# CHECKINNMATE

## A HOTEL MANAGEMENT SYSTEM

UCS2313 – Object Oriented Programming Lab

PROJECT REPORT

Submitted By


Vidisha Desai 3122 22 5001 154

Vijay Srinivas K 3122 22 5001 158

Vishal Sairam 3122 22 5001 163



Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Kalavakkam – 603110


December 2023

# TABLE OF CONTENTS

# PROBLEM STATEMENT

The Hotel Management System is a comprehensive software solution designed to streamline and optimize the operations of a modern hotel. This system provides a user-friendly interface for hotel staff, administrators, and guests, facilitating efficient management of various tasks related to reservations, billing, facilities, and customer interactions.

# MOTIVATION FOR THE PROBLEM

Our motivation for the problem lies strongly in the fact that this topic involves OOPS concepts in the various aspects of its working. Whether it be the Rooms hierarchy, Amenity hierarchy, or the User hierarchy, there is always an element of Object Oriented Principles in it. The problem is also something which is relevant in the real world, where many hotels require an application for their management. We believe that pursuing this project would enhance not only our knowledge of OOPs concepts but also our general coding skills and understanding of how to solve real world problems. Also working with a front end enhances our understanding of how to bring code to visual life as well. Overall, this project is something that we as a group intended to learn from, and having completed it, can certainly say that it has done so for sure.

# SCOPE

The Hotel Management System has a broad scope, encompassing various aspects of hotel operations, guest management, and amenities. The key components and their scope are :

- User Management:

  The system manages different user roles, such as administrators, users and guests.
  It ensures secure and efficient account management for each user.

- Reservation System:

  Provides a streamlined process for guests to make reservations.
  Tracks and manages room reservations, including check-in and check-out procedures.

- Billing System:

  Handles billing for room charges and additional services. Supports smooth and transparent billing procedures for guests.

- Room Management:

  Defines a flexible room hierarchy. Optimizes room management processes, such as assigning guests, checking room occupancy, and handling room-specific features.

- Feedback System:

  Includes a feedback system for guests to provide insights. Demonstrates a commitment to continuous improvement based on user and guest feedback.

- Facility Management:

  Manages various hotel facilities and amenities. Ensures efficient utilization of amenities, such as gyms, swimming pools, game rooms, and mini-bars.

- Guest Experience:

  Prioritizes guest satisfaction through personalized services and efficient handling of reservations and requests. Aims to enhance the overall guest experience during their stay.

# LIMITATIONS

- One major drawback of the system is that, the front-end is made using Swing of Java. Swing, the Java GUI toolkit, has drawbacks including variable look and feel across operating systems, heavyweight components impacting performance, a steep learning curve, limited support for modern UI features, and challenges in integrating with JavaFX. Its outdated appearance, performance concerns, and lack of native mobile support also make it less suitable for contemporary desktop and mobile application development compared to newer frameworks.

