evolution, conceptual framework, definitions, key players, present trends (relating to internet penetration, growth prospects, modes of payments preferred etc.), future prospects and barriers of PgLocator. All the data included is second, base and proper references have been given wherever necessary.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system Which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system. · Less human error · Strength and strain of manual labour can be reduced · High security· Data redundancy can be avoided to some extent· Data consistency· Easy to handle· Easy data updating· Easy record keeping· Backup data can be easily generated

Chapter: - 3
Scope of the project

Now-a-days, the process of house hunting is simplified with the introduction of the PG locators. The pain of unnecessarily spending some amount of money on the housing brokers is also reduced to a certain level with the introduction of this app. To develop an application that provides PG's information regarding the location, facilities, food, price, maps, transportation facility, and safety measures is the main purpose of this project.

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Chapter: - 4
Requirement Gathering

4.1 Functional Requirements:

4.1.1 Software Requirements:

1) Platform: Windows 10

   Platform is any hardware used to host an application or service.
2) Language: Java

   A programming language is a formal computer language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behaviour of a machine or to express algorithms.

4.1.2 Hardware Requirements:

1) Processor: Ryzen 5 2400g

   A processor is the logic circuitry that responds to and processes the basic instructions that drive a computer.
2) RAM: 8 GB

   Random-access memory (RAM) is a form of computer data storage which stores frequently used program instructions to increase the general speed of a system.
3) Monitor: 22"

   A computer monitors or a computer display is an electronic visual display for computers.

4.2 Non-Functional Requirements:

1) Person:

   A person/user to search a Pg or room or hostel.
2) Admin:

   Admin manages all the transactions between User and Vendor which helps the app to execute effectively.
3) Owner of hostel/Pg:

   To rent the room to the user.

Chapter: - 5

System Analysis

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers.

System analysis or study is an important phase of any system development process. The system is viewed as a whole, the inputs are identified and the system is subjected to close study to identify the problem areas. The solutions are given as a proposal. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal.

5.1 Existing System:

The current system for shopping is to visit the shop manually and from the available product choose the item customer want and buying the item by payment of the price of the item.

1. It is less user-friendly.

2. User must go to hostel and select room.

3. It is difficult to locate the required place.

4. Description of the product limited.

5. It is a time-consuming process.

6. Not in reach of distant users.

5.2 Proposed System

In the proposed system customer need not go to the location for renting the pg. He can view the details of the pg/room on the app and can even contact the owner if they decide to rent it. The room owner will be vendor of the system. Room/pg owner can appoint moderators who will help owner in managing the customers.

5.3 System Requirement Specification

5.3.1 General Description:

Problem Statement:

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

As we know noa days there are a lot of room/pg available in India as the number of students are ever increasing, it is very hard for the students to manually got o every pg and know its details and it was very time consuming, this project helps users to find the perfect pg/room for them and book it online without manually having to there.

5.3.2 System Requirements:

5.3.3.1 Non-Functional Requirements:

1) Efficiency Requirement

When the system is implemented, the user should be able to find good pg from home.

2) Reliability Requirement

The system should provide a reliable environment to both customers and owner. All requests should be reaching at the admin without any errors.

3) Usability Requirement

The android application is designed for user friendly environment and ease of use.

4) Implementation Requirement

Implementation of the system using android studio, XML for front end with firebase as back end and it will be used for database connectivity. And the database part is developed by google. Responsive designing is used for making the app compatible for any type of screen.

5) Delivery Requirement

The whole system is expected to be delivered in four months of time with a weekly evaluation by the project guide.

5.3.3.2 Functional Requirements

1) USER: -

➢ User Login: -

Description of User: -
This feature used by the user to login into system. A user must login with his user's name and password to the system after registration. If they are invalid, the user not allowed to enter the system.

Functional requirement: -
1) Username and password will be provided after user registration is confirmed.
2) Password should be hidden from others while typing it in the field

➢ Register New User: -

Description of feature: -
A new user will have to register in the system by providing essential details in order to view the products in the system.

Functional requirement: -
1) System must be able to verify and validate information.

9

2) The system must encrypt the password of the customer to provide security.

➢ Viewing an Item: -

Description of feature: -
The user will be able to view different room/pg at different locations after login. The user should be able to search pg and even view their details.
Functional requirement: -

System must ensure that, only a registered customer can purchase items.

2) ADMIN: -
➢ Manage User: -

Description of feature; -
The administrator can delete user, view user and block user.

➢ Manage Vendors: -

Description of feature: -
The administrator can delete vendors, block vendors and search for a vendor.

➢ Manage pg/room: -

Description of feature: -
The administrator can delete product and view room/pg.

3) Vendor's: -

➢ Add pg detail: -
Description of Feature: -
The Vendor can Add new details into the database with all its information.

➢ Edit Product: -

Description of Feature: -
The Vendor can Edit the information of this pg and also Delete the pg from database;

➢ View Product: -

Description of Feature: -
The Vendor will be able to View all the pg details in this tab and is given the option to delete all pg details from database.

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

➢ Manage Order's: -

Description of Feature: -
The Vendor will be able to view and accept or reject any offer from this tab.

Function Requirement's: -
1) The Vendor should be able to view only his added pg's.
2) The orders should be displayed to the vendor who has the pg.

Chapter: - 6

CODE and Output

```
package com.dd.pglocator.nearbyScreen;

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

import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.dd.pglocator.LoginActivity2;
import com.dd.pglocator.MainActivity;
import com.dd.pglocator.PaymentActivity;
import com.dd.pglocator.R;
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 com.muddzdev.styleabletoast.StyleableToast;
```

11

```java
public class PgBookingActivity extends AppCompatActivity {

    EditText e1, e2,e3,e4;
    TextView t1;
    int rentt, monthst, depositt;

    public String s1,s2,s5,s3,s4,pgid,pgname;
    public String rent,deposit;
    Integer is2;
    DatabaseReference myRef, refg;
    FirebaseAuth auth;
    FirebaseUser user;
    PgBookingModel pgBooking;
    long id=0;
    Switch switch1;
    String userid,custname,mobilenos,email;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pg_booking);
        getSupportActionBar().hide();

        e1 = findViewById(R.id.editText_guests);
        e2 = findViewById(R.id.editTextmonth);
        e3 = findViewById(R.id.editTextDate1);
        e4 = findViewById(R.id.peditTextDate2);
        t1 = findViewById(R.id.deposit_txt);

        switch1=findViewById(R.id.booking_switch);



       // Toast.makeText(getApplicationContext(),s2,Toast.LENGTH_LONG).show();

        pgid=getIntent().getStringExtra("id");
        pgname=getIntent().getStringExtra("name");
        rent=getIntent().getStringExtra("rent");
        deposit=getIntent().getStringExtra("deposit");

        TextView pgname_txt = findViewById(R.id.pgname_txt);
        pgname_txt.setText(pgname);

        auth= FirebaseAuth.getInstance();
        user=auth.getCurrentUser();
        userid=user.getUid();
```

12

```java
FirebaseDatabase database = FirebaseDatabase.getInstance();
myRef = database.getReference().child("PgBookings");

pgBooking=new PgBookingModel();


myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if(dataSnapshot.exists())
        {
            id=(dataSnapshot.getChildrenCount());
        }

    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});

final FirebaseDatabase db = FirebaseDatabase.getInstance();
final DatabaseReference myref = db.getReference("Customers");
int id=0;
myref.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot ds) {
        //Toast.makeText(EditActivity.this,"adasdsdf",Toast.LENGTH_LONG).show();
        for (DataSnapshot npsnapshot : ds.getChildren()){

            custname=ds.child(userid).child("customername").getValue().toString();
            mobilenos=ds.child(userid).child("mobileno").getValue().toString();
            email=ds.child(userid).child("email").getValue().toString();


        }
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {

Toast.makeText(PgBookingActivity.this,databaseError.getMessage(),Toast.LENGTH_LONG).show();
    }
});

switch1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
```

