Project Report on

HOTEL BOOKING APPLICATION (ANDROID)

Instructor: Zhiyong Shan
Course: CS780

Advanced Software Engineering

By
Group No: # 11
Sivaprasad Reddy Bogireddy – K966S589
Harshavardhan Reddy Narreddy – U848H239
Harsha Vardhan Reddy Channu – R577P973
Anjaneyulu Kolluri – M743Z369

Contents

RAM GROUP OF HOTEL'S ANDROID APPLICATION

Description

Android Application is designed for customers who would like to book a room in the "Ram Group f Hotels". It enables our customers to look through and reserve rooms, trips, and food and drink from our breakfast, lunch, and dinner menus all over the city. Additionally, it enables the customer to evaluate and rate the "Ram Group of Hotels" and their experience there, as well as produce an up-to-date invoice so they can keep track of their expenditures while visiting us. Additionally, it has administrative features that enable the hotel manager and receptionists to view, modify, and delete users, items, activities, and bookings so they can monitor the hotel's operations and make any necessary corrections to the data.

Technologies Used:

Front End: Java Software: Android Studio

Back End: Firebase Data Store: Firebase

Functionalities:

- Sign Up/ Log In Functionality
  Signup/Login Implemented with fragments. New users can register for the app in the sign-up section by entering their first and last names, email addresses, passwords, and regions. For already existing users, the log in fragment provides inputs for their credentials email address and password to access our app's main features. Both use Firebase Authentication, Animation and Custom View, as well as different kinds of verification for email and password. There are toast messages that show usage hints and any problems encountered during the Sign Up/Log In process.
- Room Booking
  This feature allows users to book rooms. On the "rooms" activity, there are 3 threads which are used to create an image-changing animation every 2 seconds, as well as descriptions to every type of rooms available (Comfort, Kings, and Family Suites), which are placed in a "Expanded View". Additional user inputs include a Date Picker for the booking's date, "EditTexts" for the booking's duration (in days), and "Numbers" for the number of guests who will be staying at the hotel. The "Book" button for each type of room activates a dynamic fragment designed to display the booking confirmation at the top of the screen. The "bookings" section of Firebase, which is organized based on the current user, contains information about reservations. There is additional proof for the.
  RULE: A user can reserve multiple rooms, but only one at a time.
- Activities Reservation
  Almost the same idea as Room Booking page but separated into two different activities ("Activities" and "Booking"). According to the "Room Booking," there are images corresponding to each activity's content that are updated frequently through threads. A description and a "Book" button are also present. The Booking Page, where the Quantity of People and Date Picker inputs are located, is reached after the user clicks the button. After making the reservation, the user is returned to the "Activities" activity, where a

confirmation section is displayed at the top of the screen. In Firebase, the "activities" object, which is organized based on the current user, contains information about reservations. Additionally, the input date is validated.
RULE: A user can book multiple activities, but only one at a time.

- Room Service ordering
  Developed in a similar way as the “Activities” activity, but for ordering food services. We have three sections on our menu: breakfast, lunch, and dinner, and dessert. The "room service booking" activity is where the user is directed after choosing one of our categories. Here, a dynamic table is used to make it simple for users to order products from that category. This table's content is generated "on the fly," and it varies based on the category the user selected during the previous activity. Additionally, information regarding each food item ordered is stored in Firebase under the "service" object, which is organized according to the current user.
  RULE: A user can order multiple meals from one menu (e.g. Breakfast) simultaneously. If the user makes more than one order of the same meal at different times, the Invoice will show each record. This is done to differentiate separate orders.
- Rank and Review
  Users can write reviews of our hotel services using the "Rank and Review" activity. A rating bar, an animation based on a "CANVAS" object used as a "emoji," a "EditText" to retrieve the content of the user's review, and a "emoji"-based rating bar are all included in this activity. As the rate rises, the emoji's "mood" shifts from depressed to upbeat. The "review" object in Firebase, which is organized based on the current user, stores the review along with the value of the rating bar. A "Toast" is displayed as confirmation after the data is stored..
  RULE: A user can submit only one review. The database will save the most recent one, overwriting previous entries. This way we will make sure the review is fair from every user.
- Invoice
  The Invoice Page shows all of the user's bookings and purchases along with the prices for each item and the most recent total. The Order id, date, service (the name of what was bought or reserved), and corresponding price of each transaction made by the current user are displayed in a "ListView" with an "Adapter" that was implemented. Based on the current user, our Firebase database's "bookings," "activities," and "orders" objects are used to read the information used to display this table. Data is handled and put in the correct fields. To fit the "ListView," all information has been correctly formatted. Database event listeners ensure that all information is current..
- Admin Page
  The Admin Page was designed to allow the hotel manager(s) to review the hotel’s records of customers, bookings, products and activities, as well as sales (by viewing which activities have been booked or which items have been sold). It also enables the manager(s) to make changes to user’s information if needed. This activity also provides tools for updating a user’s first name, last name, and the region. Additionally, it enables the manager(s) to delete users, along with his/her entire history with the hotel. Since the

methods for reading and manipulating data are placed inside database event listeners, the data displayed changes automatically with no need to execute queries again.

Features

• Animation

To make the sign-up and log-in processes more enjoyable and user-friendly, sliding and shaking animation was added. Additionally, "Custom Views" animations that were created from scratch were added to the "Rank and Review" activity to enhance the visuals.

• TextView Customized (for passwords)

Implemented in the "Log In" fragment to give users the option to reveal their password so they can double-check their entry. Action Up and Action Down events trigger a response from it.

• Fragments

The "Sign Up" and "Log In" ones are designed as fragments that are used as fully functional activities in our app (they handle actual code and interaction with other pages). Additionally, there are fragments on the app's home screen that are utilized as a dynamic component of that activity.

• Threads

An image-changing animation has been added to the "Room," "Activities," and "Room Service" activities to improve the app's visuals.

• The "Activities" activity now includes notifications that inform users of successful bookings and give them a summary of those bookings. This was put into practice using database listeners.

- ExpandView

  By clicking on the relevant elements on the page, ExpandView enables you to reveal and conceal "large" amounts of text. We did that by selecting images of arrows that were facing up and facing down for this project. The text expands and the "down" arrow changes to an "up" arrow when a user clicks on it. The majority of that text is hidden when you click the "up" arrow, and the "up" arrow changes to a "down" arrow. extensively utilized throughout the app for all descriptions that have a substantial amount of text.

- Dynamic Fragment

  The app displays booking confirmation messages to the user using dynamic fragments. A dynamic fragment appears at the top of the screen to confirm and let the user know that their "transaction" has been successfully completed when a booking is made or an order is fulfilled and if it is successfully stored into our database. The fragment itself is a red rectangle that spans the entire screen and is tall enough to accommodate confirmation text at the page's top. The fragment automatically closes after two seconds. The app made extensive use of this to inform users of all successful bookings and/or orders.

- Dynamic Table
  In the "Room Service Booking," a dynamic table is employed to display all the items associated with the selected menu. The main feature and challenge involved making sure the appropriate options were displayed in accordance with the selected menu and making sure the controllers in that table were independent and functional. The user can select the desired quantity of each food item by clicking the "+" and "-" buttons next to the corresponding food item, and this was successfully implemented. Once the customer confirms their order, the data from this table is stored in our database.
- Rating Bar
  This was put into practice in the "Rank and Review" exercise. Connecting the Rating Bar to the CANVAS emoji object, which causes the emoji's expression to change as the user increases or decreases the review's rating, was the main feature and challenge. As the rating changes, the emoji's expression shifts from sad to happy. To accomplish this, the emoji's face was redrawn each time the rating bar's score changed.
- ListView with adapter
  developed to show a user's invoice while retrieving from the database the history of their recent transactions. This information is organized and added to an object called a List, which was added to a List of the type Invoice Handler (a class that provides methods to populate the layout for Invoice using the Adapter class).

Access to admin profile within the app
In order to see the Admin Page, the following profile should be used:

Email: ramgroupofhotels@gmail.com

Password: Siva2777@

Workflows:

While developing an any application, there are majorly 5 steps/workflows, and they are

- Requirements workflow
  - This is the first step while starting any Software product Development
  - It will define what are the client's needs and objectives.
- Analysis workflow
  - This workflow will do the analysis of the client requirements.
  - The detailed analysis will increase the more fulfilments of client's requirements.
- Design workflow
  – Design workflow will refine the artifacts of the analysis workflow by iterating process till the material is in the form that can be implemented by the program developers.
- Implementation workflow
  – This workflow implements the actual requirements in the desired programming languages.
- Test workflow
  - Testing workflow will do the code check
  - There are to ways to test a code, one is execution based and other is non-execution based
  - Execution based testing will running the test cases one by one.
  - Non-execution-based testing will done by reading the artifact carefully.

1. Requirement workflow:
   - A Requirements workflow is a state transition model in which you define the state that artifacts of a specified type can be in and the actions that users can select to move these artifacts from one state to another. By creating a filter, you can locate topics that are in a particular state.
   - A use case models an interaction between the software product itself and the users of that software product (actors).
   - An actor is a member of the world outside the software product
   - It is usually easy to identify an actor
     – An actor is frequently a user of the software product
   - In general, an actor plays a role about the software product. This role is
     – As a user; or – As an initiator; or – As someone who plays a critical part in the use case.
   - A user of the system can play more than one role
   - one actor can be a participant in multiple use cases
   - An actor need not be a human being
   - The initial requirements are based on the initial business model

- The requirements are dynamic — there are frequent changes – Maintain a list of likely requirements, together with use cases of requirements approved by the client

Use case Diagram for Hotel Booking System.

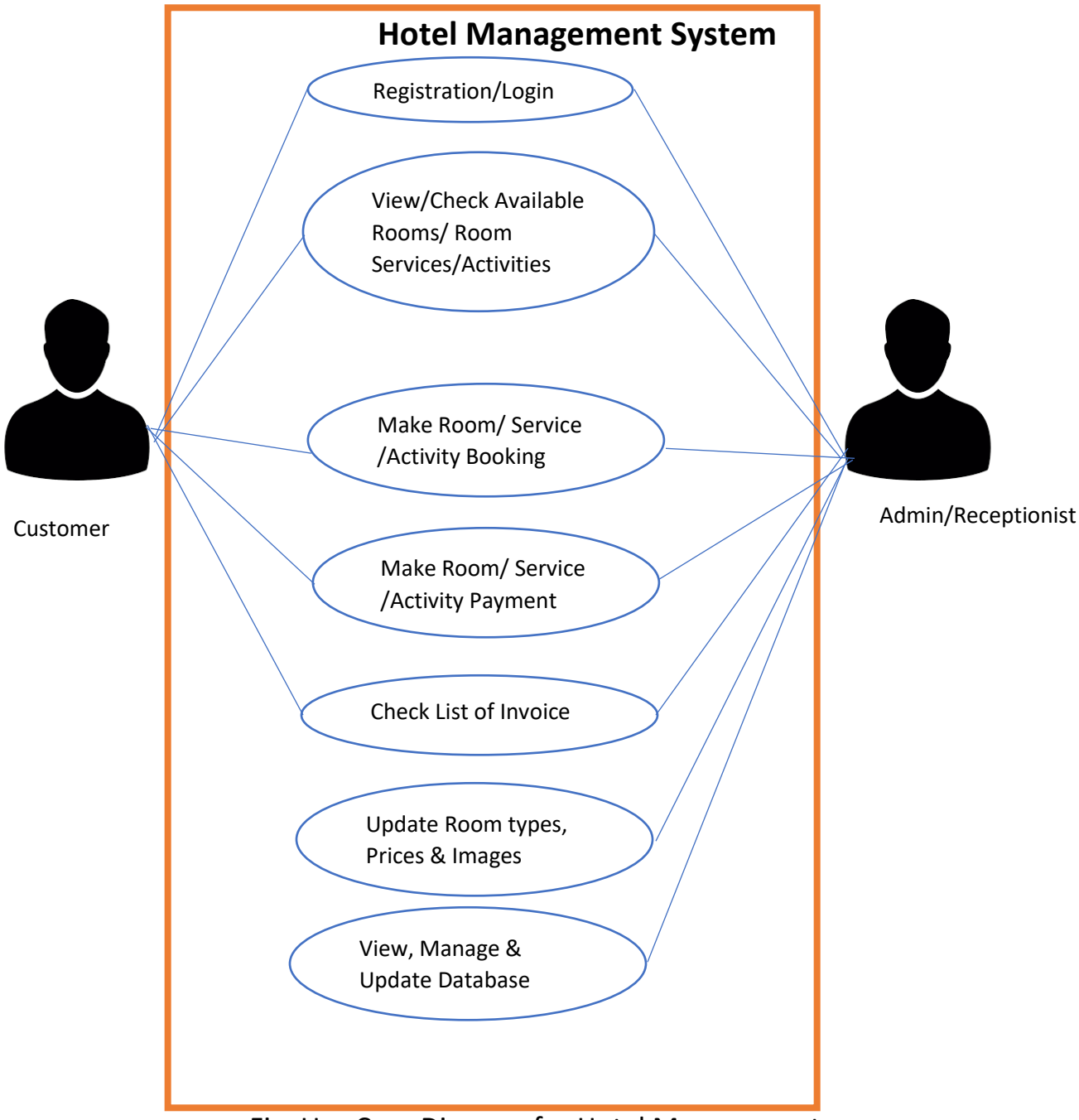


Fig: Use Case Diagram for Hotel Management

Use Case Authentication:

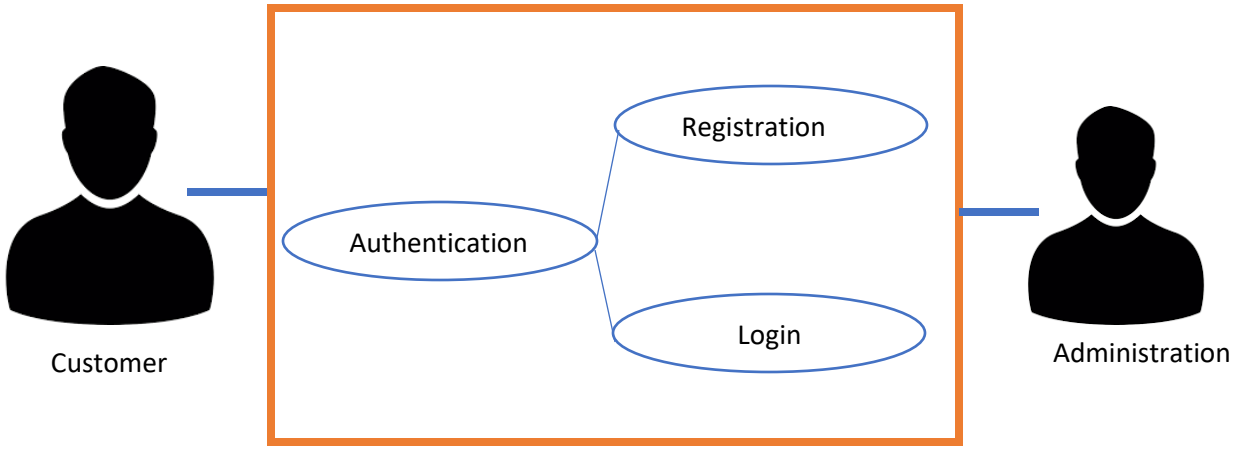


Fig: Authentication Use Case

Description: The Authentication Use Case Enables New Customers to Register and Login the Existing Customers. The New Customer will be able to Register (Signup) by entering his/her details in the Customer Interface on the Android Application. It will check the email entered by the customer with database, if no results found Customer Registration Successful. Once Customer Registered, he/she can be able to login by entering login credentials in the Login Customer Interface. The credentials will check with database and login successfully if correct.