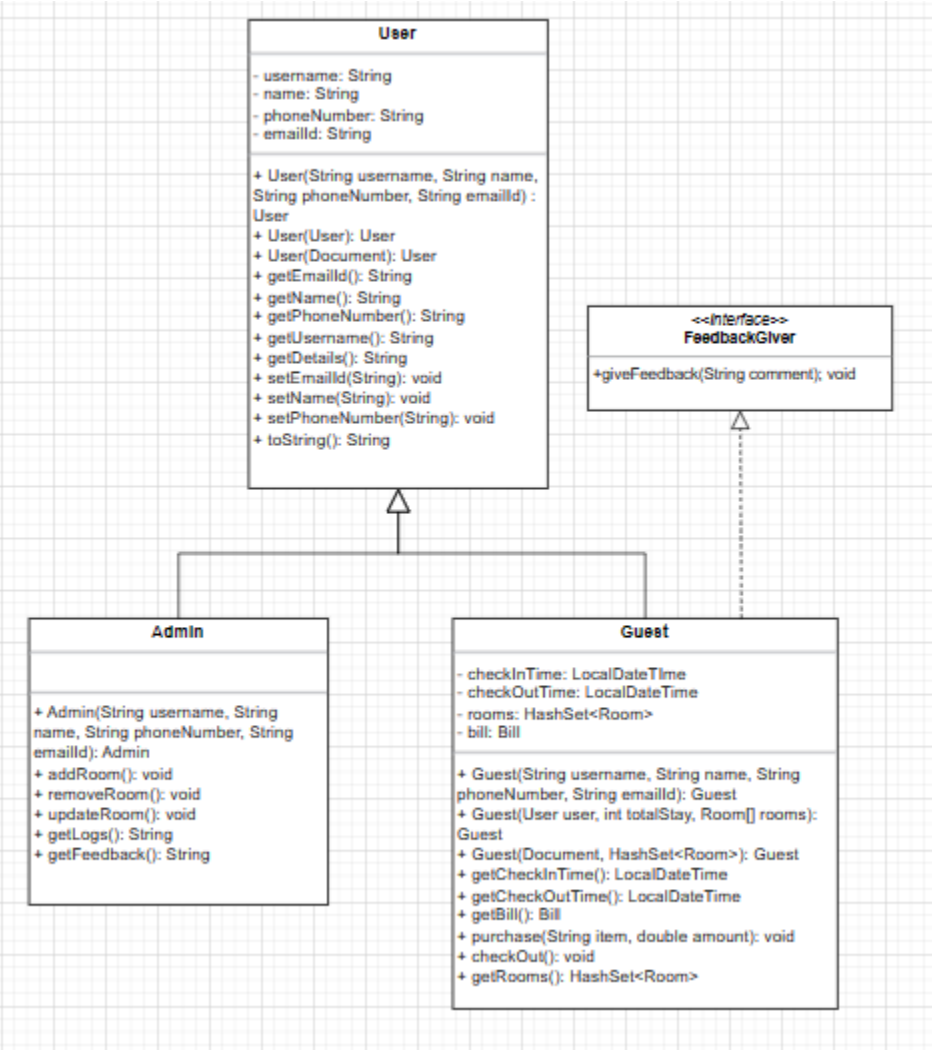
- The system is made to be locally accessed and used. The absence of portability or remote access in a system results in limited accessibility, reduced flexibility, and a dependence on specific physical infrastructure. This can lead to increased

downtime and hinders the adaptability of the system to different devices and environments.