13

```java
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        savedata();
        startActivity(new Intent(PgBookingActivity.this, PaymentActivity.class));
    }
});
}


public void savedata()
{
    try {

        final ProgressDialog progressDialog = new ProgressDialog(PgBookingActivity.this);
        progressDialog.setTitle("Saving...");
        progressDialog.show();

        // Toast.makeText(getApplicationContext(), s2, Toast.LENGTH_LONG).show();

        s1 = e1.getText().toString();
        s2 = e2.getText().toString();
        s3 = e3.getText().toString();
        s4 = e4.getText().toString();
        s5 = t1.getText().toString();

        monthst = Integer.parseInt(s2);
        rentt = Integer.parseInt(rent);

        depositt = Integer.parseInt(deposit);

        // Toast.makeText(getApplicationContext(),rentnew, Toast.LENGTH_LONG).show();
        int total = 0;
        total = (rentt * monthst) + depositt;
        // total = total * Integer.parseInt(s1);
        t1.setText(String.valueOf(total)+" ₹");


        pgBooking.setPgid(pgid);
        pgBooking.setPgname(pgname);
        pgBooking.setNoofguests(s1);
        pgBooking.setPrice(String.valueOf(total));
        pgBooking.setCheckindate(s3);
        pgBooking.setCheckoutdate(s4);
        pgBooking.setCustomername(custname);
        pgBooking.setCust_mobileno(mobilenos);
        pgBooking.setCust_email(email);

        myRef.child(String.valueOf(id + 1)).setValue(pgBooking);
        progressDialog.dismiss();
```

14

```java
        // Toast.makeText(getApplicationContext(), "Booking at PG Successfull "+total,
Toast.LENGTH_SHORT).show();
        new StyleableToast.Builder(PgBookingActivity.this)
                .text("Booking at PG Successfull")
                .font(R.font.montserrat_bold)
                .textColor(Color.WHITE)
                .backgroundColor(Color.BLACK)
                .show();


    }
    catch(Exception e)
    {
        Toast.makeText(getApplicationContext(), "Error :"+e, Toast.LENGTH_SHORT).show();

    }

  }
}

package com.dd.pglocator;

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

import com.google.android.material.bottomnavigation.BottomNavigationView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.recyclerview.widget.RecyclerView;

import com.dd.pglocator.databinding.ActivityMainBinding;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private ActivityMainBinding binding;
    Button login;


    @Override
```

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    BottomNavigationView navView = findViewById(R.id.nav_view);
    // Passing each menu ID as a set of Ids because each
    // menu should be considered as top level destinations.
    AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
            R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications)
            .build();
    NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment_activity_main);
    //NavigationUI.setupActionBarWithNavController(this, navController, appBarConfiguration);
    NavigationUI.setupWithNavController(binding.navView, navController);

}

@Override
public void onClick(View v) {
    if(v == login)
    {

    }
}
}

package com.dd.pglocator;

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

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.Toast;

import com.basgeekball.awesomevalidation.AwesomeValidation;
import com.google.android.gms.tasks.OnCompleteListener;
```

16

```java
import com.google.android.gms.tasks.Task;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.FirebaseDatabase;

public class RegisterActivity extends AppCompatActivity {

    EditText e1,e2,e3,e4,e5,e6,e7;
    Button b1;
    FloatingActionButton fab;

    private FirebaseAuth mAuth;
    FirebaseUser user;
    AwesomeValidation awesomeValidation;
    int id=0;


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

//      getSupportActionBar().hide();

        e1=findViewById(R.id.reg_nametxt);
        e2=findViewById(R.id.reg_emailtext);
        e3=findViewById(R.id.reg_mobtext);
        e4=findViewById(R.id.reg_occtext);
        e5=findViewById(R.id.reg_addresstext);
        e6=findViewById(R.id.reg_passtext);
        e7=findViewById(R.id.reg_cpasstext);

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


        fab = findViewById(R.id.fab2);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                registeruser();
                Intent i = new Intent(RegisterActivity.this,MainActivity.class);
                startActivity(i);
            }
        });
    }
```

17

```java
public void registeruser()
{
    final String name = e1.getText().toString().trim();
    final String email = e2.getText().toString().trim();
    final String mobileno = e3.getText().toString().trim();
    final String occupation = e4.getText().toString().trim();
    final String address = e5.getText().toString().trim();
    final String password = e6.getText().toString().trim();
    final String con_password = e7.getText().toString().trim();

    if (name.isEmpty()) {
        e1.setError(getString(R.string.input_error_name));
        e1.requestFocus();
        return;
    }
    if (occupation.isEmpty()) {
        e4.setError(getString(R.string.input_error_Occupation));
        e4.requestFocus();
        return;
    }
    if (address.isEmpty()) {
        e5.setError(getString(R.string.input_error_address));
        e5.requestFocus();
        return;
    }
    if (email.isEmpty()) {
        e2.setError(getString(R.string.input_error_email));
        e2.requestFocus();
        return;
    }

    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        e2.setError(getString(R.string.input_error_email_invalid));
        e2.requestFocus();
        return;
    }

    if (password.isEmpty()) {
        e6.setError(getString(R.string.input_error_password));
        e6.requestFocus();
        return;
    }

    if (password.length() < 6) {
        e6.setError(getString(R.string.input_error_password_length));
        e6.requestFocus();
        return;
    }
```

18

```java
if (mobileno.isEmpty()) {
    e3.setError(getString(R.string.input_error_phone));
    e3.requestFocus();
    return;
}


if (mobileno.length() != 10) {
    e3.setError(getString(R.string.input_error_phone_invalid));
    e3.requestFocus();
    return;
}
if (password.length() != con_password.length()) {
    e7.setError(getString(R.string.input_error_correct_password));
    e7.requestFocus();
    return;
}
if (con_password.length() < 6) {
    e7.setError(getString(R.string.input_error_password_length));
    e7.requestFocus();
    return;
}
if (con_password.isEmpty()) {
    e7.setError(getString(R.string.input_error_password));
    e7.requestFocus();
    return;
}


final ProgressDialog mdialog=new ProgressDialog(RegisterActivity.this);
mdialog.setMessage("Please Wait");
mdialog.show();
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if (task.isSuccessful()) {
                    Customer_model user = new
Customer_model(name,email,address,mobileno,occupation,password,con_password);
                    FirebaseDatabase.getInstance().getReference("Customers")
                        .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                        .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            mdialog.dismiss();
                            if (task.isSuccessful()) {
```

19

```
                        Toast.makeText(RegisterActivity.this, getString(R.string.registration_success),
Toast.LENGTH_LONG).show();
                        } else {
                            //display a failure message
                        }
                    }
                });

            } else {
                Toast.makeText(RegisterActivity.this, task.getException().getMessage(),
Toast.LENGTH_LONG).show();
            }
        }
    });

    }

}

package com.dd.pglocator;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.dd.pglocator.nearbyScreen.PgdetailsActivity;
import com.dd.pglocator.nearbyScreen.model;

import java.util.ArrayList;

public class searchadapter extends RecyclerView.Adapter<searchadapter.MyViewHolder> {
    String userid,pgname;
    public Context context;
    ArrayList<model> list;


    public searchadapter(Context context, ArrayList<model> list){
```

20

```java
        this.context = context;
        this.list=list;
    }


    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

        View view=
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.activity_search_adapter,viewGroup,false);
        return new MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int i) {
        model model1=list.get(i);
        holder.pgname.setText(list.get(i).getPgname());
        holder.pgaddress.setText(list.get(i).getAddress());
        holder.rent.setText(list.get(i).getRent());
        Glide.with(holder.ImageView.getContext()).load(model1.getPgimage()).into(holder.ImageView);

        holder.cv1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                pgname = model1.getPgname();
                userid=model1.getUserid();
                Intent i = new Intent(context.getApplicationContext(), PgdetailsActivity.class);
                i.putExtra("name",pgname);
                i.putExtra("userid",userid);
                context.startActivity(i);
            }
        });