Use Case View/Check Available Rooms/Activities/Services:

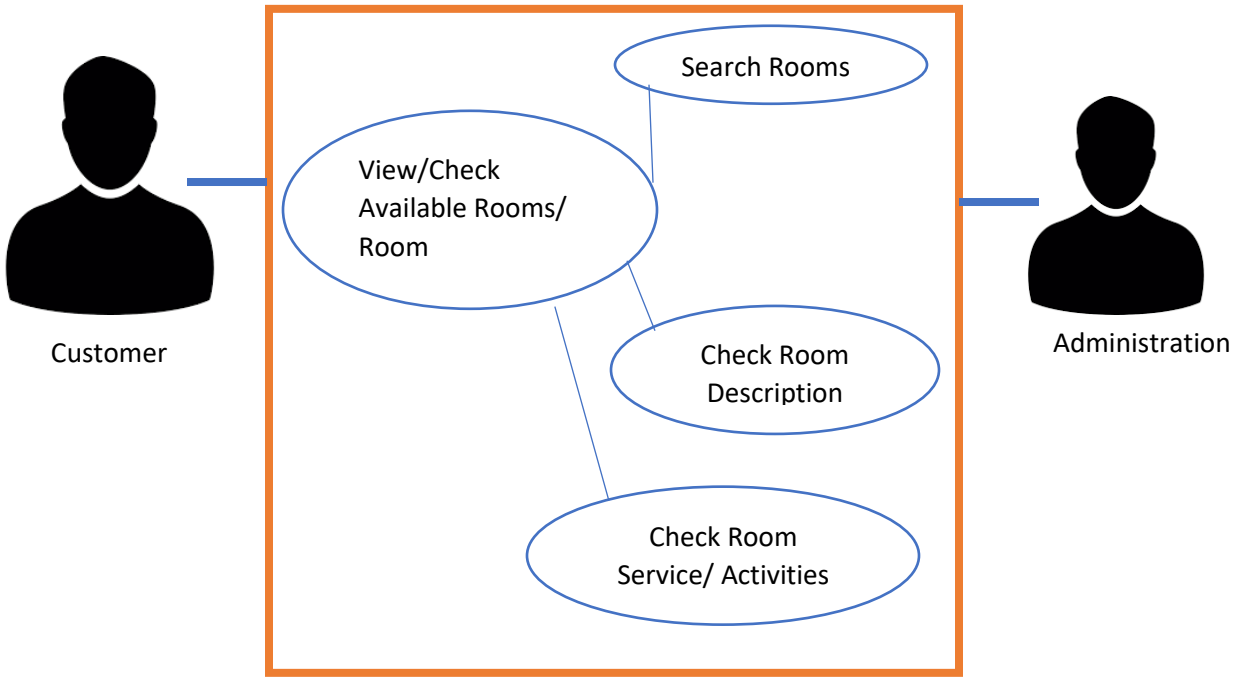


Fig: View/Check Room/Activity/Service Use Case

Description: Once Customer able to login, Customer can able to Search various available rooms with its description/Features and price...Etc. The list of rooms and available quantity can be loaded from the database. Customers can also check various Room Services like Food, Swimming, and Extra bed...Etc. and book simultaneously. Customer can also check various Activities like City Tour, Site visiting…Etc. and book simultaneously.

Use Case Make Room/Activity/Room Service Booking:

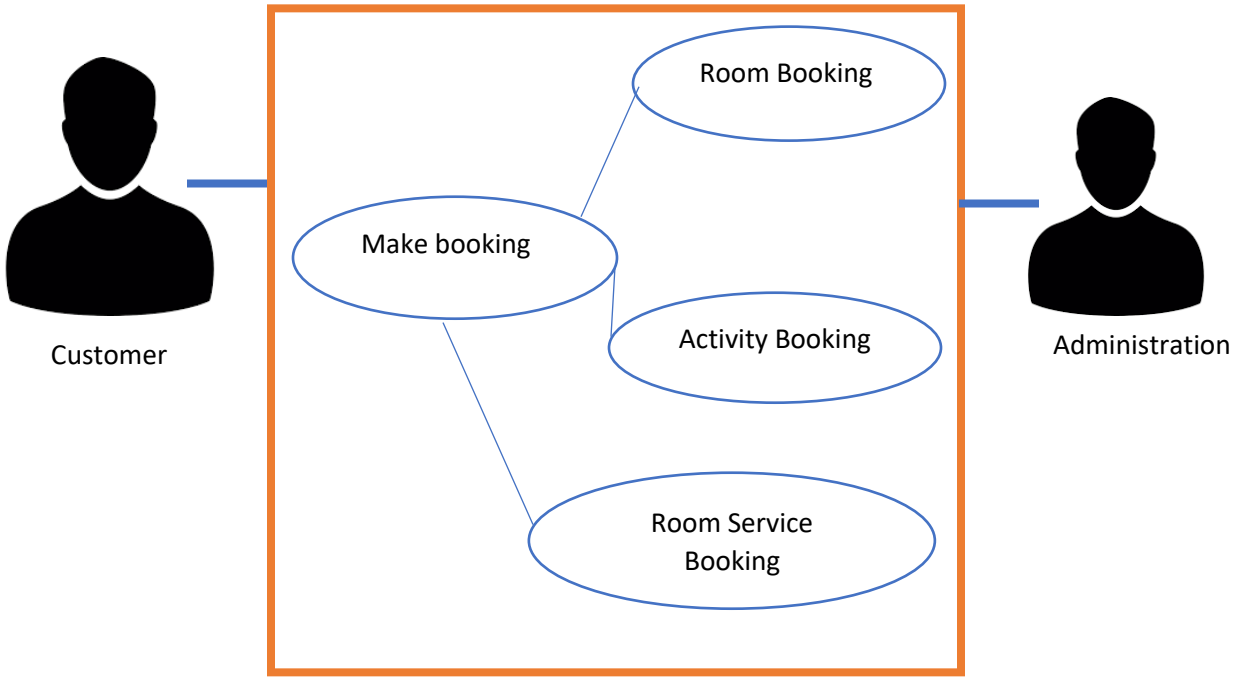


Fig: Make Room/Activity/Service Booking Use Case

Description: Once the Customer decides which item (room/activity/service) want to order, depending on availability it allow the Customer to book item by entering Number of Days, number of Guests and Start Date (for Rooms & Activity) or by selecting the available food items to book. Final price will be based on the item(s) added by the customer.

Use Case Make Room/Activity/Room Service Payment:

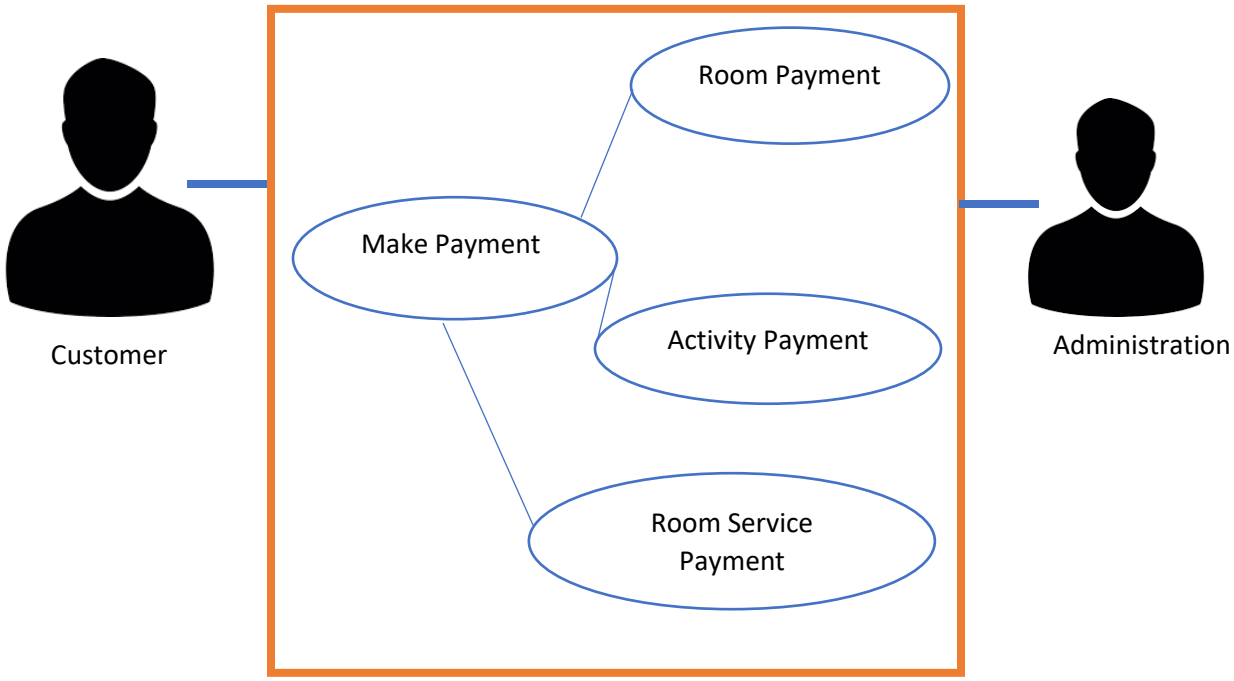


Fig: Make Room/Activity/Service Payment Use

Description: After Selecting & Entering all the requirement fields by customer, it will be redirected to the payment page. Customer need to pay the final amount shown on booking page with banking partner. Once the Payment is Successfully Completed, Customer will successfully booked order and it will be shown in the invoice of the Customer Application.

Use Case List of Invoices:

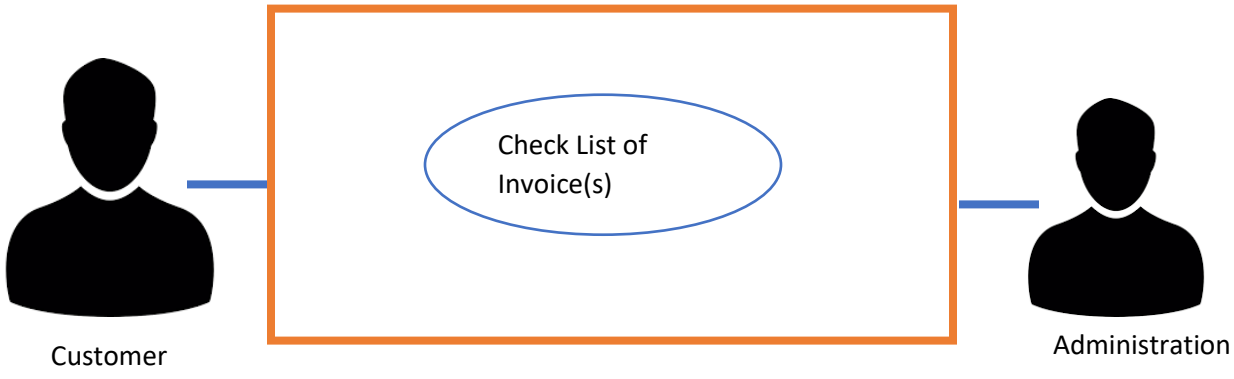


Fig: List of Invoice Use Case

Description: Once the Customer successfully make payment and booking successful, then an invoice will be generated and it will be shown in the List of Invoices Page. Customer can check all the bookings made through the android app like date wise, item wise, item description & price. Customer can check both past & future bookings as well. Each booking will be identified by unique Order Id generated at the time of Successful booking.

Use Case Update Room types, Prices & Images:

Fig: Update Room types, Prices & Images Use case

Description: Admin only has the option to update various information of rooms like Room Type, Room price, Room Images, Activities, room Services...Etc. All reflections will be done in the database and the updated content will be visible in customer UI. Admin can add and Delete new Rooms, Activities & Room Services at any time.

Use Case View, Manage & Update Database:

Fig: Update View, Manage & Update Database Use

Description: Admin can have the access to view the database containing various options like view List of Customers, Room Bookings, Activities, Room Services & find an Customer with email address, Update Customer info and Delete a particular Customer. If admin want to check-in any customer, Receptionist can check with his Order Id or Customer email address for authentication.

2. Analysis Workflow:

- A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.
- In a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class. Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

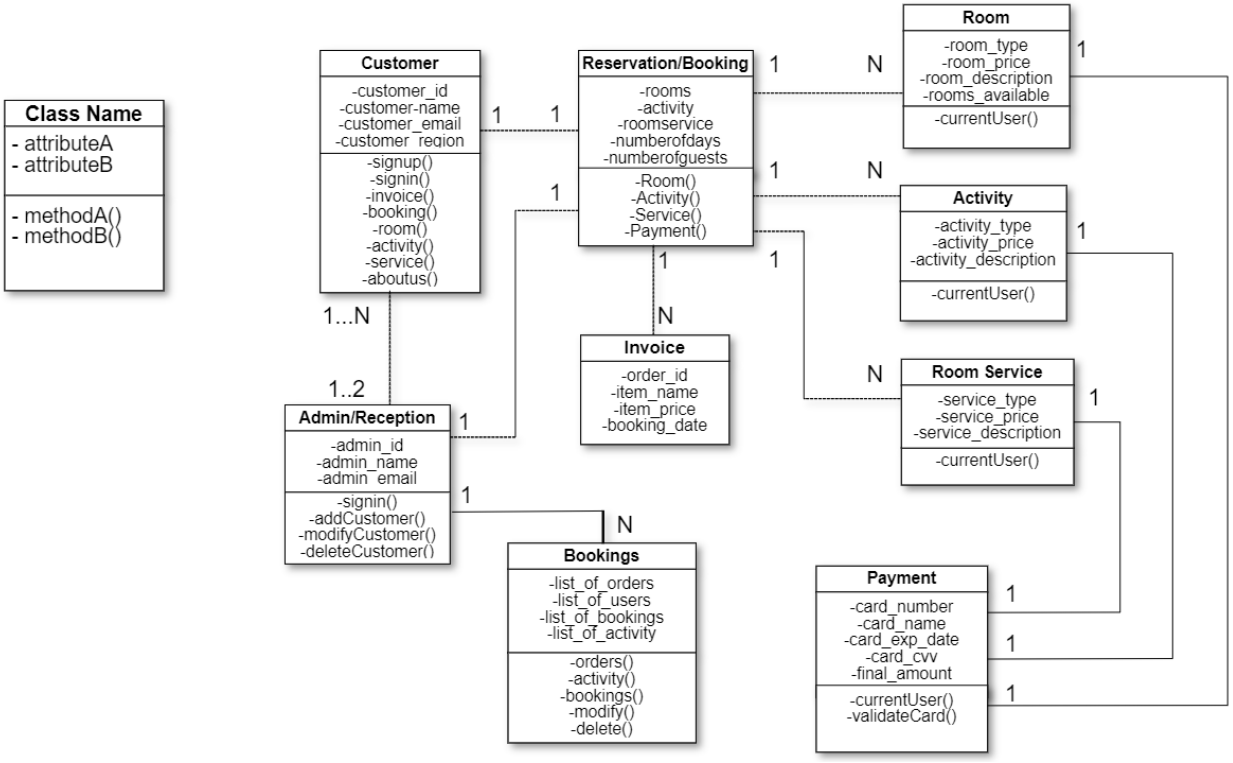


Fig: Class Diagram for Hotel Management System

CRC cards:

- CRC-Class Responsibility Collaboration
- CRC = Name of Class +Functionality (Responsibility) + List of classes it invokes (Collaboration)

  CRC card for Customer Class
  - ❖ Class Name: Customer

- Functionality:
    - Enter inputs (Fname, Lname, Emal..etc) in Signup page to signup.
    - Enter input (Email, Password) in Login page to Login Successfully
    - Check List of Available Rooms
    - Select Room & Book simultaneously
    - Make payment for booking
    - Check List of activities available & book
    - Check List of Room Services available & book
    - Show List of Orders in Invoice class
    -
- Collaboration
    - signUp
    - logIn
    - room
    - activity
    - roomService
    - invoice
    - rankReview
    - payment

CRC card for Admin class

- Class Name: Admin
- Functionality:
    - Enter input (Email, Password) in Login page to Login Successfully
    - Check List of Available Rooms
    - Select Room & Book simultaneously
    - Make payment for booking
    - Check List of activities available & book
    - Check List of Room Services available & book
    - Show List of Orders in Invoice class
    - Manage list of Bookings
    - View Customer details
    - Modify Customer Details
    - Delete Customer Details
- Collaboration
    - customer
    - logIn
    - room
    - activity
    - roomService
    - invoice
    - payment
    - bookings

CRC card for Room Class

- ❖ Class Name: Room
- ❖ Functionality:
    - Display List of Available Rooms
    - View description to Room
    - Allow customers to select Desired Room
    - Enter Booking details
- ❖ Collaboration
    - price
    - booking
    - invoice
    - payment
    - customer

CRC card for Activity Class

- ❖ Class Name: Activity
- ❖ Functionality:
    - Display List of Available Activities
    - View description to Activity
    - Allow customers to select Desired Activity
    - Enter Booking details
- ❖ Collaboration
    - price
    - booking
    - invoice
    - payment
    - customer

CRC card for Room Service Class

- ❖ Class Name: roomService
- ❖ Functionality:
    - Display List of Available RoomService
    - View description to RoomService
    - Allow customers to select Desired RoomService
    - Enter Booking details
- ❖ Collaboration
    - price
    - booking
    - invoice
    - payment
    - customer

CRC card for Payment Class

- ❖ Class Name: Payment
- ❖ Functionality:

- Allow Customers to make payment for room or activity or service

❖ Collaboration
  - room
  - activity
  - roomService
  - invoice
  - customer

CRC card for Invoice Class

❖ Class Name: invoice

❖ Functionality:
  - Show list of room, activity and service bookings in application screen

❖ Collaboration
  - price
  - booking
  - invoice
  - payment
  - customer

CRC card for Bookings Class

❖ Class Name: bookings

❖ Functionality:
  - Show list of users
  - Show list of room bookings
  - show list of activity bookings
  - show list of service bookings
  - Display customer details
  - Modify customer
  - Delete customer

❖ Collaboration
  - customer
  - invoice
  - admin

Sequence Diagram for Hotel Management System:

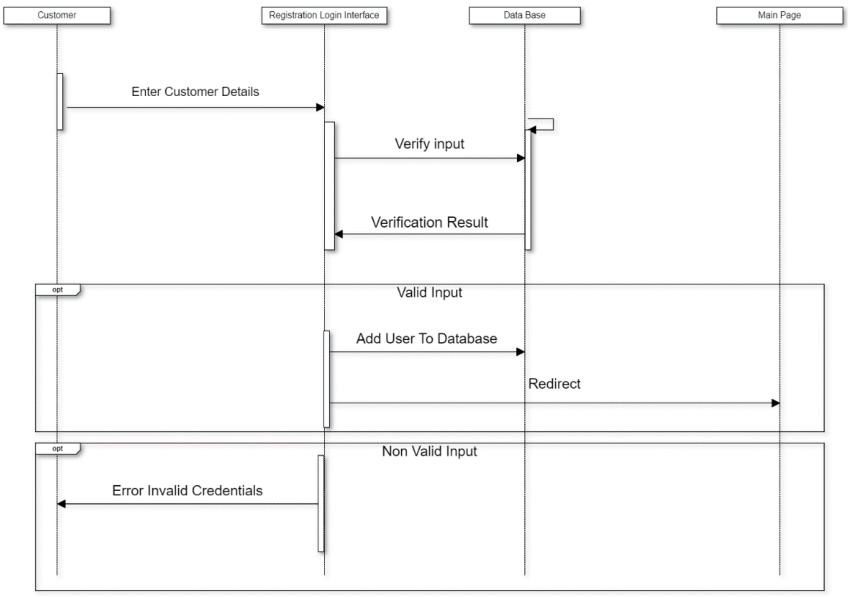


Fig: Sequence Diagram for Customer/Admin Login

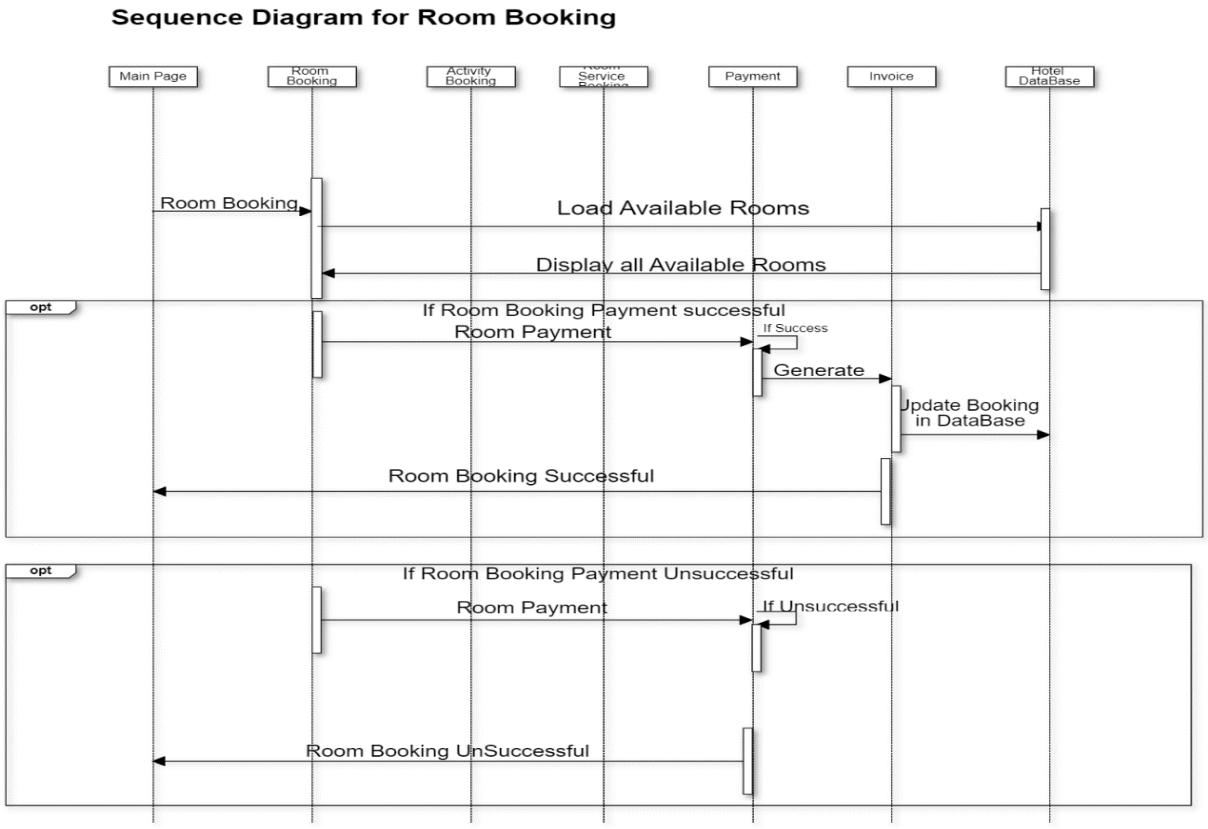


Fig: Sequence Diagram for Customer Room Booking

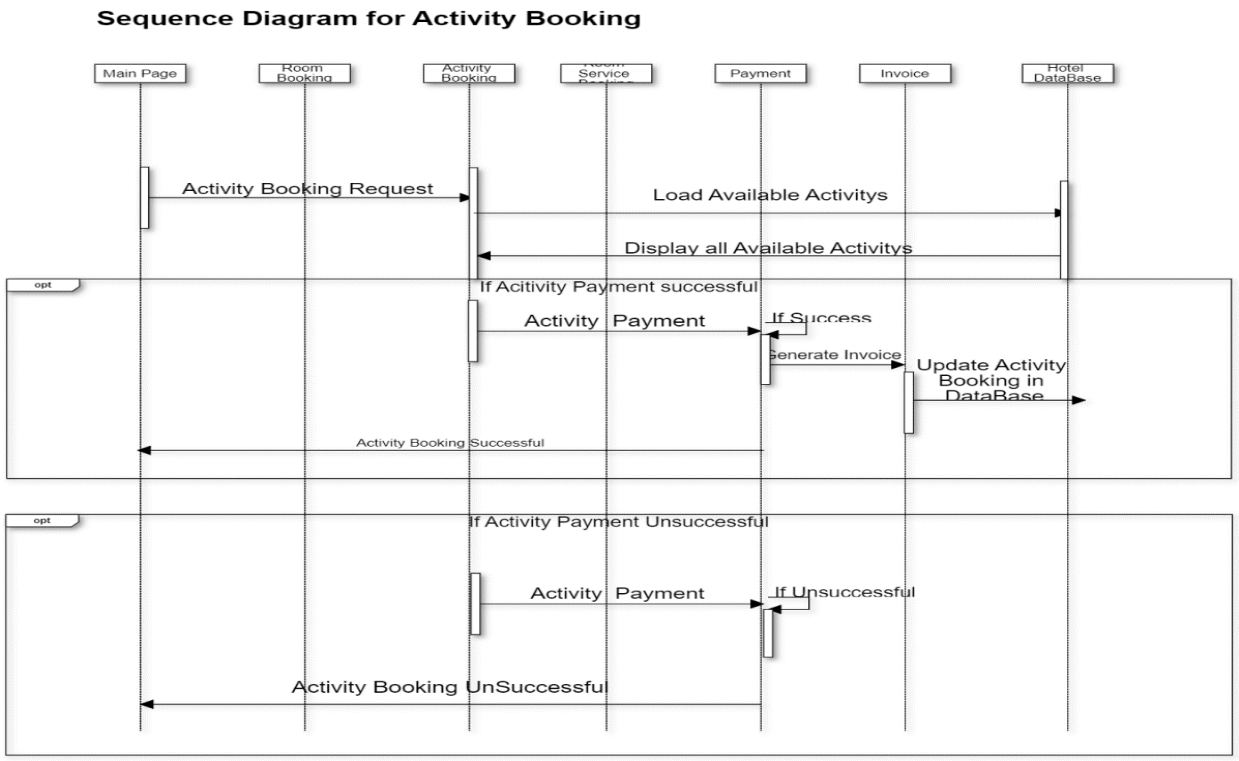


Fig: Sequence Diagram for Customer Activity Booking.