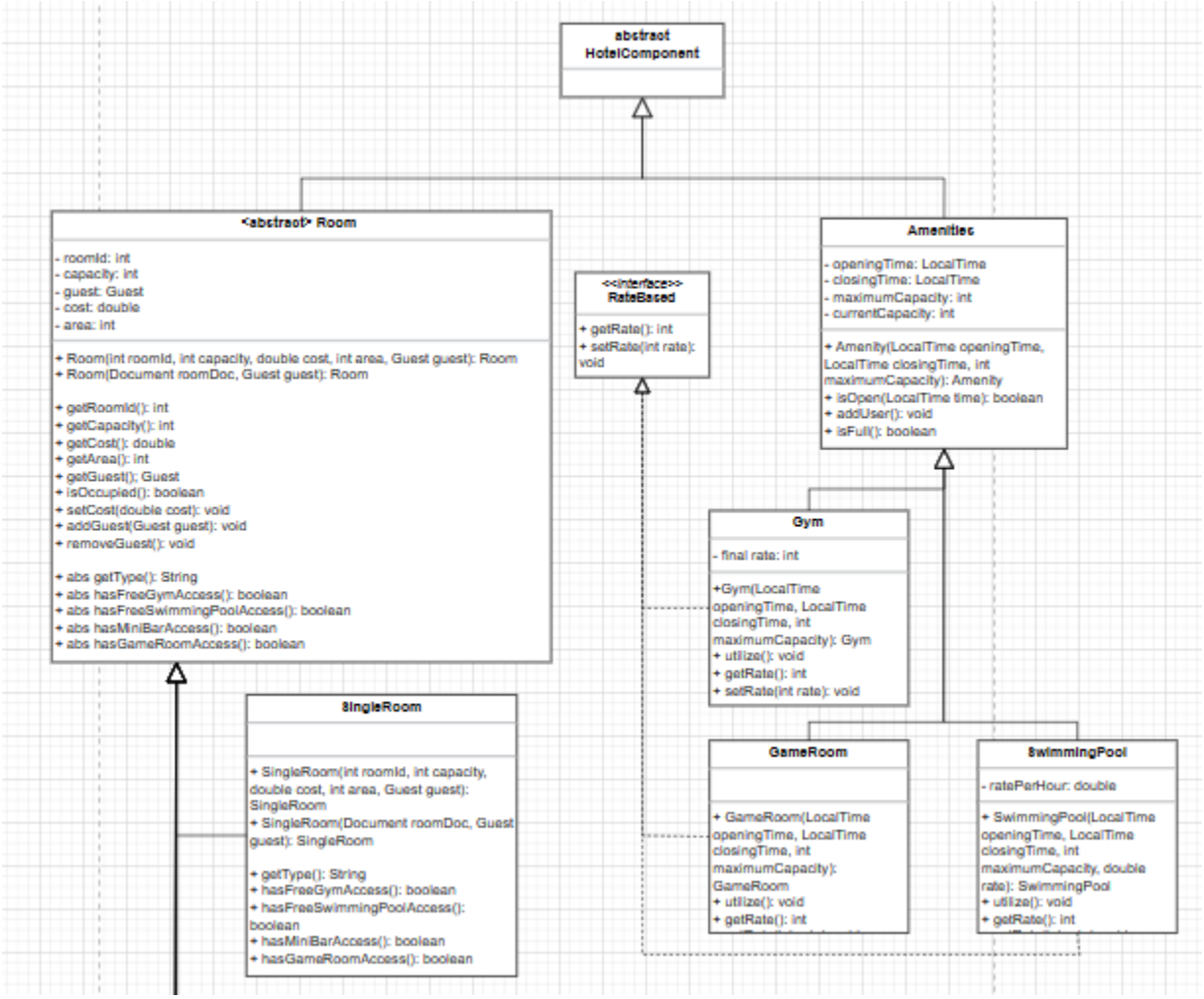
- The system uses a local MongoDB database as NoSQL database. However, it is slow. Slow server access to a database results in impaired system performance, increased latency, user frustration, and challenges in scalability. It can lead to potential data inconsistencies, impact real-time applications, and increase dependency on network stability. Meeting service level agreements becomes challenging, placing a heavier workload on users and potentially affecting business operations.