    }

    @Override
    public int getItemCount() {
        return list.size();
    }

    public static class MyViewHolder extends RecyclerView.ViewHolder{
        private TextView pgname,pgaddress,rent;
        private ImageView ImageView;
        private CardView cv1;

        public MyViewHolder(View itemView){
```

21

```
        super(itemView);


        pgname=itemView.findViewById(R.id.pgname_search);
        pgaddress=itemView.findViewById(R.id.pgaddress_search);
        rent=itemView.findViewById(R.id.rent_search);
        ImageView = (ImageView)itemView.findViewById(R.id.imageView2);
        cv1=itemView.findViewById(R.id.cardv);
    }
  }
}

package com.dd.pgadmin;

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

import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Patterns;
import android.view.View;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

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.AuthResult;
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.StorageTask;
import com.google.firebase.storage.UploadTask;

public class AddActivity extends AppCompatActivity {

    EditText e1,e2,e3,e4,e5,e6,e7,e8,e9,e10;
```

22

```java
    RadioButton r1,r2,r3,r4;
    Button b1,b2;
    RadioGroup rg1,rg2,rg3,rg4;

    private FirebaseAuth mAuth;
    FirebaseUser user;
    private static final int CHOOSE_IMAGE = 1;
    private Uri imgUrl;
    private StorageTask mUploadTask;
    public  String Imageurl;
    StorageReference storageReference;
    FirebaseStorage storage;
    String userid;

    PgDetailsmodel pgDetailsmodel;
    int id=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);
        getSupportActionBar().hide();

        e1=findViewById(R.id.pgname);
        e2=findViewById(R.id.pgdescription);
        e3=findViewById(R.id.pgaddress);
        e4=findViewById(R.id.pgcontact);
        e5=findViewById(R.id.bhk);
        e6=findViewById(R.id.noofbeds);
        e7=findViewById(R.id.boysorgirls);
        e8=findViewById(R.id.montlyrent);
        e9=findViewById(R.id.deposit);

        pgDetailsmodel=new PgDetailsmodel();

//      r1=findViewById(R.id.rbac);
//      r2=findViewById(R.id.rbnac);
//      r3=findViewById(R.id.rbgym);
//      r4=findViewById(R.id.rbngym);
//      r5=findViewById(R.id.rbfur);
//      r6=findViewById(R.id.rbnfur);
//      r7=findViewById(R.id.rbfood);
//      r8=findViewById(R.id.rbnfood);

        rg1=findViewById(R.id.acnonac);
        rg2=findViewById(R.id.gymnogym);
        rg3=findViewById(R.id.foodnofood);
        rg4=findViewById(R.id.furnofur);
```

23

```
        b1=findViewById(R.id.button);
        b2=findViewById(R.id.button2);

        mAuth = FirebaseAuth.getInstance();
        user=mAuth.getCurrentUser();
        userid=user.getUid();

        storage = FirebaseStorage.getInstance();
        storageReference = storage.getReference();



        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                chooseImage();
            }
        });



        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                registerUser();
            }
        });
    }

    private void chooseImage() {
        Intent i1 = new Intent();
        i1.setType("image/*");
        i1.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(i1, CHOOSE_IMAGE);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CHOOSE_IMAGE && resultCode == RESULT_OK && data != null &&
    data.getData() != null) {
            imgUrl = data.getData();
            // Picasso.with(this).load(imgUrl).into(imageView1);
        }
    }
    public String GetFileExtension(Uri uri) {

        ContentResolver contentResolver = getContentResolver();
        MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();
```

24

```
        // Returning the file Extension.
        return mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri)) ;


    }


    private void registerUser() {

        if (imgUrl != null) {
            final ProgressDialog mdialog=new ProgressDialog(AddActivity.this);
            mdialog.setMessage("Please Wait");
            mdialog.show();
            final StorageReference ref = storageReference.child("Pgimages/" + System.currentTimeMillis() + "." +
GetFileExtension(imgUrl));
            mUploadTask = ref.putFile(imgUrl)
                    .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                        @Override
                        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                            //Toast.makeText(getApplicationContext(),"sdfdsfsdfsdf",Toast.LENGTH_LONG).show();
                            ref.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    Imageurl = uri.toString();

                                    int a = rg1.getCheckedRadioButtonId();
                                    r1 = (RadioButton) findViewById(a);

//Toast.makeText(getApplicationContext(),String.valueOf(r1.getText()),Toast.LENGTH_SHORT).show();
                                    int b = rg2.getCheckedRadioButtonId();
                                    r2 = (RadioButton) findViewById(b);
                                    //
Toast.makeText(getApplicationContext(),String.valueOf(r2.getText()),Toast.LENGTH_SHORT).show();
                                    int c = rg3.getCheckedRadioButtonId();
                                    r3 = (RadioButton) findViewById(c);
                                    int d = rg4.getCheckedRadioButtonId();
                                    r4 = (RadioButton) findViewById(d);



                                    final String pgname = e1.getText().toString().trim();
                                    final String pgdescription = e2.getText().toString().trim();
                                    final String pgaddress = e3.getText().toString().trim();
                                    final String contactno = e4.getText().toString().trim();
                                    final String bhksize = e5.getText().toString().trim();
                                    final String nofbeds = e6.getText().toString().trim();
                                    final String boysorgirl = e7.getText().toString().trim();
                                    final String rent = e8.getText().toString().trim();
                                    final String deposit = e9.getText().toString().trim();
```

25

```
                final String acnoac = r1.getText().toString();
                final String gymnogym = r2.getText().toString();
                final String foodnofood = r3.getText().toString();
                final String furnofur = r4.getText().toString();

                final FirebaseDatabase database = FirebaseDatabase.getInstance();
                final DatabaseReference myref = database.getReference("Nearbypgs");
                pgDetailsmodel.setPgname(pgname);
                pgDetailsmodel.setDescription(pgdescription);
                pgDetailsmodel.setAddress(pgaddress);
                pgDetailsmodel.setContactno(contactno);
                pgDetailsmodel.setBhksize(bhksize);
                pgDetailsmodel.setNo_of_beds(nofbeds);
                pgDetailsmodel.setBoys_girls(boysorgirl);
                pgDetailsmodel.setRent(rent);
                pgDetailsmodel.setDeposit(deposit);
                pgDetailsmodel.setAc_noac(acnoac);
                pgDetailsmodel.setGym_nogym(gymnogym);
                pgDetailsmodel.setFood_nofood(foodnofood);
                pgDetailsmodel.setFur_nofur(furnofur);
                pgDetailsmodel.setUserid(userid);
                pgDetailsmodel.setPgimage(Imageurl);


myref.child(FirebaseAuth.getInstance().getCurrentUser().getUid()).setValue(pgDetailsmodel);
                mdialog.dismiss();
                Toast.makeText(getApplicationContext(),"Added
Successfully",Toast.LENGTH_SHORT).show();
                }
            });
        }
    })


        .addOnFailureListener(new OnFailureListener() {
          @Override
          public void onFailure(@NonNull Exception e) {
             Toast.makeText(getApplicationContext(), "Failed " + e.getMessage(),
Toast.LENGTH_SHORT).show();
          }
        });

    }


  }
}
```

```java
package com.dd.pgadmin;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

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

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;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;


import java.util.ArrayList;

import com.squareup.picasso.Picasso;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.OnProgressListener;
```

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

```java
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.Query;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.ValueEventListener;

import org.jetbrains.annotations.NotNull;

import java.util.List;

public class Booking_adapter extends RecyclerView.Adapter<Booking_adapter.ViewHolder> {
    public Context context;
    ArrayList<PgBookingModel> listData;

    public Booking_adapter(Context context, ArrayList<PgBookingModel> listData) {
        this.listData = listData;
        this.context = context;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.activity_booking_adapter, parent,
false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        PgBookingModel pgBookingModel = listData.get(position);
        holder.t1.setText(pgBookingModel.getCustomername());
        holder.t2.setText(pgBookingModel.getPgname());
        holder.t3.setText(pgBookingModel.getNoofguests());
        holder.t4.setText(pgBookingModel.getCust_mobileno());
        holder.t5.setText(pgBookingModel.getCheckindate());
        holder.t6.setText(pgBookingModel.getCheckoutdate());
        holder.t7.setText(pgBookingModel.getPrice());
    }