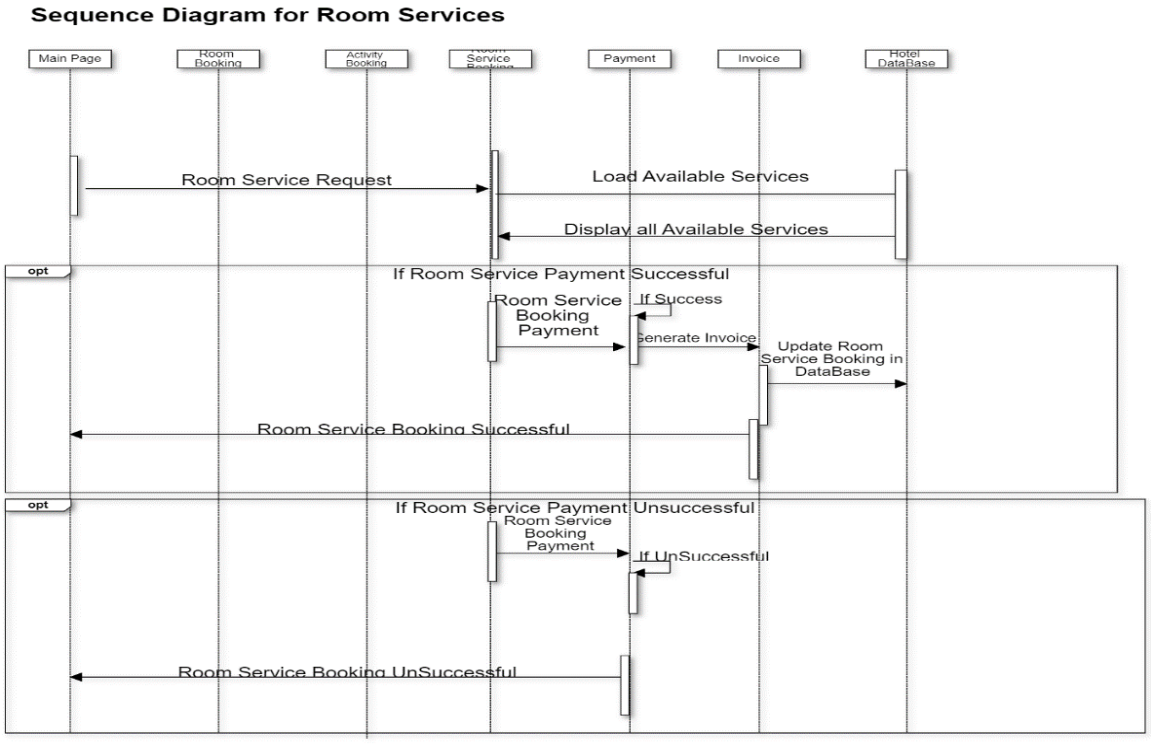


Fig: Sequence Diagram for Customer Room Service Booking

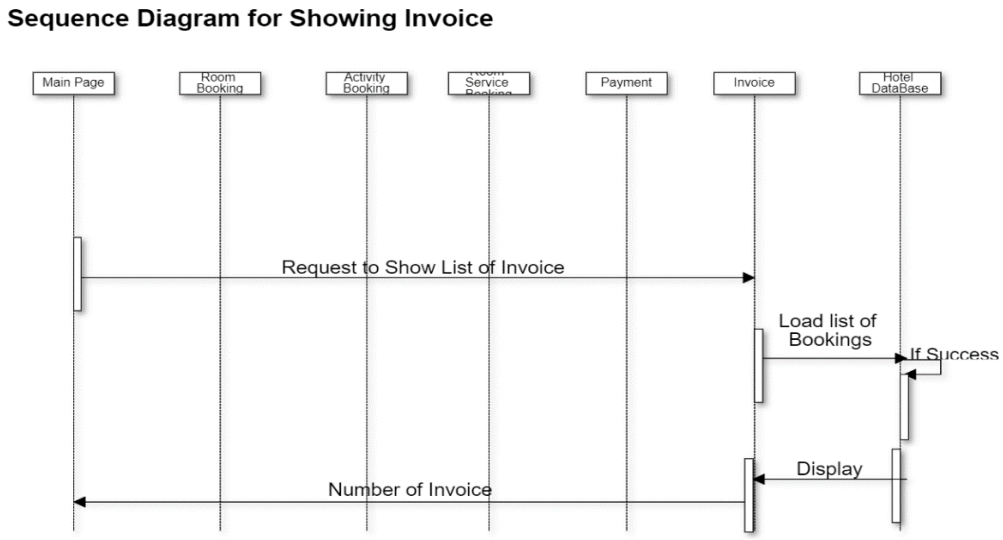
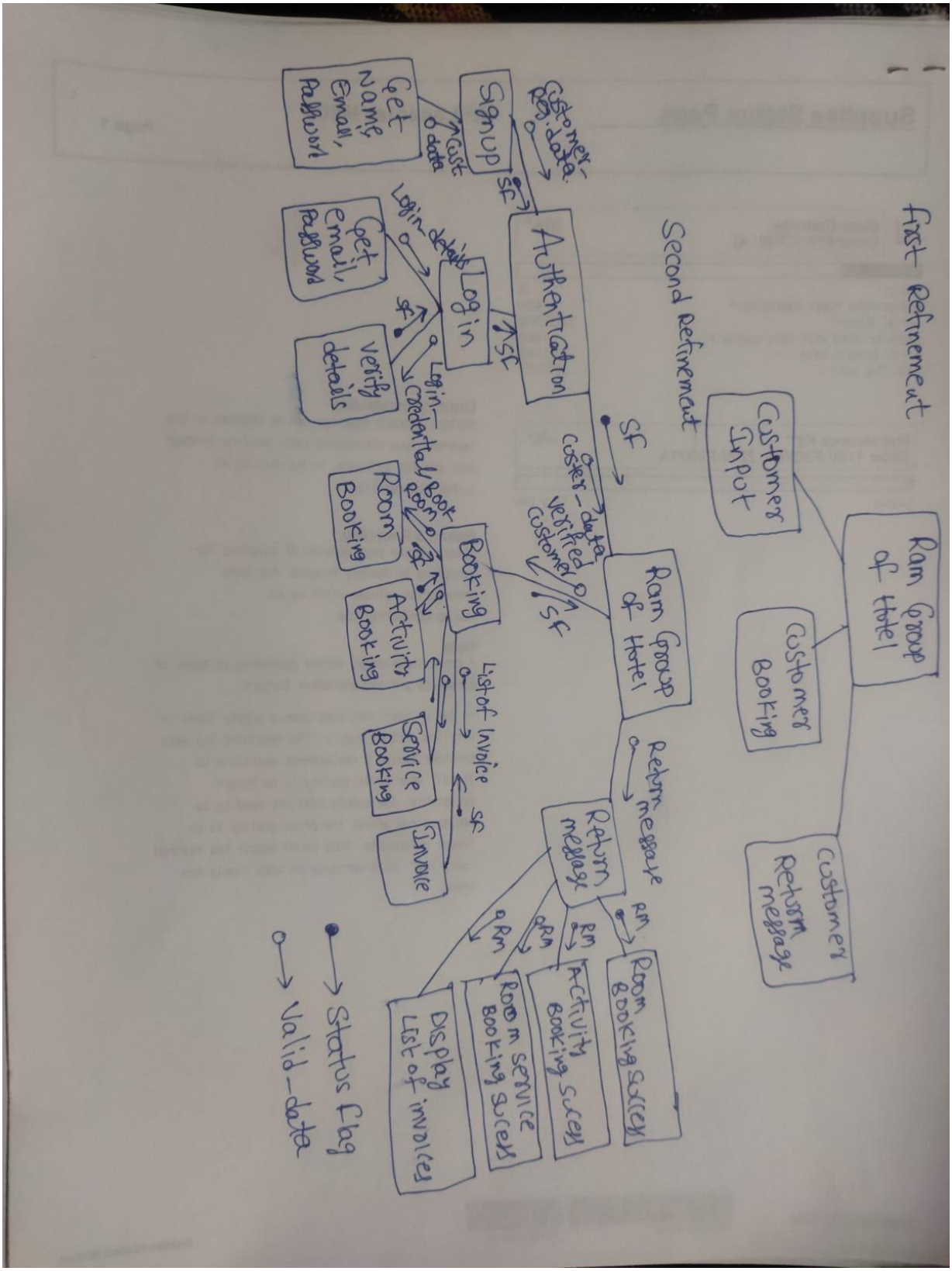


Fig: Sequence Diagram for Customer Invoice List

3. Design Workflow:

- Refine the artifacts of the analysis workflow until the material is in a form that can be implemented by the programmers.
- There are two types of designing a product.
  - Operation-oriented design
  - Data-oriented design
- Here I am approaching with Operation-Oriented Design using Data Flow Analysis
- Data flow analysis is a classical design technique for achieving modules with high cohesion. Use it with most specification methods (Structured Systems Analysis here)

Module Diagram:



Module Diagram Description:

Authentication: The Authentication Module will allow the new customers to signup by entering the First name, Last Name, Email Address, password for login. Once the Customer signed up, then he will be a valid user to the hotel application. This module also allows the existing customers (after Sign Up) to login into the Ram Group of Hotel Application. Once the Customer logged in, customer can utilize all services.

Booking: The booking module will allow the valid customers to view/search the list of available services like rooms, Activities, Room Services...etc. Simultaneously customer can book Rooms, Activities, and Room services by entering the number of days they want and which date they want to book the service. Then after Successful payment, service will be booked. Customer can request to view the list of bookings in invoice page. Once the customer successfully booked the room, the same will be updated in the database. The Admin/Receptionist has the right to view the list of bookings and list of customers.

Return Message: This is the Output module. The Return Message Module will display the successful message of each transaction (operation) like Room Booking Successful, Activity Booking Successful, and Room Service booking Successful and display the list of bookings (invoices). This module also includes displaying the list of various bookings and users in the admin accounts.

Pseudocode:

Pseudo Code for Signup & Login:

```
Class signup{          //Signup for New Customers
Void Getdetails();     //Get Details of new Customer
Signup()
        {
        if(validatedetails true)
                createAccount();
        }              //Signup the new user and create account
}

Class login{           //Login for the Existing Customers
Void getdetails();     //Get login email, password in Login interface
Void authentication()  //Authenticate the customer using login credentials
Login();{              //login on Successful authentication
        if(validation true){
                gotohomepage();
        }
}
```

Pseudo Code for Room Booking (for valid logged in users):

```
Class signup()         //Signup for New Customers

Class login()          //Login for the Existing Customers

Class home{            //Initial Login Page after Logged In
```

```
Void menu()             //View the list of available options on menu
Room();                 //Room Booking Constructor
Activity();             //Activity Booking Constructor
Roomservice();          //Room Service Booking Constructor
Invoice();              //Invoice constructor for showing list of exams
Rankreview();           //Rankreview constructor for giving rank on bookings
}

Class room{             //Customer Room Booking Class
Void selectroom();      //View list of available rooms and select one
{
       Viewrooms();
       Viewroomdesrciption(); //View the Room Description
       Selectroom();
}
Void gettenantdetails(); //Get the tenant details for room booking
{
 int number_of_guests;
 int number_of_days;
 date booking_date;
Validatedetails();
}
Payment();              //make the payment for room based on final price
{
if(paymentsuccess true)
bookingconfirmation();
updatehoteldatabase();
}
}
```

Pseudo Code for Activity Booking (for valid logged in customers):

```
Class signup()          //Signup for New Customers
Class login()           //Login for the Existing Customers

Class home{             //Initial Login Page after Logged In
Void menu()             //View the list of available options on menu
Room();                 //Room Booking Constructor
Activity();             //Activity Booking Constructor
Roomservice();          //Room Service Booking Constructor
Invoice();              //Invoice constructor for showing list of exams
Rankreview();           //Rankreview constructor for giving rank on bookings
}

Class activity{
Void selectactivity() //view list of activities available
{
       Viewactivity();
       Viewactivitydesrciption();     //View the Activity Description
       Selectactivity();
}

Void gettenantdetails(){
 int number_of_guests;
 date booking_date;
Validatedetails();
}
}
```

```
Payment()
{
if(paymentsuccess true)
bookingconfirmation();
updatehoteldatabase();
}
```

Pseudo Code Room Service Booking:

```
Class signup()          //Signup for New Customers
Class login()           //Login for the Existing Customers

Class home{             //Initial Login Page after Logged In
Void menu()             //View the list of available options on menu
Room();                 //Room Booking Constructor
Activity();             //Activity Booking Constructor
Roomservice();          //Room Service Booking Constructor
Invoice();              //Invoice constructor for showing list of exams
Rankreview();           //Rankreview constructor for giving rank on bookings
}

Class roomservice{
Void selectservice()
{
        Viewroomserices();
        Selectactivity();
        Servicequantity();     //Select the quantity of the service
        {
        int item_quantity;
        }

}
Payment(){
if(paymentsuccess true)
bookingconfirmation();
updatehoteldatabase();
}
}
```

Pseudo Code for viewing Invoice:

```
Class signup()          //Signup for New Customers
Class login()           //Login for the Existing Customers

Class invoice{
Void customerdetails();
Void listofbookings();
Void finalprice();
Void viewinvoice();
}
```

4. Source Code:

Login Class

```
package com.example.surface.ramhotelapp;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;

import androidx.annotation.NonNull;

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


public class LogInFragment extends Fragment implements View.OnClickListener {

    private static final String TAG = "LoginActivity";
    private FirebaseAuth auth;

    private static View view;


    //private static EditText username;

    private static EditText password, emailText;
    private static Button btnLog, btnSignInLog;
    private static LinearLayout loginLayout;
    private static Animation shake;
    private static FragmentManager fragmentManager;

   public LogInFragment() {

   }

   @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstance) {
```

```
        view = inflater.inflate(R.layout.fragment_log_in, container, false);
        initViews();
        setListeners();

        auth = FirebaseAuth.getInstance();

        return view;
    }

    private void initViews() {
         fragmentManager = getFragmentManager();
         //username = (EditText) view.findViewById(R.id.etUsername);
         emailText = (EditText) view.findViewById(R.id.etEmailLogIn);
         password = (EditText) view.findViewById(R.id.etPassword);
         btnLog = (Button) view.findViewById(R.id.btnLogIn);
         btnSignInLog = (Button) view.findViewById(R.id.btnSignInLog);
         loginLayout = (LinearLayout) view.findViewById(R.id.login_layout);

         shake = AnimationUtils.loadAnimation(getActivity(), R.anim.shake);
    }

    private  void setListeners() {
         btnLog.setOnClickListener(this);
         btnSignInLog.setOnClickListener(this);

    }

    @SuppressLint("ResourceType")
    public void signUp(View v) {
        fragmentManager
                .beginTransaction()
                .setCustomAnimations(R.anim.right_enter, R.anim.left_out)
                .replace(R.id.frame, new SignUpFragment(),
                        Utils.SignUpFragment).commit();
    }

    @SuppressLint("ResourceType")
    @Override
     public void onClick(View v) {

        if (v.getId() == R.id.btnLogIn) {
            validate();
        }
        else if (v.getId() == R.id.btnSignInLog) {
            fragmentManager
                    .beginTransaction()
                    .replace(R.id.frame, new SignUpFragment(),
                            Utils.SignUpFragment).commit();
        }
    }

    private void LoginUser(String email, final String password) {
        try{
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                auth.signInWithEmailAndPassword(email,
password).addOnCompleteListener((Activity) getContext(), new
OnCompleteListener<AuthResult>() {
```

```
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (!task.isSuccessful()) {
                            loginLayout.startAnimation(shake);
                            Toast.makeText(getActivity(), "Authentication
error. Please enter a valid email and password or sign up",
                                    Toast.LENGTH_LONG).show();
                        }
                        else {
                            Toast.makeText(getActivity(), "Login Successful",
Toast.LENGTH_SHORT)
                                    .show();
                            Intent intent = new Intent(getActivity(),
HomeActivity.class);
                            intent.putExtra("userName",
emailText.getText().toString());
                            startActivity(intent);
                        }
                    }
                });
            }
        }
        catch (Exception ex) {
            Toast.makeText(getActivity(), "An error has occurred",
Toast.LENGTH_SHORT)
                    .show();
        }
    }

    private void validate() {
        //String getUsername = username.getText().toString();
        String getPass = password.getText().toString();

        if (getPass.equals("") || getPass.length() == 0) {
            loginLayout.startAnimation(shake);
            Toast.makeText(getActivity(), "You should enter both username and
password",
                    Toast.LENGTH_LONG).show();
        }
        else {
            LoginUser(emailText.getText().toString(),
password.getText().toString());

        }
    }
}
```

HomeActivity Class

```
package com.example.surface.ramhotelapp;

import android.app.FragmentTransaction;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.Fragment;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class HomeActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    String nameValue;
    String username;

    Button b1;
    Button b2;
    Button b3;
    Button adminButton;
    TextView user1;

    private int mButtonChoice = 4;

    private FirebaseAuth auth;
    private FirebaseAuth mAuth = FirebaseAuth.getInstance();
    private String uid = mAuth.getCurrentUser().getUid().toString();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Blocks orientation change
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LOCKED);

        setContentView(R.layout.activity_home);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //Get Firebase auth instance
        auth = FirebaseAuth.getInstance();

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
```

```
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();


        // Adding username to navigation bar

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
        View v = navigationView.getHeaderView(0);
        TextView name = (TextView) v.findViewById(R.id.usernameNav);
        Bundle extras = getIntent().getExtras();
        nameValue = extras.getString("userName");
        name.setText(nameValue);

        // Make adminPage button visible

        adminButton = (Button) v.findViewById(R.id.nav_admin);

        if(nameValue.equals("siva.sotc@gmail.com")
||nameValue.equals("ramgroupofhotels@gmail.com") )
        {
            toggleVisibility(navigationView.getMenu(), R.id.nav_admin, true);
        }

        // end

        if (findViewById(R.id.fragment_container) != null) {

            if (savedInstanceState != null) {
                return;
            }
            Fragment firstFragment = new Fragment();

            firstFragment.setArguments(getIntent().getExtras());

            getSupportFragmentManager().beginTransaction()
                    .add(R.id.fragment_container, firstFragment).commit();
        }
        b1=(Button) findViewById(R.id.btnOne);
        b2=(Button) findViewById(R.id.btnTwo);
        b3=(Button) findViewById(R.id.btnThree);
        b1.setOnClickListener(view -> {
            ServiceFrag f1=new ServiceFrag();
            f1.setArguments(getIntent().getExtras());
            final FragmentTransaction transaction =
getFragmentManager().beginTransaction();
            transaction.replace(R.id.fragment_container, f1);
            transaction.addToBackStack(null);
            transaction.commit();
        });
        b2.setOnClickListener(view -> {
            ServiceFragment2 f2=new ServiceFragment2();
            f2.setArguments(getIntent().getExtras());
```

```
            FragmentTransaction transaction =
getFragmentManager().beginTransaction();
            transaction.replace(R.id.fragment_container, f2);
            transaction.addToBackStack(null);

            transaction.commit();
        });
        b3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ServiceFrag3 f3=new ServiceFrag3();
                f3.setArguments(getIntent().getExtras());

                FragmentTransaction transaction =
getFragmentManager().beginTransaction();
                transaction.replace(R.id.fragment_container, f3);
                transaction.addToBackStack(null);

                transaction.commit();

            }
        });
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.home, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
```

```
        int id = item.getItemId();

        if (id == R.id.nav_room) {
            Intent i = new Intent(HomeActivity.this, Room.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        } else if (id == R.id.nav_activities) {
            Intent i = new Intent(HomeActivity.this,
ActivitiesActivity.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        } else if (id == R.id.nav_roomservice) {
            Intent i = new Intent(HomeActivity.this,
RoomServiceActivity.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        } else if (id == R.id.nav_invoice) {
            Intent i = new Intent(HomeActivity.this, Invoice.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        } else if (id == R.id.nav_admin) {
            Intent i = new Intent(HomeActivity.this,
AdminPage_Activity.class);
            startActivity(i);
        } else if (id == R.id.nav_review) {
            Intent i = new Intent(HomeActivity.this, Rank.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        }
        else if (id==R.id.nav_contact)
        {
            Intent i=new Intent(HomeActivity.this,Contactus.class);
            startActivity(i);
        }
        else if (id== R.id.nav_signout) {
            FirebaseAuth.getInstance().signOut();
            Toast.makeText(HomeActivity.this, "You have successfully Signed
Out", Toast.LENGTH_SHORT).show();
            Intent i = new Intent(HomeActivity.this, MainActivity.class);
            i.putExtra("userName", nameValue);
            startActivity(i);
        }

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    // Method for set visibility of adminPage button

    private void toggleVisibility(Menu menu, int id, boolean visible){
        menu.findItem(id).setVisible(visible);
    }
}
```