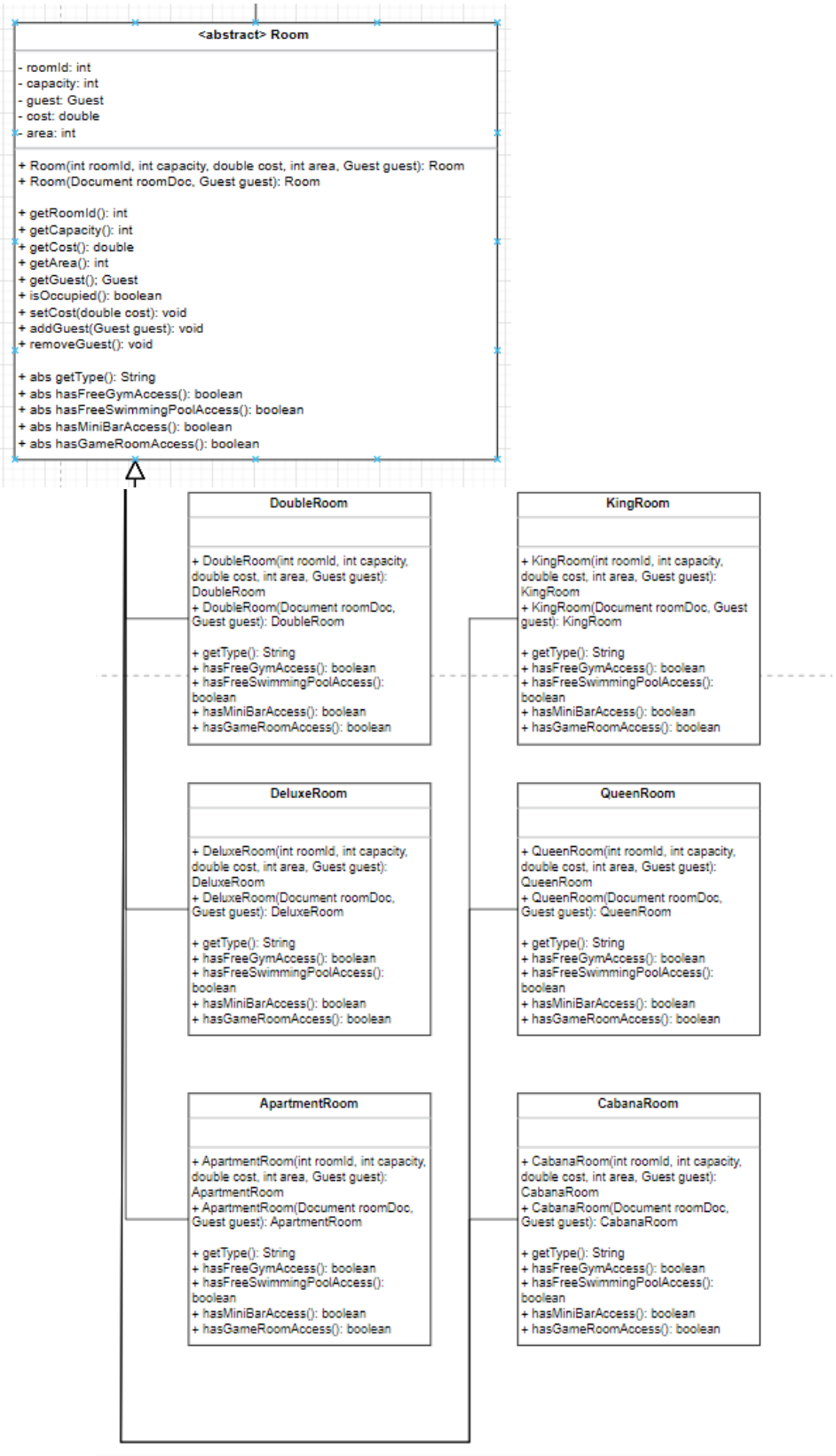
# CLASS DIAGRAM :