    @Override
    public int getItemCount() {
        return listData.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder {
```

28

```java
        TextView t1,t2,t3,t4,t5,t6,t7;
        DatabaseReference myRef, demoRef;

        public ViewHolder(View itemView) {
            super(itemView);

            t1=itemView.findViewById(R.id.custname);
            t2=itemView.findViewById(R.id.pgname);
            t3=itemView.findViewById(R.id.noofguests);
            t4=itemView.findViewById(R.id.mobileno);
            t5=itemView.findViewById(R.id.checkin);
            t6=itemView.findViewById(R.id.checkoutdate);
            t7=itemView.findViewById(R.id.price);
        }
    }
}


package com.dd.pgadmin;

import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.webkit.MimeTypeMap;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

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

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
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 com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
```

29

```java
import com.google.firebase.storage.StorageTask;
import com.google.firebase.storage.UploadTask;



public class EditActivity extends AppCompatActivity {

    EditText e1,e2,e3,e4,e5,e6,e7,e8,e9,e10;
    RadioButton r1,r2,r3,r4;
    RadioButton rt1,rt2,rt3,rt4,rt11,rt22,rt33,rt44;
    Button b1,b2;
    RadioGroup rg1,rg2,rg3,rg4;

    private FirebaseAuth mAuth;
    FirebaseUser user;
    private static final int CHOOSE_IMAGE = 1;
    private Uri imgUrl;
    private StorageTask mUploadTask;
    public  String Imageurl;
    StorageReference storageReference;
    FirebaseStorage storage;
    int id=0;
    PgDetailsmodel pgDetailsmodel;
    String name;
    public String userid;
    String ac,food,gym,fur;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit);
        getSupportActionBar().hide();

        e1=findViewById(R.id.pgname);
        e2=findViewById(R.id.pgdescription);
        e3=findViewById(R.id.pgaddress);
        e4=findViewById(R.id.pgcontact);
        e5=findViewById(R.id.bhk);
        e6=findViewById(R.id.noofbeds);
        e7=findViewById(R.id.boysorgirls);
        e8=findViewById(R.id.montlyrent);
        e9=findViewById(R.id.deposit);

        pgDetailsmodel=new PgDetailsmodel();

        rg1=findViewById(R.id.acnonac);
        rg2=findViewById(R.id.gymnogym);
        rg3=findViewById(R.id.foodnofood);
```

30

```
rg4=findViewById(R.id.furnofur);

b1=findViewById(R.id.button);
b2=findViewById(R.id.button2);

mAuth = FirebaseAuth.getInstance();
user=mAuth.getCurrentUser();
userid=user.getUid();

storage = FirebaseStorage.getInstance();
storageReference = storage.getReference();

rt1 = findViewById(R.id.rbac);
rt11 = findViewById(R.id.rbnac);

rt2 = findViewById(R.id.rbgym);
rt22 = findViewById(R.id.rbngym);

rt3 = findViewById(R.id.rbfood);
rt33 = findViewById(R.id.rbnfood);

rt4 = findViewById(R.id.rbfur);
rt44 = findViewById(R.id.rbnfur);




final FirebaseDatabase database = FirebaseDatabase.getInstance();
final DatabaseReference ref = database.getReference("Pgowner");

ref.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

        for (DataSnapshot npsnapshot : dataSnapshot.getChildren()){
            name=dataSnapshot.child(userid).child("pgname").getValue().toString();
            //Toast.makeText(EditActivity.this,name,Toast.LENGTH_LONG).show();
        }
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
        Toast.makeText(EditActivity.this,databaseError.getMessage(),Toast.LENGTH_LONG).show();
    }
});
final FirebaseDatabase db = FirebaseDatabase.getInstance();
final DatabaseReference myref = db.getReference("Nearbypgs");
int id=0;
myref.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
```

31

```java
    public void onDataChange(DataSnapshot ds) {
        //Toast.makeText(EditActivity.this,"adasdsdf",Toast.LENGTH_LONG).show();
        for (DataSnapshot npsnapshot : ds.getChildren()){

            String pgname=ds.child(userid).child("pgname").getValue().toString();
            String description=ds.child(userid).child("description").getValue().toString();
            String address=ds.child(userid).child("address").getValue().toString();
            String bhksize=ds.child(userid).child("bhksize").getValue().toString();
            String noofbeds=ds.child(userid).child("no_of_beds").getValue().toString();
            String rent=ds.child(userid).child("rent").getValue().toString();
            String deposit=ds.child(userid).child("deposit").getValue().toString();
            String boys_girls=ds.child(userid).child("boys_girls").getValue().toString();
            String mobileno=ds.child(userid).child("contactno").getValue().toString();

            ac = ds.child(userid).child("ac_noac").getValue(String.class);
            food = ds.child(userid).child("food_nofood").getValue(String.class);
            gym = ds.child(userid).child("gym_nogym").getValue(String.class);
            fur = ds.child(userid).child("fur_nofur").getValue(String.class);

            // Toast.makeText(EditActivity.this,ac,Toast.LENGTH_LONG).show();


            e1.setText(pgname);
            e2.setText(description);
            e3.setText(address);
            e4.setText(mobileno);
            e5.setText(bhksize);
            e6.setText(noofbeds);
            e7.setText(boys_girls);
            e8.setText(rent);
            e9.setText(deposit);

        }
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
        Toast.makeText(EditActivity.this,databaseError.getMessage(),Toast.LENGTH_LONG).show();
    }
});


// RadioButtonDsiplay();

b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        chooseImage();
    }
```

32

```java
    });


    b2.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Editpg();
      }
    });
  }


  private void chooseImage() {
    Intent i1 = new Intent();
    i1.setType("image/*");
    i1.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(i1, CHOOSE_IMAGE);

  }
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == CHOOSE_IMAGE && resultCode == RESULT_OK && data != null &&
data.getData() != null) {
      imgUrl = data.getData();
    }
  }
  public String GetFileExtension(Uri uri) {

    ContentResolver contentResolver = getContentResolver();
    MimeTypeMap mimeTypeMap = MimeTypeMap.getSingleton();
    // Returning the file Extension.
    return mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri)) ;

  }


  private void Editpg() {

    if (imgUrl != null) {
      final ProgressDialog mdialog=new ProgressDialog(EditActivity.this);
      mdialog.setMessage("Please Wait");
      mdialog.show();
      //Toast.makeText(EditActivity.this, "HELLO", Toast.LENGTH_SHORT).show();
      final StorageReference ref = storageReference.child("Pgimages/" + System.currentTimeMillis() + "." +
GetFileExtension(imgUrl));
      mUploadTask = ref.putFile(imgUrl)
          .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
```

33

```java
@Override
public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
  //Toast.makeText(getApplicationContext(),"sdfdsfsdfsdf",Toast.LENGTH_LONG).show();
  ref.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
    @Override
    public void onSuccess(Uri uri) {
      Imageurl = uri.toString();

      int a = rg1.getCheckedRadioButtonId();
      r1 = (RadioButton) findViewById(a);
      //Toast.makeText(getApplicationContext(),"sdf"+a,Toast.LENGTH_SHORT).show();
      int b = rg2.getCheckedRadioButtonId();
      r2 = (RadioButton) findViewById(b);
      int c = rg3.getCheckedRadioButtonId();
      r3 = (RadioButton) findViewById(c);
      int d = rg4.getCheckedRadioButtonId();
      r4 = (RadioButton) findViewById(d);



      final String pgname = e1.getText().toString().trim();
      final String pgdescription = e2.getText().toString().trim();
      final String pgaddress = e3.getText().toString().trim();
      final String contactno = e4.getText().toString().trim();
      final String bhksize = e5.getText().toString().trim();
      final String nofbeds = e6.getText().toString().trim();
      final String boysorgirl = e7.getText().toString().trim();
      final String rent = e8.getText().toString().trim();
      final String deposit = e9.getText().toString().trim();
      final String acnoac = r1.getText().toString();
      final String gymnogym = r2.getText().toString();
      final String foodnofood = r3.getText().toString();
      final String furnofur = r4.getText().toString();

      final FirebaseDatabase database = FirebaseDatabase.getInstance();
      final DatabaseReference myref = database.getReference("Nearbypgs");
      pgDetailsmodel.setPgname(pgname);
      pgDetailsmodel.setDescription(pgdescription);
      pgDetailsmodel.setAddress(pgaddress);
      pgDetailsmodel.setContactno(contactno);
      pgDetailsmodel.setBhksize(bhksize);
      pgDetailsmodel.setNo_of_beds(nofbeds);
      pgDetailsmodel.setBoys_girls(boysorgirl);
      pgDetailsmodel.setRent(rent);
      pgDetailsmodel.setDeposit(deposit);
      pgDetailsmodel.setAc_noac(acnoac);
      pgDetailsmodel.setGym_nogym(gymnogym);
      pgDetailsmodel.setFood_nofood(foodnofood);
```

34