Admin Class

```
package com.example.surface.ramhotelapp;

import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
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;


public class AdminPage_Activity extends AppCompatActivity {

    // Connect with database
    private FirebaseDatabase mFirebaseDatabase;
    private FirebaseAuth mAuth;

    // Declaring references to database entities
    private DatabaseReference dbRefUsers;
    private DatabaseReference dbRefActivities;
    private DatabaseReference dbRefBookings;
    private DatabaseReference dbRefOrders;
    private DatabaseReference dbRefRoomReview;

    // Used for holding the value of uniqueId of user information about
should be found
    private String key;

    // Used for holding the value of uniqueId of user information about
should be updated
    private String key1 = "";

    // Used for checking if data is found and displaying messages to user
based of search results
    private boolean found;
    private boolean found1;
    private boolean found2 = false;

    // Declaring user inputs
    EditText name, email, firstName, lastName, region;
    RadioButton rdbTable1, rdbTable2, rdbTable3, rdbTable4, rdbTable5;
```

```
    Button show, find, update, delete, next, save;
    RadioGroup radioGroup;
    TextView resultText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Blocks orientation change
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LOCKED);
        setContentView(R.layout.activity_admin_page_);

        // Initializing the references
        mFirebaseDatabase = FirebaseDatabase.getInstance();
        dbRefUsers = mFirebaseDatabase.getReference("users");
        dbRefActivities = mFirebaseDatabase.getReference("activities");
        dbRefBookings = mFirebaseDatabase.getReference("bookings");
        dbRefOrders = mFirebaseDatabase.getReference("orders");
        dbRefRoomReview = mFirebaseDatabase.getReference("room_review");
        mAuth = FirebaseAuth.getInstance();
        mFirebaseDatabase = FirebaseDatabase.getInstance();

        // Initializing the user inputs
        email = (EditText) findViewById(R.id.etEmail);
        firstName = (EditText) findViewById(R.id.etFirstName);
        lastName = (EditText) findViewById(R.id.etLastName);
        region = (EditText) findViewById(R.id.etRegion);
        radioGroup = (RadioGroup) findViewById(R.id.rbGroup);
        rdbTable1 = (RadioButton) findViewById(R.id.rdbTable1);
        rdbTable2 = (RadioButton) findViewById(R.id.rdbTable2);
        rdbTable3 = (RadioButton) findViewById(R.id.rdbTable3);
        rdbTable4 = (RadioButton) findViewById(R.id.rdbTable2);
        rdbTable5 = (RadioButton) findViewById(R.id.rdbTable3);
        find = (Button) findViewById(R.id.btnFind);
        update = (Button) findViewById(R.id.btnUpdate);
        delete = (Button) findViewById(R.id.btnDelete);
        show = (Button) findViewById(R.id.btnShow);
        save = (Button) findViewById(R.id.btnSave);
        next = (Button) findViewById(R.id.btnNext);
        resultText = (TextView) findViewById(R.id.text);

        // Set properties of inputs to make the page adaptive
        radioGroup.setVisibility(View.GONE);
        email.setVisibility(View.GONE);
        next.setVisibility(View.GONE);
        resultText.setVisibility(View.VISIBLE);
        save.setText("Close");

        // Try catch for preventing crashes
        try
        {
            // Method for displaying data in database
            show.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
```

```
                    email.setVisibility(View.GONE);
                    resultText.setVisibility(View.GONE);
                    next.setVisibility(View.GONE);

                    firstName.setVisibility(View.GONE);
                    lastName.setVisibility(View.GONE);
                    region.setVisibility(View.GONE);

                    radioGroup.setVisibility(View.VISIBLE);
                    next.setVisibility(View.VISIBLE);
                    save.setVisibility(View.VISIBLE);
                    next.setText("Display");
                    save.setText("Close");

                    next.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            resultText.setVisibility(View.VISIBLE);

                            // Check selected radio button
                            int radioButtonID =
radioGroup.getCheckedRadioButtonId();
                            RadioButton radioButton = (RadioButton)
radioGroup.findViewById(radioButtonID);
                            String selectedTable = (String)
radioButton.getText();

                            // Depends on the radio button selected different
requests to database happen
                            switch (selectedTable)
                            {
                                case "Users":
                                    // Listening events for the user`s table
                                    dbRefUsers.addValueEventListener(new
ValueEventListener() {
                                        @Override

                                        // Method for checking if data
changing
                                        public void onDataChange(DataSnapshot
dataSnapshot) {
                                            int i = 1;
                                            resultText.setText("");

                                            // Count of users in database
                                            String count =
String.valueOf(dataSnapshot.getChildrenCount());
                                            resultText.append("      There
are " + count + " users in database: " + '\n' + '\n');

                                            // Foreach loop for every user
                                            for(DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                            {
                                                // Display information of
each user found
                                                resultText.append(
```

```
                                                    "   User #" + i +
'\n'
                                                            + "   Email:
" + postSnapshot.child("email").getValue().toString()
                                                            + '\n' + "
First name: " + postSnapshot.child("firstName").getValue().toString()
                                                            + '\n' + "
Last name: " + postSnapshot.child("lastName").getValue().toString()
                                                            + '\n' + "
Region: " +postSnapshot.child("region").getValue().toString()
                                                            + '\n' + "
Key: " + postSnapshot.getKey()
                                                            + '\n' + "
<---------------------------------------------------------------------------
--------->"+'\n');

                                                // Used for numbering the
users
                                                i++;
                                            }
                                        }
                                        @Override
                                        public void onCancelled(DatabaseError
databaseError) {

Toast.makeText(getApplicationContext(), "     Something went wrong",
Toast.LENGTH_SHORT).show();
                                        }
                                    });
                                    break;

                                case "Activities":
                                    dbRefActivities.addValueEventListener(new
ValueEventListener() {
                                        @Override
                                        public void onDataChange(DataSnapshot
dataSnapshot) {
                                            resultText.setText("");
                                            resultText.append("      List of
booked activities in database: " + '\n' + '\n');

                                            // Foreach loop for each user in
database
                                            for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                            {
                                                // Foreach loop for each user
activity in database
                                                for (DataSnapshot
postSnapshot1: postSnapshot.getChildren())
                                                {
                                                    resultText.append(
                                                            "   Activity: " +
postSnapshot1.child("activity").getValue().toString()
                                                                    + '\n' +
"   Date Of Purchase: " +
postSnapshot1.child("dateOfPurchase").getValue().toString()
```

```
                                                                    + '\n' +
"   Order Id: " +
postSnapshot1.child("Activity_Order_Id").getValue().toString()
                                                                    + '\n' +
"   Price: " + postSnapshot1.child("price").getValue().toString()
                                                                    + '\n' +
"   Quantity: " +postSnapshot1.child("quantity").getValue().toString()
                                                                    + '\n' +
"   UserID: " + postSnapshot1.child("userID").getValue().toString()
                                                                    + '\n' +
"   <-------------------------------------------------------------------------
-------------->"+'\n');
                                                }
                                            }
                                        }
                                        @Override
                                        public void onCancelled(DatabaseError
databaseError) {

Toast.makeText(getApplicationContext(), "     Something went wrong",
Toast.LENGTH_SHORT).show();
                                        }
                                    });
                                    break;

                                case "Bookings":
                                    dbRefBookings.addValueEventListener(new
ValueEventListener() {
                                        @Override
                                        public void onDataChange(DataSnapshot
dataSnapshot) {
                                            resultText.setText("");
                                            resultText.append("      List of
bookings in database: " + '\n' + '\n');
                                            for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                            {
                                                for (DataSnapshot
postSnapshot1: postSnapshot.getChildren())
                                                {
                                                    resultText.append(
                                                            "   Date: " +
postSnapshot1.child("date").getValue().toString()
                                                                    + '\n' +
"   Duration: " + postSnapshot1.child("duration").getValue().toString()
                                                                    + '\n' +
"   Order Id: " + postSnapshot1.child("Room_Order_Id").getValue().toString()
                                                                    + '\n' +
"   Number Of Guests: " +
postSnapshot1.child("numberOfGuests").getValue().toString()
                                                                    + '\n' +
"   Price of Booking: " +
postSnapshot1.child("priceOfBooking").getValue().toString()
                                                                    + '\n' +
"   Type: " + postSnapshot1.child("type").getValue().toString()
                                                                    + '\n' +
"   UserID: " + postSnapshot1.child("userID").getValue().toString()
```

```
                                                                                + '\n' +
"   <-------------------------------------------------------------------------
------------->"+'\n');
                                                    }
                                                }
                                            }
                                            @Override
                                            public void onCancelled(DatabaseError
databaseError) {

Toast.makeText(getApplicationContext(), "     Something went wrong",
Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                        break;

                                    case "Orders":
                                        dbRefOrders.addValueEventListener(new
ValueEventListener() {
                                            @Override
                                            public void onDataChange(DataSnapshot
dataSnapshot) {
                                                resultText.setText("");
                                                resultText.append("      List of
orders in database: " + '\n' + '\n');
                                                for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                                {
                                                    for (DataSnapshot
postSnapshot1: postSnapshot.getChildren())
                                                    {
                                                        resultText.append(
                                                                "   Date: " +
postSnapshot1.child("date").getValue().toString()
                                                                        + '\n' +
"   Order Id: " + postSnapshot1.child("Service_Order_Id" +

"").getValue().toString()
                                                                        + '\n' +
"   Item: " + postSnapshot1.child("item").getValue().toString()
                                                                        + '\n' +
"   Price: " + postSnapshot1.child("price").getValue().toString()
                                                                        + '\n' +
"   Quantity: " + postSnapshot1.child("quantity").getValue().toString()
                                                                        + '\n' +
"   Total Price" + postSnapshot1.child("totalPrice").getValue().toString()
                                                                        + '\n' +
"   UserID: " + postSnapshot.getKey().toString()
                                                                        + '\n' +
"   <-------------------------------------------------------------------------
------------->"+'\n');
                                                    }
                                                }
                                            }
                                            @Override
                                            public void onCancelled(DatabaseError
databaseError) {
```

```
Toast.makeText(getApplicationContext(), "     Something went wrong",
Toast.LENGTH_SHORT).show();
                                    }
                                });
                                break;

                            case "Reviews":
                                dbRefRoomReview.addValueEventListener(new
ValueEventListener() {
                                    @Override
                                    public void onDataChange(DataSnapshot
dataSnapshot) {
                                        resultText.setText("");
                                        resultText.append("      List of
reviews in database: " + '\n' + '\n');
                                        for (DataSnapshot postSnapshot1:
dataSnapshot.getChildren())
                                        {
                                            resultText.append(
                                                    "   Rank: " +
postSnapshot1.child("rank").getValue().toString()
                                                            + '\n' + "
Review: " + postSnapshot1.child("review").getValue().toString()
                                                            + '\n' + "
UserID: " + postSnapshot1.child("userID").getValue().toString()
                                                            + '\n' + "
<---------------------------------------------------------------------------
--------->"+'\n');
                                        }
                                    }
                                    @Override
                                    public void onCancelled(DatabaseError
databaseError) {

Toast.makeText(getApplicationContext(), "Something went wrong",
Toast.LENGTH_SHORT).show();
                                    }
                                });
                                break;
                            }
                        }
                    });

                    save.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            next.setVisibility(View.GONE);
                            save.setVisibility(View.GONE);
                            radioGroup.setVisibility(View.GONE);
                            resultText.setVisibility(View.GONE);

                        }
                    });
                }
            });
```

```
            // Method for searching all user`s data in database
            find.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    radioGroup.setVisibility(View.GONE);
                    save.setVisibility(View.GONE);
                    email.setText("");
                    email.setVisibility(View.VISIBLE);
                    next.setVisibility(View.VISIBLE);

                    found1 = false;

                    next.setText("Find");
                    resultText.setText("");

                    firstName.setVisibility(View.GONE);
                    lastName.setVisibility(View.GONE);
                    region.setVisibility(View.GONE);

                    next.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            key = "";
                            found1 = false;
                            resultText.setText("");

                            // Listening events in users table
                            dbRefUsers.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {

                                    // foreach loop in users table
                                    for(DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                    {
                                        // Check if snapshot of data related
to user`s email

if(postSnapshot.child("email").getValue().toString().equals(email.getText().t
oString()))
                                        {
                                            found1 = true;
                                            resultText.setText("");
                                            resultText.append(
                                                    "   Email: " +
postSnapshot.child("email").getValue().toString()
                                                    + '\n' + "
First name: " + postSnapshot.child("firstName").getValue().toString()
                                                    + '\n' + "   Last
name: " + postSnapshot.child("lastName").getValue().toString()
                                                    + '\n' + "
Region: " + postSnapshot.child("region").getValue().toString()
                                                    + '\n' + "   Key:
" + postSnapshot.getKey()
                                                    + '\n' + "   <---
```

```
------------------------------------------------------------------------------
----->"+'\n');

                                        // Save user key from users table
for future searches in other entities
                                        key = postSnapshot.getKey();
                                    }
                                }

                                // Display results if user`s email wasn`t
found in users entity
                                if(!found1)
                                {

resultText.setVisibility(View.VISIBLE);
                                    resultText.setText("   User " +
email.getText().toString() + " is not found");
                                    //
Toast.makeText(getApplicationContext(), "User " + email.getText().toString()
+ " is not found", Toast.LENGTH_SHORT).show();
                                }
                            }


                            @Override
                            public void onCancelled(DatabaseError
databaseError) {
                                Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                            }
                        });

                        // Searching rest of the tables by key which was
saved from users table
                        dbRefActivities.addValueEventListener(new
ValueEventListener() {
                            @Override
                            public void onDataChange(DataSnapshot
dataSnapshot) {
                                for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                {
                                    for (DataSnapshot postSnapshot1:
postSnapshot.getChildren())
                                    {
                                        if
(postSnapshot1.child("userID").getValue().toString().equals(key))
                                        {
                                            resultText.append("      User
" + email.getText().toString() + " booked: " + '\n' + '\n');
                                            resultText.append(
                                                    "   Activity: " +
postSnapshot1.child("activity").getValue().toString()
                                                            + '\n' + "
Date Of Purchase: " +
postSnapshot1.child("dateOfPurchase").getValue().toString()
                                                            + '\n' + "
```

```
Price: " + postSnapshot1.child("price").getValue().toString()
                                                                + '\n' + "
Quantity: " + postSnapshot1.child("quantity").getValue().toString()
                                                                + '\n' +"
<-------------------------------------------------------------------------
--------->"+ '\n');
                                            }
                                        }
                                    }
                                }
                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                    Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                }
                            });

                            dbRefBookings.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {
                                    for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                    {
                                        for (DataSnapshot postSnapshot1:
postSnapshot.getChildren())
                                        {
                                            if
(postSnapshot1.child("userID").getValue().toString().equals(key))
                                            {
                                                resultText.append("      User
" + email.getText().toString() + " booked: " + '\n' + '\n');
                                                resultText.append(
                                                        "   Date: " +
postSnapshot1.child("date").getValue().toString()
                                                                + '\n' + "
Duration: " + postSnapshot1.child("duration").getValue().toString()
                                                                + '\n' + "
Number Of Guests: " +
postSnapshot1.child("numberOfGuests").getValue().toString()
                                                                + '\n' + "
Price of Booking: " +
postSnapshot1.child("priceOfBooking").getValue().toString()
                                                                + '\n' + "
Type: " + postSnapshot1.child("type").getValue().toString()
                                                                + '\n' +"
<-------------------------------------------------------------------------
--------->"+ '\n');
                                            }
                                        }
                                    }
                                }
                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
```

```
                                    Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                }
                            });

                            dbRefOrders.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {
                                    for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                    {
                                        for (DataSnapshot postSnapshot1:
postSnapshot.getChildren())
                                        {
                                            if
(postSnapshot.getKey().toString().equals(key))
                                            {
                                                resultText.append("      User
" + email.getText().toString() + " ordered: " + '\n' + '\n');
                                                resultText.append(
                                                        "   Date: " +
postSnapshot1.child("date").getValue().toString()
                                                                + '\n' + "
Item: " + postSnapshot1.child("item").getValue().toString()
                                                                + '\n' + "
Price: " + postSnapshot1.child("price").getValue().toString()
                                                                + '\n' + "
Quantity: " + postSnapshot1.child("quantity").getValue().toString()
                                                                + '\n' + "
Total Order:" +postSnapshot1.child("totalPrice").getValue().toString()
                                                                + '\n' + "
<-------------------------------------------------------------------------
--------->"+'\n');
                                            }
                                        }
                                    }
                                }
                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                    Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                }
                            });

                            dbRefRoomReview.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {
                                    for (DataSnapshot postSnapshot:
dataSnapshot.getChildren())
                                    {
                                        if
(postSnapshot.child("userID").getValue().toString().equals(key) ||
```

```
dataSnapshot.getKey().equals(key))
                                        {
                                            resultText.append("      User " +
email.getText().toString() + " left review: " + '\n' + '\n');
                                            resultText.append(
                                                    "   Rank: " +
postSnapshot.child("rank").getValue().toString()
                                                            + '\n' + "
Review: " + postSnapshot.child("review").getValue().toString()
                                                            + '\n' + "   <---
-------------------------------------------------------------------------
----->"+'\n');
                                        }
                                    }
                                }

                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                    Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                }
                            });
                        }
                    });
                }
            });

            // Method for update user`s information in users table
            update.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    resultText.setVisibility(View.GONE);
                    radioGroup.setVisibility(View.GONE);
                    save.setVisibility(View.GONE);
                    email.setVisibility(View.GONE);
                    save.setText("Update");
                    next.setText("Find");
                    email.setVisibility(View.VISIBLE);
                    next.setVisibility(View.VISIBLE);

                    found = false;

                    firstName.setText("");
                    lastName.setText("");
                    region.setText("");

                    firstName.setVisibility(View.GONE);
                    lastName.setVisibility(View.GONE);
                    region.setVisibility(View.GONE);



                    next.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            email.setVisibility(View.VISIBLE);
```

```
                            next.setVisibility(View.GONE);
                            save.setText("Find");

                            dbRefUsers.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {

                                    // Try catch for handling exceptions
                                    try
                                    {

                                        // Foreach loop for searching user by
email in user`s table
                                        for (DataSnapshot postSnapshot :
dataSnapshot.getChildren())
                                        {
                                            if
(postSnapshot.child("email").getValue().toString().equals(email.getText().toS
tring()))
                                            {
                                                found = true;

                                                // Display editText fields
with current information about user

firstName.setVisibility(View.VISIBLE);

lastName.setVisibility(View.VISIBLE);

region.setVisibility(View.VISIBLE);

save.setVisibility(View.VISIBLE);
                                                save.setText("Update");
                                                resultText.setText("");


firstName.setText(postSnapshot.child("firstName").getValue().toString());

lastName.setText(postSnapshot.child("lastName").getValue().toString());

region.setText(postSnapshot.child("region").getValue().toString());
                                                key1 = postSnapshot.getKey();
                                            }
                                        }
                                        if(!found)
                                        {

resultText.setVisibility(View.VISIBLE);
                                            resultText.setText("      User "
+ email.getText().toString() + " is not found");
                                            next.setVisibility(View.VISIBLE);
                                        }
                                    }
                                    catch (Exception ex){}
                                }
```

```
                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                    Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                }
                            });

                            // Update method
                            save.setOnClickListener(new
View.OnClickListener() {
                                @Override
                                public void onClick(View view) {

                                    // Setting new values which were taken
from three editText fields to the user object which is found by key1 from
search part of update method

dbRefUsers.child(key1).child("firstName").setValue(firstName.getText().toStri
ng());

dbRefUsers.child(key1).child("lastName").setValue(lastName.getText().toString
());

dbRefUsers.child(key1).child("region").setValue(region.getText().toString());

                                    resultText.setText("");
                                    Toast confirmToast =
Toast.makeText(AdminPage_Activity.this, "Updated", Toast.LENGTH_SHORT);
                                    confirmToast.show();
                                    found = true;

                                    save.setVisibility(View.GONE);
                                    next.setVisibility(View.GONE);

                                    email.setText("");
                                    firstName.setText("");
                                    lastName.setText("");
                                    region.setText("");

                                    email.setVisibility(View.GONE);
                                    firstName.setVisibility(View.GONE);
                                    lastName.setVisibility(View.GONE);
                                    region.setVisibility(View.GONE);
                                }
                            });
                        }
                    });
                }
            });

            // Method for deleting user
            delete.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    resultText.setVisibility(View.GONE);
```

```
                    save.setVisibility(View.GONE);
                    radioGroup.setVisibility(View.GONE);
                    email.setVisibility(View.VISIBLE);
                    next.setVisibility(View.VISIBLE);
                    next.setText("Delete");

                    found2 = false;

                    next.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            dbRefUsers.addValueEventListener(new
ValueEventListener() {
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {

                                    // Try catch for handling exceptions
                                    try
                                    {
                                        // Searching the user by email
                                        for (DataSnapshot postSnapshot :
dataSnapshot.getChildren())
                                        {
                                            if
(postSnapshot.child("email").getValue().toString().equals(email.getText().toS
tring()))
                                            {

dbRefUsers.child(postSnapshot.getKey()).removeValue();

dbRefActivities.child(postSnapshot.getKey()).removeValue();

dbRefBookings.child(postSnapshot.getKey()).removeValue();

dbRefOrders.child(postSnapshot.getKey()).removeValue();

dbRefRoomReview.child(postSnapshot.getKey()).removeValue();

resultText.setVisibility(View.VISIBLE);

                                                resultText.setText("
User " + email.getText().toString() + " is not found");

next.setVisibility(View.GONE);

email.setVisibility(View.GONE);
                                                found2 = true;
                                            }
                                        }
                                        if(!found2)
                                        {

resultText.setVisibility(View.VISIBLE);

                                            resultText.setText("      User "
+ email.getText().toString() + " is not found");
```

```
                                                next.setVisibility(View.VISIBLE);
                                            }
                                        }
                                        catch (Exception ex){}
                                    }

                                    @Override
                                    public void onCancelled(DatabaseError
databaseError) {
                                        Toast.makeText(getApplicationContext(), "
Something went wrong", Toast.LENGTH_SHORT).show();
                                    }
                                });
                            }
                        });
                    }
                });
            }

            // Toast the warning message if something went wrong
            catch (Exception ex)
            {
                Toast.makeText(getApplicationContext(), "      Something went
wrong ¯\\_(ツ)_/¯", Toast.LENGTH_SHORT).show();
            }

        }
}
```