## USER HIERARCHY

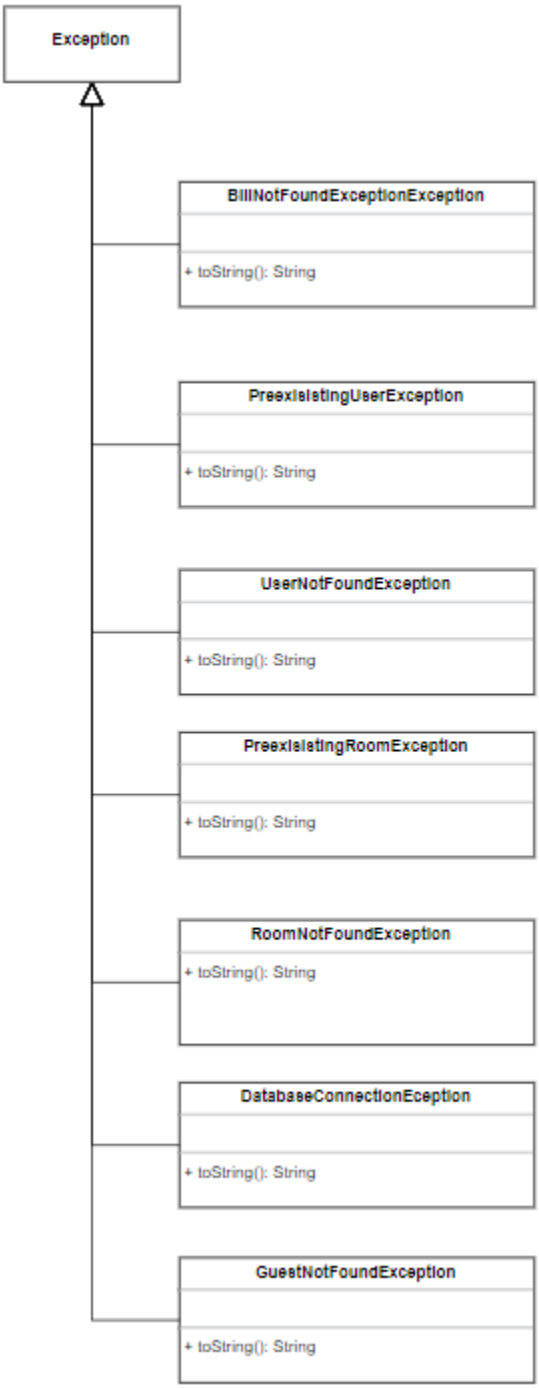


# HOTEL HIERARCHY

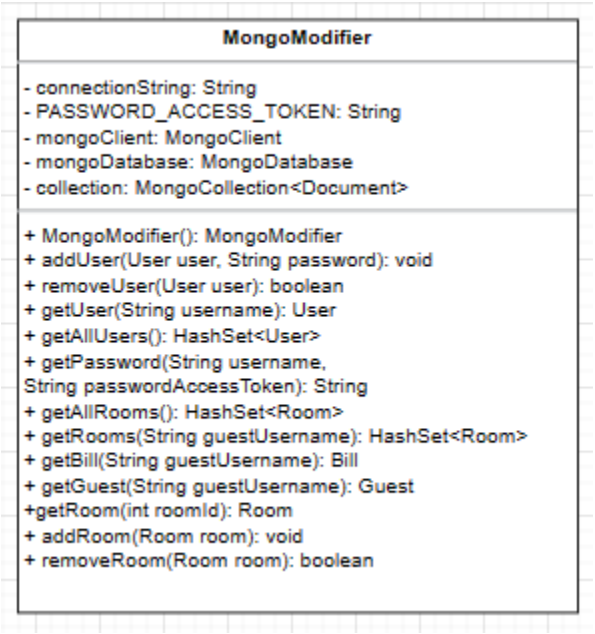


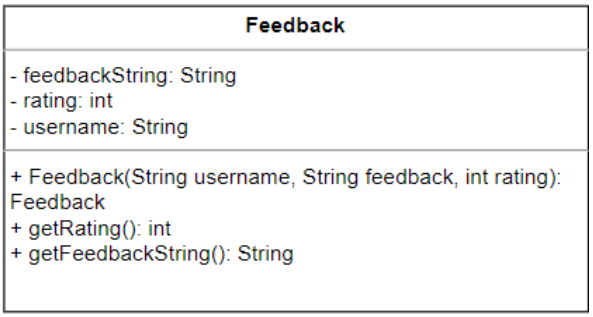

<abstract> Room
- roomId: int
- capacity: int
- guest: Guest
- cost: double
- area: int
+ Room(int roomId, int capacity, double cost, int area, Guest guest): Room
+ Room(Document roomDoc, Guest guest): Room
+ getRoomId(): int
+ getCapacity(): int
+ getCost(): double
+ getArea(): int
+ getGuest(); Guest
+ isOccupied(): boolean
+ setCost(double cost): void
+ addGuest(Guest guest): void
+ removeGuest(): void
+ abs getType(): String
+ abs hasFreeGymAccess(): boolean
+ abs hasFreeSwimmingPoolAccess(): boolean
+ abs hasMiniBarAccess(): boolean
+ abs hasGameRoomAccess(): boolean
DoubleRoom
+ DoubleRoom(int roomId, int capacity, double cost, int area, Guest guest): DoubleRoom
+ DoubleRoom(Document roomDoc, Guest guest): DoubleRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean
KingRoom
+ KingRoom(int roomId, int capacity, double cost, int area, Guest guest): KingRoom
+ KingRoom(Document roomDoc, Guest guest): KingRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean
DeluxeRoom
+ DeluxeRoom(int roomId, int capacity, double cost, int area, Guest guest): DeluxeRoom
+ DeluxeRoom(Document roomDoc, Guest guest): DeluxeRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean
QueenRoom
+ QueenRoom(int roomId, int capacity, double cost, int area, Guest guest): QueenRoom
+ QueenRoom(Document roomDoc, Guest guest): QueenRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean
ApartmentRoom
+ ApartmentRoom(int roomId, int capacity, double cost, int area, Guest guest): ApartmentRoom
+ ApartmentRoom(Document roomDoc, Guest guest): ApartmentRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean
CabanaRoom
+ CabanaRoom(int roomId, int capacity, double cost, int area, Guest guest): CabanaRoom
+ CabanaRoom(Document roomDoc, Guest guest): CabanaRoom
+ getType(): String
+ hasFreeGymAccess(): boolean
+ hasFreeSwimmingPoolAccess(): boolean
+ hasMiniBarAccess(): boolean
+ hasGameRoomAccess(): boolean