```
                    pgDetailsmodel.setFur_nofur(furnofur);
                    pgDetailsmodel.setPgimage(Imageurl);
                    pgDetailsmodel.setUserid(userid);

                    myref.child(userid).setValue(pgDetailsmodel);
                    mdialog.dismiss();
                    Toast.makeText(getApplicationContext(),"Updated
Successfully",Toast.LENGTH_SHORT).show();

                    }
                });
            }
        })

        .addOnFailureListener(new OnFailureListener() {
          @Override
          public void onFailure(@NonNull Exception e) {
            Toast.makeText(getApplicationContext(), "Failed " + e.getMessage(),
Toast.LENGTH_SHORT).show();
          }
        });
    }
  }
}
```
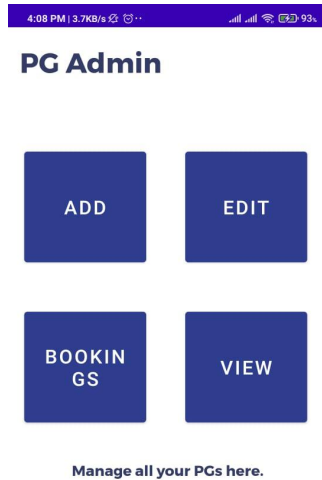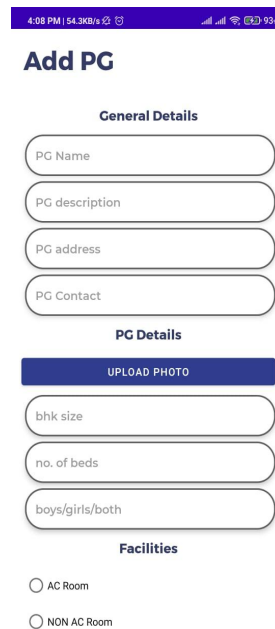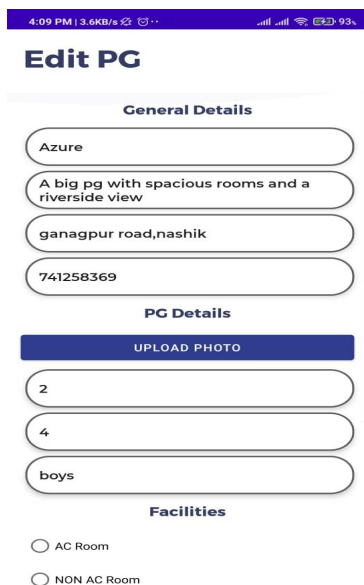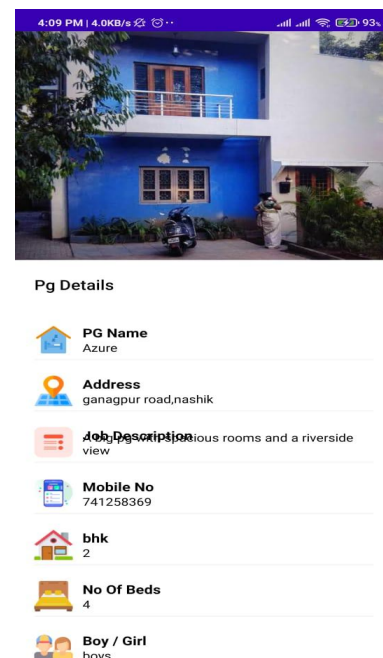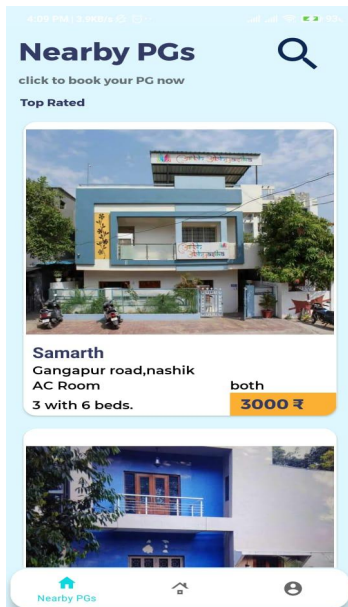
6.2 Output

6.1 Main Screen Of Admin Panel

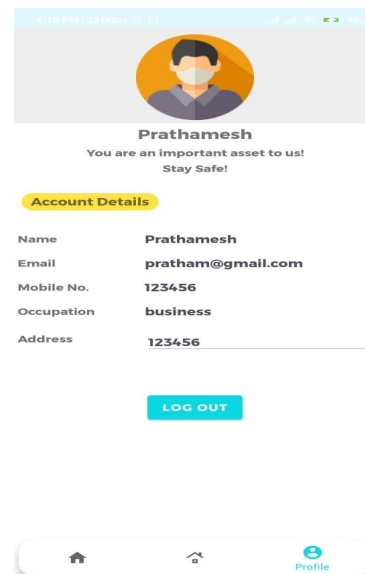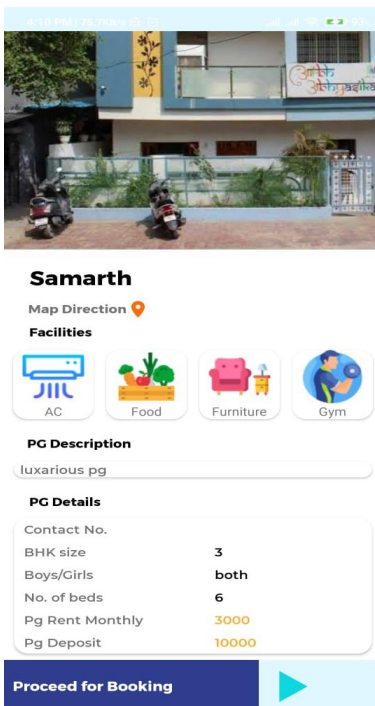6.2 Add PG Screen

6.3 Edit PG details Screen

6.4 View Pg Details

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

6.5 Home Screen
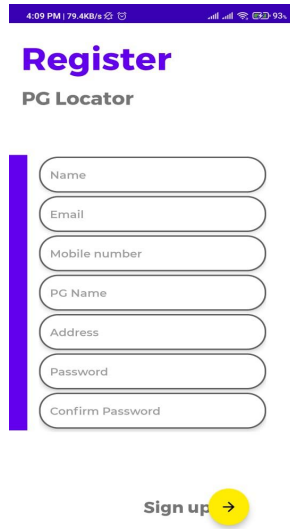
6.6 Profile

6.7 PG details Screen

6.8 View Booked Pg

37

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

6.9 Registration Screen

Chapter: - 7

Modelling and Designing

7.1 System Flow Diagram: -

Fig.7.1: System Flow Diagram

7.2 ER Diagram: -

An entity relationship model, also called an ER diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within database or information system.

7.3 Data Flow Diagram: -

A data flow diagram (DFD) is a graphical representation of "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of system, which can later be elaborated. DFD can also used for the visualization of data processing.
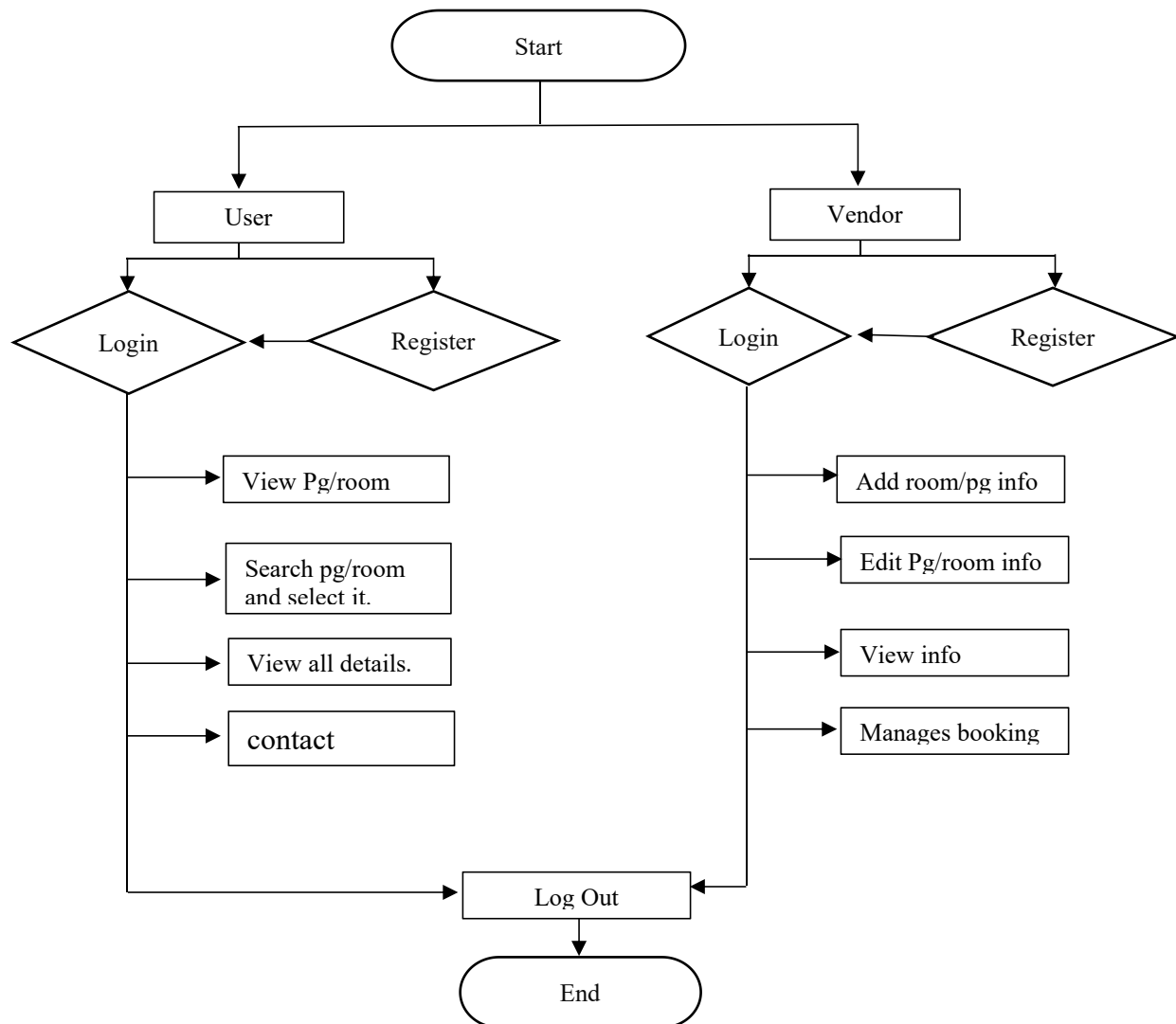
A Data Flow Diagram (DFD) is a structured analysis and design tool that can be used for flowcharting. A DFD is a network that describes the flow of data and the processes that change or transform the data throughout a system. This network is constructed by using a set of symbols that do not imply any physical implementation. It has the purpose of clarifying system requirements and identifying major transformations. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. DFD can be considered to an abstraction of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFD's are often referred to as logical data flow diagrams.

7.3.1 External Entity: -

An external entity is a source or destination of a data flow. Only those entities which originate or receive data are represented on a data flow diagram. The symbol used is a rectangular box.

7.3.2 Process: -

A process shows a transformation or manipulation of data flow within the system. The symbol used is an oval shape.

7.3.3 Data flow: -

The data flow shows the flow of information from a source to its destination. Data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Each data flow may be referenced by the processes or data stores at its head and tail, or by a description of its contents.

7.3.4 Data Store: -

A data store is a holding place for information within the system: It is represented by an open-ended narrow rectangle. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations.

➢ **Login DFD: -**

Fig.No.7.2 Login DFD
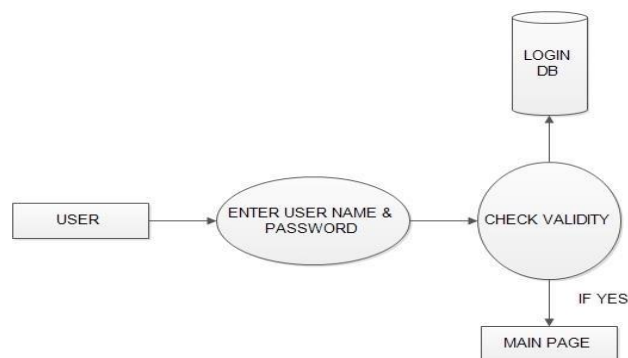
➢ **Admin DFD: -**
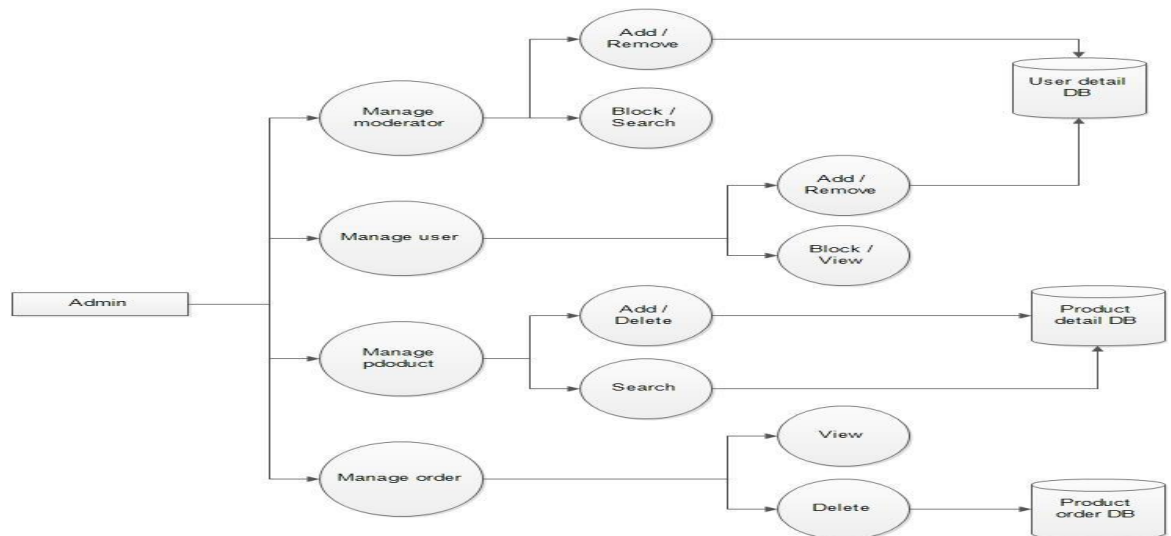
Fig.7.3 Admin DFD

7.4 Use Case diagram: -

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can   perform in collaboration with one or more external users of the system (actors). A use case diagram at its simplest is a representation of a user's   interaction with   the   system that   shows   the   relationship between   the user   and   the different   use cases   in which the user is   involved.
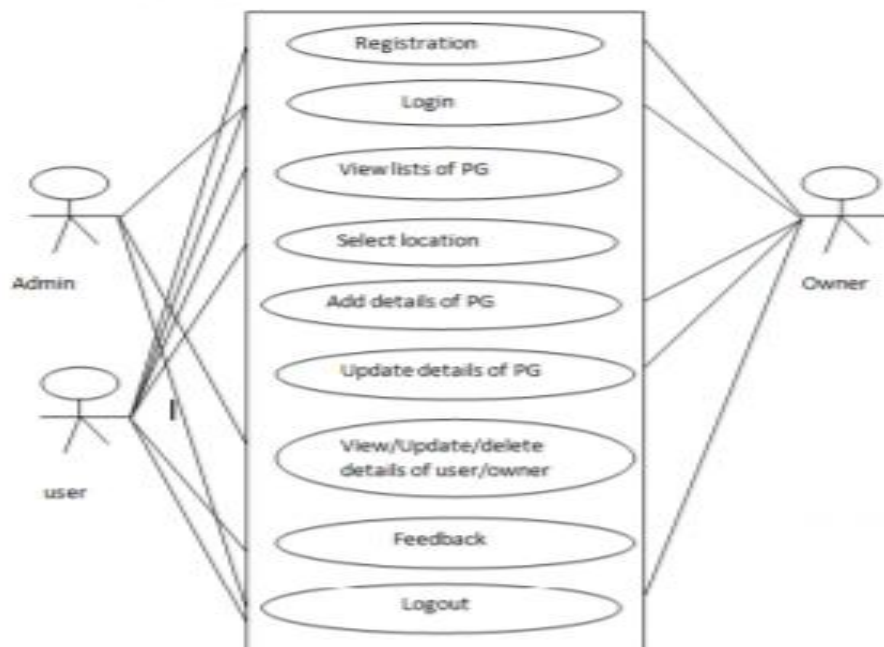
ıter Technology

Fig. 7.4 Use Case diagram

"Live inn" is a new way to help people meet their needs for living. It will benefit the community of students, working people, and therefore the larger community as a whole. "Live inn", is an android based application that makes the common people (user) to search for the Paying Guest (PG) all over the city. The PG owners can advertise the PG details by registering into the website which is based on Admin side. Once the admin verifies the details and authenticate the login, the owners will be able to login and can also able to upload the PG details into the application. The users can view those details and make use of this information to search for the PGs according to their needs. This application helps the users to view the PG details, the facilities in the PG and so on. The users can also search the PG's information based on some parameters such as colleges nearby, rent, Wi-Fi etc...This application will be useful to anyone who is in search of a place to stay. This system mainly focuses on the PG owners as well as the common users who are in search of PGs who come from different places to study as well as to work.