Invoice Class

```
package com.example.surface.ramhotelapp;

import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.widget.ListView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

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.math.BigDecimal;
import java.math.RoundingMode;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
```

```
public class Invoice extends AppCompatActivity {

    //// Declaring global variables
    private String name;
    private FirebaseDatabase mFirebaseDatabase;
    private FirebaseAuth mAuth;
    private DatabaseReference dbRef,dbBookRef,dbActRef,dbOrdRef;
    private String uid ;
    private TextView tvName;
    private double overallCharges;
    private ListView invoiceList;
    private TextView overallTxt;
    private TextView totalTxt;
    private TextView gstTxt;
    //list of type InvoiceHandler that holds invoice list
    List<InvoiceHandler> invoices= new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Blocks orientation change
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LOCKED);
        setContentView(R.layout.activity_invoice);

        //giving values to global variables
        mAuth = FirebaseAuth.getInstance();
        mFirebaseDatabase=FirebaseDatabase.getInstance();
        dbRef = mFirebaseDatabase.getReference("users");
        FirebaseUser user=mAuth.getCurrentUser();
        uid=user.getUid();
        //getting exact nodes for invoice data
        dbBookRef=mFirebaseDatabase.getReference("bookings").child(uid);
        dbActRef=mFirebaseDatabase.getReference("activities").child(uid);
        dbOrdRef=mFirebaseDatabase.getReference("orders").child(uid);
        tvName=(TextView) findViewById(R.id.name);
        totalTxt=(TextView)findViewById(R.id.totalcost);
        //list to populate global list
        invoiceList= (ListView)findViewById(R.id.invoiceList);

        dbRef.addValueEventListener(new ValueEventListener() { //calling
method to set data(user name)
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                setData(dataSnapshot);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
       dbBookRef.addValueEventListener(new ValueEventListener() {
           @Override
           public void onDataChange(DataSnapshot dataSnapshot) {//calling
method to get bookings and fill list
```

```
                fillBookings(dataSnapshot);

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
        dbOrdRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {//calling
method to fill  orders
                fillOrders(dataSnapshot);

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
        dbActRef.addValueEventListener(new ValueEventListener() {//calling
method to fill activities
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                fillActivities(dataSnapshot);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
         //fillInvoice();

    }

    private void setData(DataSnapshot dataSnapshot) {
        for(DataSnapshot ds : dataSnapshot.getChildren())
        {//retrieves user name from 'user' node using current user's uid.
            User user = dataSnapshot.child(uid).getValue(User.class);
            name=user.firstName+" "+user.lastName;
        }
        tvName.setText(name);

    }
    private void fillOrders(DataSnapshot dataSnapshot) {
        for(DataSnapshot ds : dataSnapshot.getChildren())
        {
            Orders orders = ds.getValue(Orders.class);

            InvoiceHandler invoiceHandler = new InvoiceHandler();

            invoiceHandler.setDate(orders.date);
            invoiceHandler.setOrderId(orders.Service_Order_Id);
            BigDecimal bd = new BigDecimal(orders.totalPrice);
```

```
            bd = bd.setScale(2, RoundingMode.HALF_UP);
            double d = bd.doubleValue();
            invoiceHandler.setCharges(d);
            invoiceHandler.setService(orders.item + " ( " + orders.price + "$
* " + orders.quantity + " )");
            invoices.add(invoiceHandler);
        }
        fillInvoice();
    }


    private void fillBookings(DataSnapshot dataSnapshot) {
        for(DataSnapshot ds : dataSnapshot.getChildren())
        {//retrieves details of current user's bookings
                Bookings book = ds.getValue(Bookings.class);
                InvoiceHandler invoiceHandler = new InvoiceHandler();
                invoiceHandler.setDate(book.date);
                invoiceHandler.setOrderId(book.Room_Order_Id);
                invoiceHandler.setService(book.type);
                BigDecimal bd = new BigDecimal(book.priceOfBooking);
                bd = bd.setScale(2, RoundingMode.HALF_UP);
                double d = bd.doubleValue();
                invoiceHandler.setCharges(d);
                invoices.add(invoiceHandler);
        }
        fillInvoice();
    }
    private void fillActivities(DataSnapshot dataSnapshot) {
        for (DataSnapshot ds : dataSnapshot.getChildren()) {
            //retrieves details of the activites booked by current user
            Activities activity = ds.getValue(Activities.class);
            InvoiceHandler invoiceHandler = new InvoiceHandler();
            invoiceHandler.setOrderId(activity.Activity_Order_Id);
            invoiceHandler.setDate(activity.dateOfPurchase);
            invoiceHandler.setService(activity.activity);
            BigDecimal bd = new BigDecimal(activity.price);
            bd = bd.setScale(2, RoundingMode.HALF_UP);
            double d = bd.doubleValue();
            invoiceHandler.setCharges(d);
            invoices.add(invoiceHandler);
        }
        fillInvoice();
    }
    public void fillInvoice()
    {//calculating charges
        overallCharges=0;
        for (int i = 0; i < invoices.size(); i++) {
            overallCharges+= invoices.get(i).getCharges();
        }
    //formatting charges
        NumberFormat format =
NumberFormat.getCurrencyInstance(Locale.CANADA);
        String overall = format.format(overallCharges);
        totalTxt.setText("Grand Total till this Date: " + overall);

        InvoiceAdpater adpater= new InvoiceAdpater(Invoice.this, invoices);
        invoiceList.setAdapter(adpater);
```

```
    }
}
```