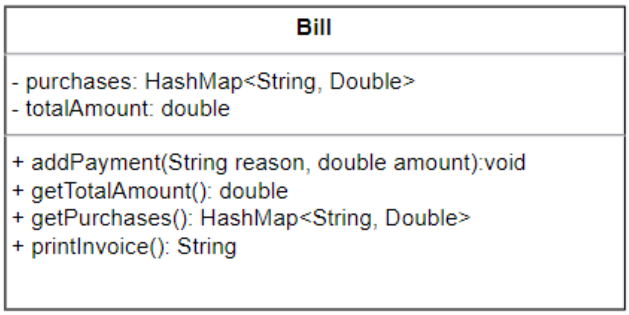
# EXCEPTION HIERARCHY



# OTHER CLASSES



# MODULES SPLIT UP :

The modules have been splitted in the following way:

- Login Module:
    - Consists of the Login and SignUp pages and their related functionalities
- User Module
    - Consists of the functionalities specific to the user such as Search rooms with constraints, booking rooms, viewing their profile, etc.
- Guest Module
    - Consists of the functionalities specific to the Guest such as Utilizing Amenities like Gym, Swimming Pool and Game Room,  Checking Out and Giving Feedback
- Admin Module
    - Consists of the functionalities specific to the Admin such as