7.5 System Design: -

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. Its emphasis on translating design. Specifications to performance specification. System design has two phases of development.
  ➢ Logical design
  ➢ Physical design

During logical design phase the analyst describes inputs (sources), output s(destinations), databases (data sores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

7.5.1 Input and Output Design: -

7.5.1.1 Input Design: -

Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing   the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer-based format in the input design. Input data are collected and organized into groups of   similar data. Once identified, the appropriate input media are selected for processing. All

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

the input data are validated and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project the student details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

7.5.1.2 Output Design: -

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

7.5.2 Database: -

Databases are the storehouses of data used in the software systems. The data is stored in key-value pairs inside the database. Several tables are created for the manipulation of the data for the system.

7.5.3 System Tools: -

The various system tools that have been used in developing both the front end and the back end of the project are being discussed in this chapter.

7.5.3.1 Front End: -

Android studio, XML, firebase is utilized to implement the frontend and Server Side.

7.5.3.2 Back End: -

The back end is implemented using Firebase which is used to design the databases.

G] System Requirements:

1] Hardware

- ➢ Processor: Intel Core i3 or more
- ➢ RAM: 4 GB or More
- ➢ Hard disk: 512GB or more
- ➢ Monitor: 15" CRT, or LCD monitor
- ➢ Keyboard: Normal or Multimedia
- ➢ Mouse: Compatible mouse

2] Software

- ➢ Technology Used: Android

43

➢ Operation System: Windows 10

Chapter: - 8

Testing and Costing

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully, it will remove all the errors from the software.

8.1 Principles of Testing: -

1. All the test should meet the customer requirements
2. To make our software testing should be performed by third party
3. Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
4. All the test to be conducted should be planned before implementing it
5. It follows pareto rule(80/20 rule) which states that 80% of errors comes from 20% of program components.
6. Start testing with small parts and extend it to large parts.

8.2 Steps of Software Testing:

1. Requirement Analysis.
2. Planning the test.
3. Developing the test case.
4. Setting up the test environment.
5. Executing the test.
6. End of test, or closing the test cycle.

8.3 Types of testing:

8.3.1. Unit Testing:
It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs.

8.3.2. Integration Testing:

The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components are combined to produce output.

8.3.3. Regression Testing:

Every time new module is added leads to changes in program. This type of testing make sure that whole component works properly even after adding components to the complete program.

8.3.4. Validation and Verification Testing:

In software testing, verification and validation are the processes to check whether a software system meets the specifications and that it fulfills its intended purpose or not. Verification and validation is also known as V & V. It may also be referred to as software quality control.

8.3.5. White Box Testing:

White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.

8.3.6. Black Box Testing:

Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality. Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

8.3.7. Regression Testing

Regression testing is the process of testing changes to computer programs to make sure that the older programming still works with the new changes. Regression testing is a normal part of the program development process and, in larger companies, is done by code testing specialists. Test department coders develop code test scenarios and exercises that will test new units of code after they have been written. These test cases form what becomes the test bucket. Before a new version of a software product is released, the old test cases are run against the new version to make sure that all the old capabilities still work. The reason they might not work is because changing or adding new code to a program can easily introduce errors into code that is not intended to be changed. Regression testing is a necessary component to any software development lifecycle. Expert Mike Kelly explains the motivations for conducting regression tests.

8.3.8. Accessibility Testing

The aim of Accessibility Testing is to determine whether the software or application is accessible for disabled people or not. Here, disability means deaf, color blind, mentally disabled, blind, old