SignUp Class

```
package com.example.surface.ramhotelapp;

import static android.content.ContentValues.TAG;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Build;
import android.os.Bundle;
import android.text.InputType;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.annotation.NonNull;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.regex.Pattern;


public class SignUpFragment extends Fragment implements View.OnClickListener
{
        //for firebase
        private FirebaseAuth mAuth;
        private DatabaseReference mDatabase;
        private Button back1;
        private static View view;

        //to hold the values in Spinner
        public static final CharSequence[] REGION_OPTION = {"North America",
"Latin America", "Europe", "Africa", "Asia", "Australia"};
        //declaring page elements
```

```
        private static EditText passwordSign, passConfirm, emailSign;
        private static Button btnSign;
        private static CheckBox hidePasswordSign;
        private static LinearLayout SignUpLayout;
        private static Animation shake;
        private static FragmentManager fragmentManager;
        private static EditText etfirstName,etlastName;

   public SignUpFragment() {

        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                Bundle savedInstance) {
            view = inflater.inflate(R.layout.fragment_sign_up, container,
false);
            mAuth =  FirebaseAuth.getInstance();
            mDatabase = FirebaseDatabase.getInstance().getReference();

            initViews();
            setListeners();
            return view;
        }
    //to initialize the variables
        private void initViews() {

            // Properly initializes and populates the "region" spinner
            Spinner spinner1 = (Spinner) view.findViewById(R.id.spRegion);
            ArrayAdapter<CharSequence> adapter = new
ArrayAdapter<CharSequence>(getActivity(),
android.R.layout.simple_spinner_item, REGION_OPTION);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
); // Specify the layout to use when the list of choices appears
            spinner1.setAdapter(adapter); // Apply the adapter to the spinner


            fragmentManager = getFragmentManager();
            passwordSign = (EditText) view.findViewById(R.id.etPasswordSign);
            emailSign = (EditText) view.findViewById(R.id.etEmail);
            passConfirm = (EditText) view.findViewById(R.id.etPassConfirm);
            btnSign = (Button) view.findViewById(R.id.btnSignUp);
            hidePasswordSign = (CheckBox)
view.findViewById(R.id.chboxHidePasswordSign);
            SignUpLayout = (LinearLayout)
view.findViewById(R.id.signup_layout);
            etfirstName = (EditText) view.findViewById(R.id.etFirstName);
            etlastName = (EditText) view.findViewById(R.id.etLastName);
            back1 =(Button) view.findViewById(R.id.backtologin);
            shake = AnimationUtils.loadAnimation(getActivity(),
R.anim.shake);
        }
        //to set onclick listeners for checkbox and button
        private  void setListeners() {
            btnSign.setOnClickListener(this);
```

```
            back1.setOnClickListener(this);
            //show and hide password
            hidePasswordSign
                    .setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

                        @Override
                        public void onCheckedChanged(CompoundButton button,
                                                     boolean isChecked) {


                            if (isChecked) {

                                hidePasswordSign.setText("Hide Password");


passwordSign.setInputType(InputType.TYPE_CLASS_TEXT);

passwordSign.setTransformationMethod(HideReturnsTransformationMethod
                                        .getInstance());
                            } else {
                                hidePasswordSign.setText("Show Password");


passwordSign.setInputType(InputType.TYPE_CLASS_TEXT
                                        |
InputType.TYPE_TEXT_VARIATION_PASSWORD);

passwordSign.setTransformationMethod(PasswordTransformationMethod
                                        .getInstance());

                            }

                        }
                    });
        }

        //onclick on the SignUp button
        @Override
        public void onClick(View v) {

            if (v.getId() == R.id.btnSignUp) {
                validate();
            }
            else if(v.getId() == R.id.backtologin)
            {
                fragmentManager
                        .beginTransaction()
                        .replace(R.id.frame, new LogInFragment(),
                                Utils.LogInFragment).commit();
            }

    }

    //to hold the writing of a user to auth and db
        private void signUpUser(final String email, final String password,
final String firstName, final String lastName,final String region){
```

```
            try {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener((Activity) getContext(), new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult>
task) {
                            Log.d(TAG, "createUserWithEmail:onComplete" +
task.isSuccessful());
                            if(task.isSuccessful())
                            {
                                // Attempts to login with the recently
created user to add it to the database
                                try {

                                    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener((Activity) getContext(), new
OnCompleteListener<AuthResult>() {
                                        @SuppressLint("ResourceType")
                                        @Override
                                        public void onComplete(@NonNull
Task<AuthResult> task) {
                                            String uid =
mAuth.getCurrentUser().getUid().toString();
                                            User user = new User(email,
firstName, lastName, region);

mDatabase.child("users").child(uid).setValue(user);
                                            mAuth.signOut(); // Signs out the
user, as we only signed in to add it to the DB
                                            Toast.makeText(getActivity(),
                                                    "Congrats! User Signup
Successful, Please Login here",

Toast.LENGTH_LONG).show();
                                            fragmentManager
                                                    .beginTransaction()
                                                    .replace(R.id.frame, new
LogInFragment(),

Utils.LogInFragment).commit();
                                        }
                                    });
                                }
                                // if it fails to add the user to the DB
                                catch (Exception ex)    {
                                     Toast.makeText(getActivity(),
                                    "Adding User failed, please contact our
team at ramhotels@gmail.com",
                                    Toast.LENGTH_LONG).show();
                                } // End of Catch for signing in


                            }
                            else {
                                Log.d(TAG, "Authentication failed" +
```

```
task.getException());
                                SignUpLayout.startAnimation(shake);
                                Toast.makeText(getActivity(), "Authentication
failed",
                                        Toast.LENGTH_LONG).show();
                            }

                        }
                    });
                }
            }
            catch (Exception ex) {
                SignUpLayout.startAnimation(shake);
                Toast.makeText(getActivity(), "An error has occurred ¯\\_(ツ
)_/¯",
                        Toast.LENGTH_LONG).show();
            }


        }
        //check for a correct email pattern through a regular expression
        public static boolean isValidEmail(String email) {
            String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\."+
                    "[a-zA-Z0-9_+&*-]+)*@" +
                    "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
                    "A-Z]{2,7}$";

            Pattern pat = Pattern.compile(emailRegex);
            if (email == null)
                return false;
            return pat.matcher(email).matches();
        }
        //this method validates the user input and signs him up if everything
is correct
        private void validate() {
            //page elements
            String getPass = passwordSign.getText().toString();
            String getEmail = emailSign.getText().toString();
            String getPassConfirm = passConfirm.getText().toString();
            // ADDED to get the values of the new fields
            String firstName = etfirstName.getText().toString();
            String lastName = etlastName.getText().toString();
            Spinner spinner = (Spinner) view.findViewById(R.id.spRegion);
            String region = spinner.getSelectedItem().toString();

            if (getPass.isEmpty()
                || getEmail.isEmpty()
                || getPassConfirm.isEmpty()
                || firstName.isEmpty()
                || lastName.isEmpty())  {

                SignUpLayout.startAnimation(shake);
                Toast.makeText(getActivity(), "All fields are required",
                        Toast.LENGTH_LONG).show();

            }
```

```
            else {
                if  (!isValidEmail(getEmail)){
                    emailSign.startAnimation(shake);
                    Toast.makeText(getActivity(), "Please enter a valid email
address", Toast.LENGTH_SHORT).show();
                }
                else if (getPass.length() < 7) {
                    passwordSign.startAnimation(shake);
                    Toast.makeText(getActivity(), "The password should be at
least 8 characters long", Toast.LENGTH_SHORT).show();
                }
                else if (!getPassConfirm.equals(getPass)) {
                    passwordSign.startAnimation(shake);
                    passConfirm.startAnimation(shake);
                    Toast.makeText(getActivity(), "Passwords do not match",
Toast.LENGTH_SHORT).show();
                }
                else {
                    //if the validation was successful, he gets signed up and
the object is written to the db
                    signUpUser(emailSign.getText().toString(),
passwordSign.getText().toString(),firstName,lastName,region);
                }


            }
        }
    //onclick on the Back to Login button



}
```

RoomService Class

```
package com.example.surface.ramhotelapp;

import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.AsyncTask;
import android.os.Handler;
import android.os.SystemClock;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import android.widget.Toast;

public class RoomServiceActivity extends AppCompatActivity {
    //Page elements
    TextView descText1, descText2, descText3;
    ImageButton more1, less1, more2, less2, more3, less3;
    Button btn1, btn2, btn3;

    public Boolean fragmentShown = false;
    public String booked = "No";
    String nameValue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Blocks orientation change
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LOCKED);

        setContentView(R.layout.activity_room_service);
        //gets the username string for a NavBar
        Bundle extrasName = getIntent().getExtras();
        nameValue = extrasName.getString("userName");
        //home button
        FloatingActionButton fabHome = (FloatingActionButton)
findViewById(R.id.floatingActionButton3);
        fabHome.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(RoomServiceActivity.this,
HomeActivity.class);
                i.putExtra("userName", nameValue);
                startActivity(i);
            }
        });
        //three ImageViews used for Threads
        final ImageView imgActivity1 = (ImageView)
findViewById(R.id.imageViewService);
        final ImageView imgActivity2 = (ImageView)
findViewById(R.id.imageView2Service);
        final ImageView imgActivity3 = (ImageView)
findViewById(R.id.imageView3Service);
        //Page elements
        descText1 = (TextView) findViewById(R.id.description_text1Service);
        more1 = (ImageButton) findViewById(R.id.imgBtnMoreService);
        less1 = (ImageButton) findViewById(R.id.imgBtnLessService);

        descText2 = (TextView) findViewById(R.id.description_text2Service);
        more2 = (ImageButton) findViewById(R.id.imgBtnMore1Service);
        less2 = (ImageButton) findViewById(R.id.imgBtnLess1Service);

        descText3 = (TextView) findViewById(R.id.description_text3Service);
        more3 = (ImageButton) findViewById(R.id.imgBtnMore3Service);
        less3 = (ImageButton) findViewById(R.id.imgBtnLess3Service);

        btn1 = (Button) findViewById(R.id.button1Service); // Book Breakfast
        btn2 = (Button) findViewById(R.id.button2Service); // Book Lunch
```

```
        btn3 = (Button) findViewById(R.id.button3Service); // Book Dinner
        //puts the username string to NavBar
        final Intent i = new Intent(RoomServiceActivity.this,
roomservice_booking.class);
        i.putExtra("userName", nameValue);
        // Thread to change the image of a room service (breakfast)
        new Thread(new Runnable() {
            @Override
            public void run() {
                int i=0;
                while (i < 4) {
                    SystemClock.sleep(2000);
                    i++;
                    final int curCount = i;
                    if (curCount == 0) {
                        imgActivity1.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity1.setImageResource(R.drawable.breakfast);
                            }
                        });
                    }
                    else if (curCount == 1) {
                        imgActivity1.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity1.setImageResource(R.drawable.breakfast1);
                            }
                        });
                    }
                    else if (curCount == 2) {
                        imgActivity1.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity1.setImageResource(R.drawable.breakfast2);
                            }
                        });
                    }
                    else if (curCount == 3){
                        i = -1;
                    }
                }
            }
        }).start();

        new Thread(new Runnable() {
            @Override
            public void run() {
                int i=0;
                while (i < 4) {
                    SystemClock.sleep(2000);
                    i++;
                    final int curCount = i;
                    if (curCount == 0) {
```

```
                        imgActivity2.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity2.setImageResource(R.drawable.lunch1);
                            }
                        });
                    }
                    else if (curCount == 1) {
                        imgActivity2.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity2.setImageResource(R.drawable.lunch2);
                            }
                        });
                    }
                    else if (curCount == 2) {
                        imgActivity2.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity2.setImageResource(R.drawable.lunch3);
                            }
                        });
                    }
                    else if (curCount == 3){
                        i = -1;
                    }
                }
            }
        }).start();

        new Thread(new Runnable() {
            @Override
            public void run() {
                int i=0;
                while (i < 4) {
                    SystemClock.sleep(2000);
                    i++;
                    final int curCount = i;
                    if (curCount == 0) {
                        imgActivity3.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity3.setImageResource(R.drawable.dessert1);
                            }
                        });
                    }
                    else if (curCount == 1) {
                        imgActivity3.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity3.setImageResource(R.drawable.dessert2);
                            }
```

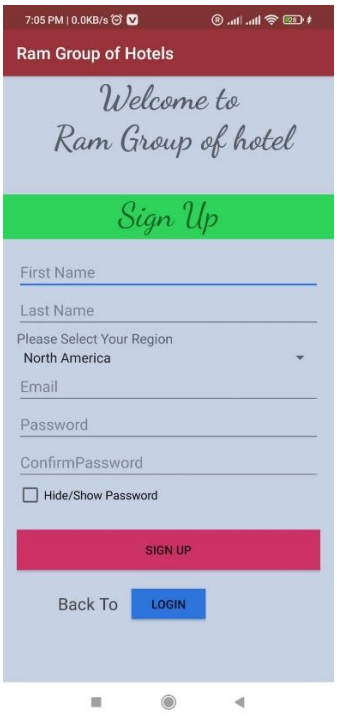
```
                        });
                    }
                    else if (curCount == 2) {
                        imgActivity3.post(new Runnable() {
                            @Override
                            public void run() {

imgActivity3.setImageResource(R.drawable.dessert3);
                            }
                        });
                    }
                    else if (curCount == 3){
                        i = -1;
                    }
                }
            }
        }).start();

        // Book Breakfast
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                i.putExtra("booked?", booked);
                i.putExtra("meal", "breakfast");
                startActivity(i);
            }
        });
        // Book Lunch
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                i.putExtra("booked?", booked);
                i.putExtra("meal", "lunch");
                startActivity(i);

            }
        });
        // Book Dinner
        btn3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                i.putExtra("booked?", booked);
                i.putExtra("meal", "dessert");
                startActivity(i);

            }
        });
        //expand the menu text for breakfast
        more1.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {

                more1.setVisibility(View.GONE);
                less1.setVisibility(View.VISIBLE);
                descText1.setMaxLines(Integer.MAX_VALUE);
```

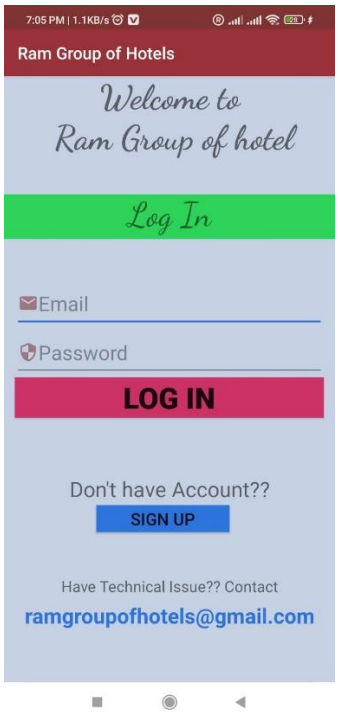
```
        }
    });
    //hide the menu text for breakfast and leave only 2 lines
    less1.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {

            less1.setVisibility(View.GONE);
            more1.setVisibility(View.VISIBLE);
            descText1.setMaxLines(2);

        }
    });

    more2.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {

            more2.setVisibility(View.GONE);
            less2.setVisibility(View.VISIBLE);
            descText2.setMaxLines(Integer.MAX_VALUE);

        }
    });

    less2.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {

            less2.setVisibility(View.GONE);
            more2.setVisibility(View.VISIBLE);
            descText2.setMaxLines(2);

        }
    });

    more3.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {

            more3.setVisibility(View.GONE);
            less3.setVisibility(View.VISIBLE);
            descText3.setMaxLines(Integer.MAX_VALUE);

        }
    });

    less3.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
```

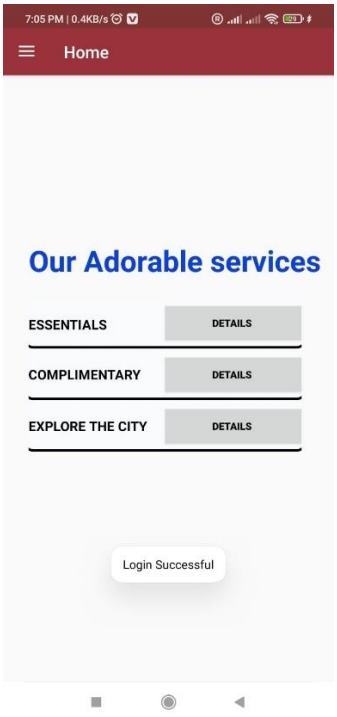
```
                less3.setVisibility(View.GONE);
                more3.setVisibility(View.VISIBLE);
                descText3.setMaxLines(2);

            }
        });

        // Show confirmation fragment if user pressed "Book" button
        try
        {
            Bundle extras = getIntent().getExtras();
            booked = extras.getString("booked?");

            if(booked.equals("Yes"))
            {
                showConfirmation();
            }
        }
        catch (Exception ex)
        {

        }
    }


    // Methods to make confirmation fragment visible
    public void displayFragment()
    {
        ConfirmationFragment confirmationFragment =
ConfirmationFragment.newInstance(3);

        FragmentManager fragmentManager = getFragmentManager();
        android.app.FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        fragmentTransaction.add(R.id.fragment_containerService,
confirmationFragment).addToBackStack(null).commit();

        fragmentShown = true;
    }
    // Methods to make confirmation fragment invisible
    public void closeFragment()
    {
        FragmentManager fragmentManager = getFragmentManager();
        ConfirmationFragment confirmationFragment = (ConfirmationFragment)
fragmentManager
                .findFragmentById(R.id.fragment_containerService);
        if(confirmationFragment != null)
        {
            FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
            fragmentTransaction.remove(confirmationFragment).commit();
        }

        fragmentShown = false;
    }
    // Method for handling fragment visibility in the page
    public void showConfirmation()
```