Test	Test Case	Test Data	Excepted Result	Actual Result	Status

age and other disabled groups. Various checks are performed such as font size for visually disabled, color and contrast for color blindness, etc.

8.3.9Ad-hoc Testing

The name itself suggests that this testing is performed on an Ad-hoc basis i.e. with no reference to the test case and also without any plan or documentation in place for such type of testing. The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality. Ad-hoc Testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case but sometimes it is possible that defects found during ad-hoc testing might not have been identified using existing test cases.

8.4 User Login: -

Test case id	Test Case	Test Data	Excepted Result	Actual Result	Status
TC1	Whether clicking on submit button without E-mail id and password it allows login or not.	Click on submit button	System does not allow user to login.	System displays message & resume to the same page.	Pass
TC2	Whether click on submit button with invalid email id and password it displays the message or not.	Email id: abc@gmail.com Password: abc1	It should display message 'Incorrect Credential's'	It displays message.	Pass
TC3	Whether by clicking on submit button with correct username and password it logins or not.	Email id: liveinn@gmail.com Password: mobcart1	System allow user to login	System allows user to access application	Pass

Fig. 8.4.1 User Login Test Cases

cases id					
TC1	Whether the system accepts email in correct format.	Email id = liveinn@gmail .com	System allows only character, number, symbol and @gmail.com	System successfully accepted email id with its validation.	Pass
TC2	Whether the system accepts name in character format only and not number.	Name = sham	System allows only character	System successfully accepted name with its validation.	Pass
TC3	When we click on Password, password is show in "…..." format.	Password = …………..	System shows the password in "……" format.	System successfully show the password in "………" format.	Pass
TC4	When we click on password field, we have to enter the character, number and special symbol.	Password = livin@123	System allows the only character, number and special symbol.	System successfully accepted password with its validation.	Pass
TC5	When we click on mobile filed, system should accept only numeric value.	Mobile Number = 7057901582	System allows only 10 digit numbers.	System successfully allows the mobile number.	Pass
TC6	When we click on Register button, system should check and store all record in database and it's allows to login the user.	Click on Register button.	System store all record in the database after checking data.	Successfully Stored all record with in the database.	Pass
TC7	When we click on Login button , system should go to the login page.	Click on login button.	System match the email id and password for login the user.	Successfully login .	Pass
Test cases	Test Case	Test Data	Excepted Result	Actual Result	Stat us

8.4 User Registration: -

Fig. 8.4.2 User Registration Test Cases

8.4 Vendor Registration:-

Test Case id	Test Case	Test Data	Excepted Result	Actual Result	Status
TC1	Whether the system accepts email in correct format.	Email id = liveinn@gmail.com	System allows only character, number, symbol and @gmail.com	System successfully accepted email id with its validation.	Pass
TC2	Whether the system accepts name in character format only and not number.	Name = rajesh	System allows only character	System successfully accepted name with its validation.	Pass
TC3	When we click on Password, password is show in "…..." format.	Password = …………..	System shows the password is "……" format.	System successfully show the password in "………" format.	Pass
TC4	When we click on password field, we have to enter the character, number and special symbol.	Password = Liv@13	System allows the only character, number and special symbol.	System successfully accepted password with its validation.	Pass
TC5	When we click on mobile filed, system should accept only numeric value.	Mobile Number = 7057901582	System allows only 10 digit numbers.	System successfully allows the mobile number.	Pass
TC6	Whether the GST number entered is in correct format.	GST No. = RNT232837TRI	System allows only character and number.	System successfully allows to GST number.	Pass
TC7	Whether we click on Register button, we have to store all record in database and it allows to login the user.	Click on Register button.	System store all record in the database.	Successfully Stored all record with in the database.	Pass
TC8	When we click on login button it should redirect to login page.	Click on login button.	System match the email id and password for login to the vendor.	Successfully login .	Pass
Test Case id	Test Case	Test Data	Excepted Result	Actual Result	Status

Fig. 8.4.3 Vendor Registration Test Cases

8.4 Vendor Login :-

Fig. 8.4.4 Vendor Login Test Cases

48

TC1	Whether clicking on submit button without E-mail id and password it allows login or not.	Click on submit button	System does not allow vendor to login.	System displays message & resume to the same page.	Pass
TC2	Whether click on submit button with invalid email id and password it displays the message or not.	Email id: abc@gmail.com	It should display message 'Invalid Credential's'	It displays message	Pass
TC3	Whether by clicking on submit button with correct username and password it logins or not.	Email id: livinn@gmail.com	System allow vendor to login	System allows Vendor to access application	Pass

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Chapter: - 9

Result And Application

9.1 Result

Result includes the information about how the operation executes and displays the result by giving various inputs. "Live inn" is a new way to help people meet their needs for living. It will benefit the community of students, working people, and therefore the larger community as a whole. "Live inn", is an android based application that makes the common people (user) to search for the Paying Guest (PG) all over the city.
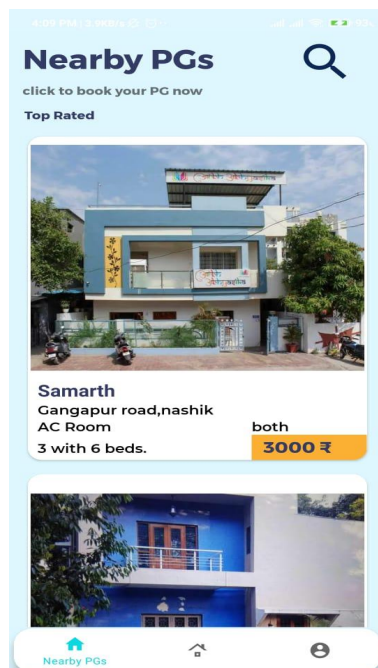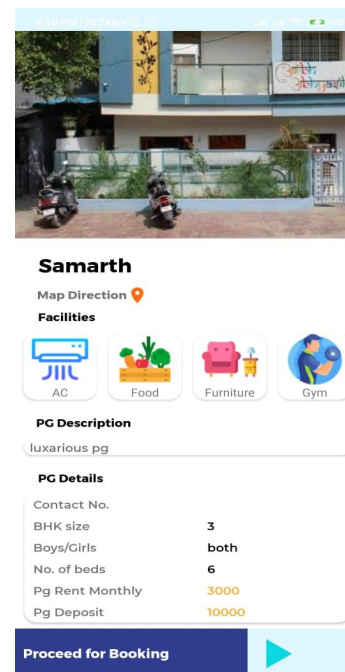

Fig 9.1 Home Screen

Fig 9.2 PG details

9.2. Costing of project:

Basic COCOMO computer software development effort as a function of program size.

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC). [10]

COCOMO applies to three classes of software projects:

1. Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
2. Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
3. Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects (hardware, software, operational,)

People required (P) = Effort Applied / Development Time [count] where, KLOC is the estimated number of delivered lines (expressed in thousands) of code for project. The coefficients a_b, b_b, c_{ob} and d_b. Are given in the following table (note: the values listed below are from the original analysis, with a modern reanalysis [4] producing different values):

Software project	A_b	B_b	c_{ob}	D_b.
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

This software comes under embedded software project. The calculation is as follows: Effort Applied (E) = $3.6(7)_{1.20}$ = 37.18

Here, a_b is 3.6 and b_b is 1.20 as mentioned in above table and KLOC i.e. lines of code are 7 which is expressed in thousand.
Development Time (D) =
$2.5(37.18)_{0.32}$ = 7.95 Here, c_{ob} is 2.5
and $_d$
$_B$ is 0.32 as mentioned in above table. People required (P) = 37.18/7.95= 4.674 According to this calculation our project requires

4   Numbers of people.

9.2.1 Working Hours:

Table: Working Hours

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Sr. No.	Work to perform	Date	Duration (In Hours)
1	Selection of Project	Daily	15
2	Flow chart preparation	Daily	15
3	Software Module Development	Daily	20
4	Module Testing	Daily	30
5	Project Report	Daily	20

9.2.2 Costing:

Table: Costing

Main Hours(From the above table)	100
Cost of one man power	Rs. 25/- per Hour
Rent of computer	Rs. 20/- per Hour
Stationary & other Expenses	Rs. 1500/-

9.2.3 Total Cost:

Table: Total Cost

Cost of total man power(100*25)	Rs. 2500/-
Total rent for computer for 9 months(100*20)	Rs. 7000/-
Stationary & other expenses	Rs. 1500/-
Total	Rs. 11,000/-

Chapter: - 10
Conclusion And Future Scope

PG Locator is a user-friendly app which enables the people to easily locate authorized PGs only. time has been an important factor in everyone's life and when people move to different places to study or to work, they definitely have to waste time in searching for accommodation. This app turns to be very time saving for us. There have also been certain cons of this app. the app has been reported to hang/lag in underdeveloped phones. According to a survey conducted by the pew research Centre, in India ONLY 17% of people are using smart phones. so, if the app is reported to hang in underdeveloped phones, then it is something that is to be taken into account and a solution must be provided.

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Chapter 11

Annexure-I

Abbreviations	Full Form
FRD	Firebase Realtime Database
API	Application programming interface
IDE	Integrated Device Electronics
XML	Extensible Markup Language
JDK	Java Development Kit
SDK	Software development kit

MVP's Rajarshi Shahu Maharaj
polytechnic, Nashik

Computer Technology

Chapter 12

References

[1]. Beginning Android Programming by Haseman.

 [2]. Android Application Development by James C. Sheusl.

[3]. Programming Android by Zigurd Mednieks, Laird Dornim, G. Blake Meike, Masumi Nakamura.

[4]. Android Programming by Bill Phillips, Brian Hardy.

[5]. the Complete Reference JAVA by Herbert Schildt.

[6]. Ian Sommerville "Software Engineering" Person Education Ltd,9th Edition

[7]. [3]MichaelBlaha,James Rumbaugh "OOMD" Person Education Ltd,2nd Edition

 [8]. [4]Elmasri and Nava the "Fundamentals of Database System" Person Education Ltd 5 h Edition

 [10]. [5]Mauro Pezze,Michal Young "Software Testing" John Willy and Sons Pte. Ltd Edition