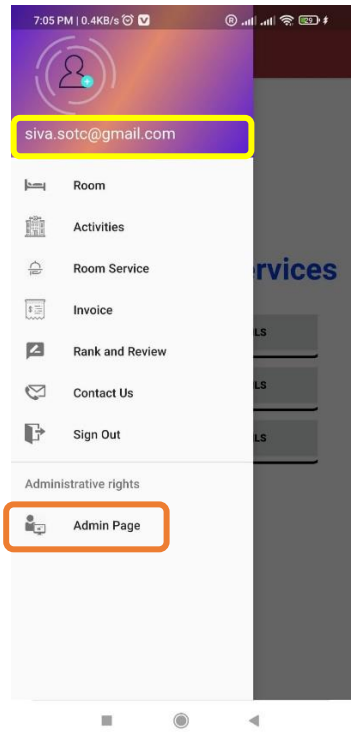
```
    {
        displayFragment();

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                closeFragment();
            }
        }, 1000);
    }
}
```

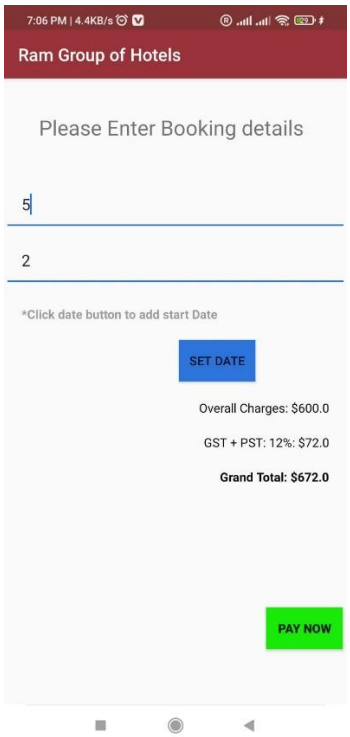
5. Results & Screenshots

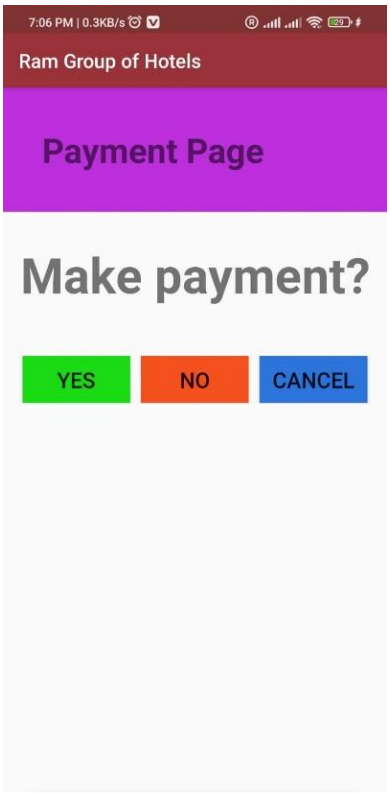


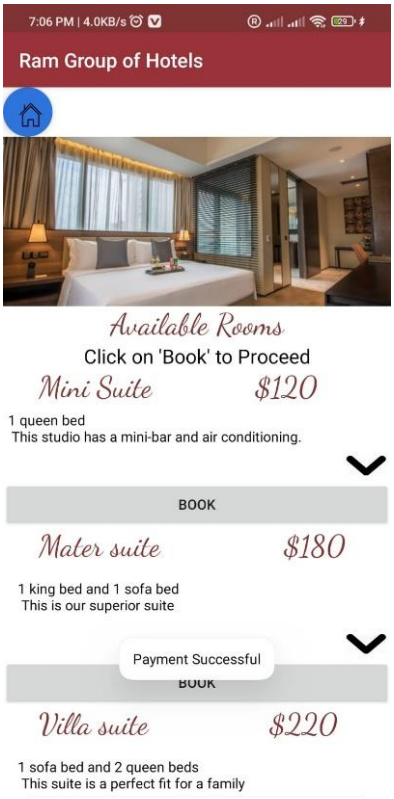


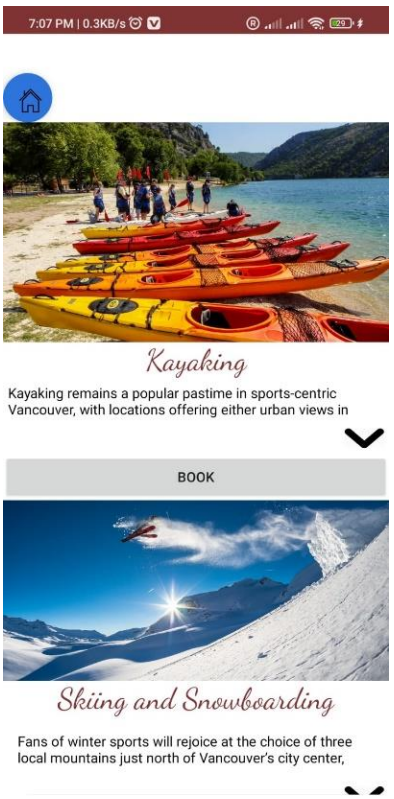


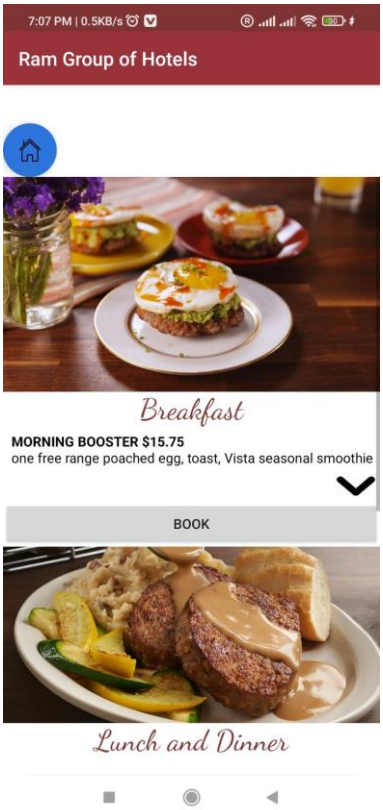


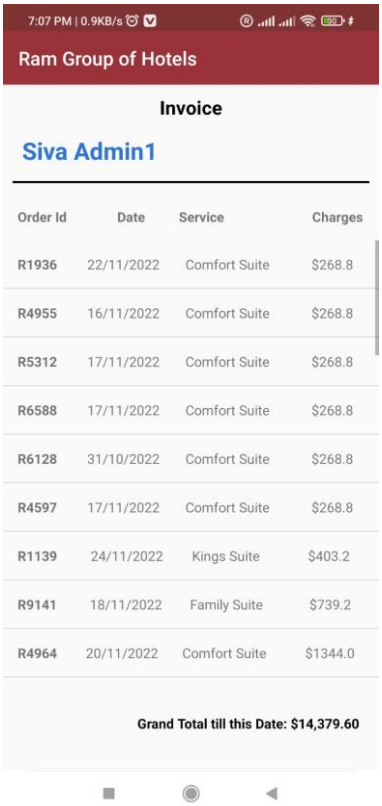


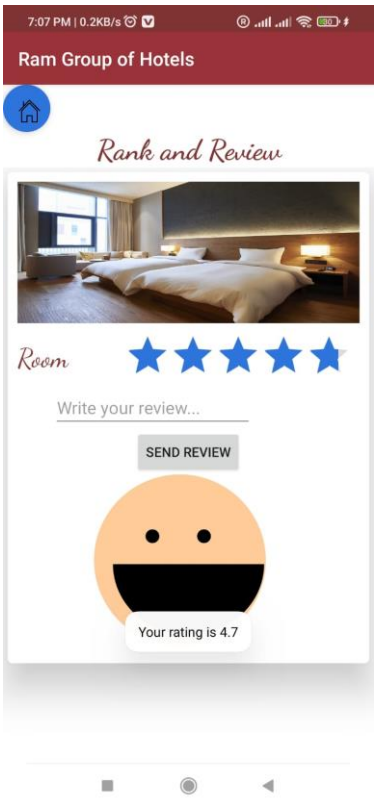

7:06 PM | 0.3KB/s
Ram Group of Hotels
Payment Page
Make payment?
YES
NO
CANCEL

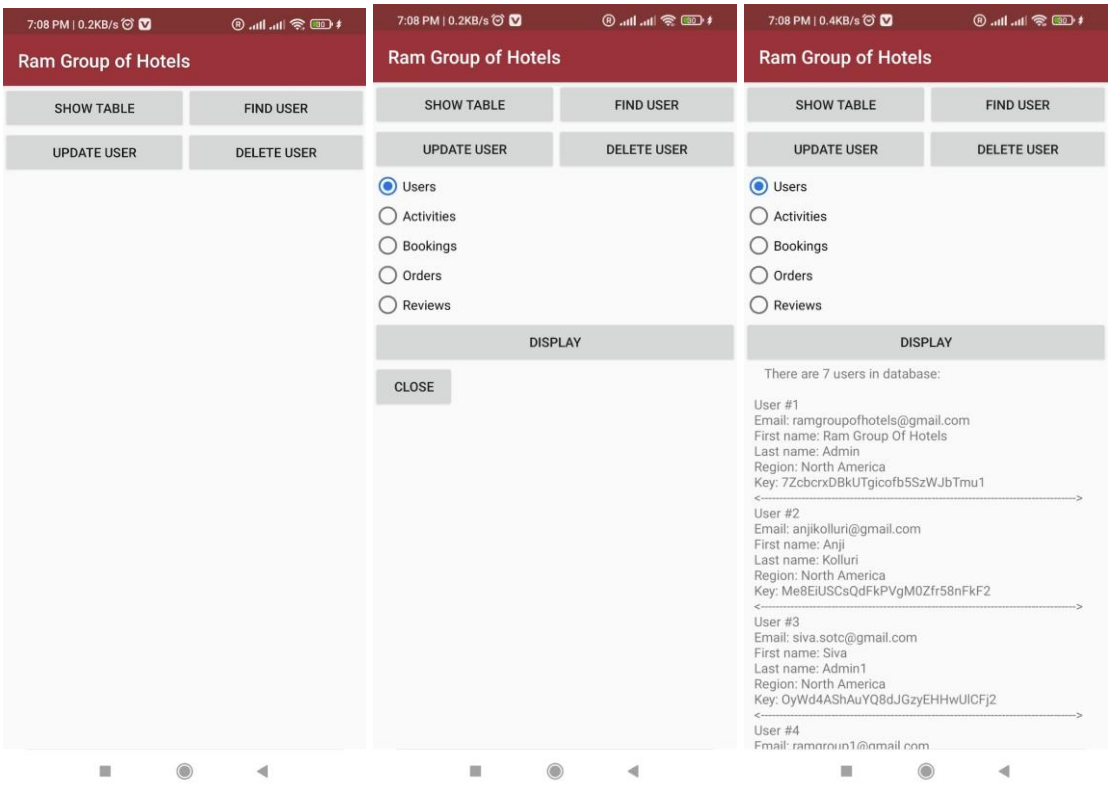

7:06 PM | 4.0KB/s
Ram Group of Hotels
Available Rooms
Click on 'Book' to Proceed
Mini Suite
$120
1 queen bed
This studio has a mini-bar and air conditioning.
BOOK
Mater suite
$180
1 king bed and 1 sofa bed
This is our superior suite
Payment Successful
BOOK
Villa suite
$220
1 sofa bed and 2 queen beds
This suite is a perfect fit for a family


7:07 PM | 0.3KB/s
Kayaking
Kayaking remains a popular pastime in sports-centric Vancouver, with locations offering either urban views in
BOOK
Skiing and Snowboarding
Fans of winter sports will rejoice at the choice of three local mountains just north of Vancouver's city center,


7:07 PM | 0.5KB/s
Ram Group of Hotels
Breakfast
MORNING BOOSTER $15.75
one free range poached egg, toast, Vista seasonal smoothie
BOOK
Lunch and Dinner


7:07 PM | 0.9KB/s
Ram Group of Hotels
Invoice
Siva Admin1
Order Id Date Service Charges
R1936 22/11/2022 Comfort Suite $268.8
R4955 16/11/2022 Comfort Suite $268.8
R5312 17/11/2022 Comfort Suite $268.8
R6588 17/11/2022 Comfort Suite $268.8
R6128 31/10/2022 Comfort Suite $268.8
R4597 17/11/2022 Comfort Suite $268.8
R1139 24/11/2022 Kings Suite $403.2
R9141 18/11/2022 Family Suite $739.2
R4964 20/11/2022 Comfort Suite $1344.0
Grand Total till this Date: $14,379.60


7:07 PM | 0.2KB/s
Ram Group of Hotels
Rank and Review
Room
Write your review...
SEND REVIEW
Your rating is 4.7


7:08 PM | 0.2KB/s
Ram Group of Hotels
SHOW TABLE
FIND USER
UPDATE USER
DELETE USER
7:08 PM | 0.2KB/s
Ram Group of Hotels
SHOW TABLE
FIND USER
UPDATE USER
DELETE USER
Users
Activities
Bookings
Orders
Reviews
DISPLAY
CLOSE
7:08 PM | 0.4KB/s
Ram Group of Hotels
SHOW TABLE
FIND USER
UPDATE USER
DELETE USER
Users
Activities
Bookings
Orders
Reviews
DISPLAY
There are 7 users in database:
User #1
Email: ramgroupofhotels@gmail.com
First name: Ram Group Of Hotels
Last name: Admin
Region: North America
Key: 7ZcbcrxDBkUTgicofb5SzWJbTmu1
User #2
Email: anjikolluri@gmail.com
First name: Anji
Last name: Kolluri
Region: North America
Key: Me8EiUSCsQdFkPVgM0Zfr58nFkF2
User #3
Email: siva.sotc@gmail.com
First name: Siva
Last name: Admin1
Region: North America
Key: OyWd4AShAuYQ8dJGzyEHHwUlCFj2
User #4

References:

https://developer.android.com/develop/ui/views/layout/declaring-layout Layouts in Android Studio 2021.3.1

https://developer.android.com/reference/android/view/View Different Views in Android Studio 2021.3.1

https://developer.android.com/guide/components/intents-filters Intents and intent filters in Android Studio 2021.3.1

https://developer.android.com/guide/app-compatibility App Compatibility in Android Studio 2021.3.1

https://developer.android.com/guide/fragments Fragments and Fragment layouts in Android Studio 2021.3.1

https://www.geeksforgeeks.org/check-email-address-valid-not-java/. GeeksforGeeks. Check if email address valid or not in Java.

https://stackoverflow.com/questions/45686276/how-to-use-firebase-email-login-with-a-fragment#. Stack Overflow. How to use Firebase Email login with a fragment.

https://developer.android.com/guide/topics/ui/notifiers/notifications Notifications Overview.

https://stackoverflow.com/questions/40000899/how-to-link-between-authenticated-users-and-database-in-firebase. How to link between Authenticated users and Database in Firebase.

https://www.androidhive.info/2016/10/android-working-with-firebase-realtime-database/ Android working with Firebase.

https://androidexample365.com/a-custom-android-rating-view-with-interactive-smiles/. YuganshT79. A custom Android Rating view with Interactive Smiles.

http://www.androhub.com/login-signup-and-forgot-password-screen-design-android/. Dr Droid. Login, SignUp and Forgot Password Screen Design Android.