# IMPLEMENTATION:

## PACKAGE ACCESSORS

Java

```java
//PACKAGE ACCESSORS :

// ADMIN CLASS:

package accessors;

public class Admin extends User {
    public Admin(String username, String name, String phoneNumber,
String emailId) {
        super(username, name, phoneNumber, emailId);
    }

    public void addRoom() {

    }
```

```
    public void removeRoom() {

    }

    public void updateRoom() {

    }

    public String getLogs() {
        return "";
    }
    public String getFeedback() {
        return "";
    }
}

// GUEST CLASS:

package accessors;

import com.mongodb.client.MongoDatabase;
import database.MongoModifier;
import feedbackcomponents.Feedback;
import feedbackcomponents.FeedbackGiver;
import hotelcomponents.rooms.Room;
import hotelexceptions.UserNotFoundException;
import org.bson.Document;
import payments.Bill;

import java.time.*;
import java.util.Arrays;
import java.util.HashSet;

public class Guest extends User implements FeedbackGiver,
Comparable<Guest> {
    private LocalDateTime checkInTime;
    private LocalDateTime checkOutTime;
    private HashSet<Room> rooms;
    private Bill bill;
```

```
    public Guest(User user, LocalDate checkInDate, LocalDate
checkOutDate, HashSet<Room> rooms) throws UserNotFoundException {
        super(user);
        checkInTime = checkInDate.atTime(14, 0, 0);
        checkOutTime = checkOutDate.atTime(12, 0, 0);; //Will come up
with formula later
        bill = (new MongoModifier()).getBill(getUsername());
        this.rooms = rooms;
    }
    public Guest(Document guestDoc, HashSet<Room> rooms) throws
UserNotFoundException {
        super((new
MongoModifier()).getUser(guestDoc.getString("originalUsername")));
        checkInTime =
LocalDateTime.parse(guestDoc.getString("checkInTime"));
        checkOutTime =
LocalDateTime.parse(guestDoc.getString("checkOutTime"));
        bill = (new MongoModifier()).getBill(getUsername());
        this.rooms = rooms;
    }
    public Guest(Document guestDoc) throws UserNotFoundException {
        super((new
MongoModifier()).getUser(guestDoc.getString("originalUsername")));
        checkInTime =
LocalDateTime.parse(guestDoc.getString("checkInTime"));
        checkOutTime =
LocalDateTime.parse(guestDoc.getString("checkOutTime"));
        bill = (new MongoModifier()).getBill(getUsername());
        this.rooms = null;
    }

    public LocalDateTime getCheckInTime() {
        return checkInTime;
    }

    public LocalDateTime getCheckOutTime() {
        return checkOutTime;
    }

    public Bill getBill() {
```

```
        return bill;
    }
    public void purchase(String item, double amount) {
        bill.addPayment(item, amount);
    }
    public void checkOut() {

    }

    public HashSet<Room> getRooms() {
        return rooms;
    }

    public Room getMostValubleRoom() {
        return new Room() {
            @Override
            public String getType() {
                return "MostValuble";
            }

            @Override
            public boolean hasFreeGymAccess() {
                boolean b = false;
                for (Room room : rooms) {
                    b = b | room.hasFreeGymAccess();
                }
                return b;
            }

            @Override
            public boolean hasFreeSwimmingPoolAccess() {
                boolean b = false;
                for (Room room : rooms) {
                    b = b | room.hasFreeSwimmingPoolAccess();
                }
                return b;
            }

            @Override
            public boolean hasMiniBarAccess() {
```

```
                boolean b = false;
                for (Room room : rooms) {
                    b = b | room.hasMiniBarAccess();
                }
                return b;
            }

            @Override
            public boolean hasFreeGameRoomAccess() {
                boolean b = false;
                for (Room room : rooms) {
                    b = b | room.hasFreeGameRoomAccess();
                }
                return b;
            }

            @Override
            public String getDescription() {
                return null;
            }
        };
    }
    @Override
    public int compareTo(Guest guest) {
        return (int) Duration.between(checkOutTime,
guest.checkOutTime).getSeconds();
    }

    @Override
    public void giveFeedback(Feedback feedback) {
        MongoModifier mongo = new MongoModifier();
        mongo.addFeedback(feedback);
    }
}


// USER CLASS:

package accessors;
import feedbackcomponents.FeedbackGiver;
```

```
import org.bson.Document;

public class User {
    private String username;
    private String name;
    private String phoneNumber;
    private String emailId;

    public User(String username, String name, String phoneNumber, String
emailId) {
        this.username = username;
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.emailId = emailId;
    }
    public User(User user) {
        this.username = user.username;
        this.name = user.name;
        this.phoneNumber = user.phoneNumber;
        this.emailId = user.emailId;
    }
    public User(Document doc) {
        this.username = (String)doc.get("username");
        this.name = (String)doc.get("name");
        this.phoneNumber = (String)doc.get("phonenumber");
        this.emailId = (String)doc.get("email");
    }

    public String getEmailId() {
        return emailId;
    }

    public String getName() {
        return name;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public String getUsername() {
        return username;
    }
```

```
    @Override
    public String toString() {
        return "User{" +
                "username='" + username + '\'' +
                ", name='" + name + '\'' +
                ", phoneNumber='" + phoneNumber + '\'' +
                ", emailId='" + emailId + '\'' +
                '}';
    }

}
```

## PACKAGE DATABASE:

Java

```
//PACKAGE DATABASE:

//MONGOMODIFIER CLASS:

package database;
import accessors.Guest;
import accessors.User;
import com.mongodb.ConnectionString;
import com.mongodb.MongoClientSettings;
import com.mongodb.client.*;
import com.mongodb.client.model.Filters;
import feedbackcomponents.Feedback;
import hotelcomponents.HotelComponent;
import hotelcomponents.rooms.*;
import hotelexceptions.*;
import org.bson.Document;
import payments.Bill;
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.HashSet;
```

```
public class MongoModifier {
    private static final String connectionString =
"mongodb+srv://HotelUser:4hzn3PHUVtyFsCHP@checkinnmatecluster.9vr7fw9.mo
ngodb.net/?retryWrites=true&w=majority";
    private static final String PASSWORD_ACCESS_TOKEN =
"awZYbHxLl4WykOhSiujjJm7U9VuEYx2X";
    private MongoClient mongoClient;
    private MongoDatabase mongoUserDatabase, mongoHotelDatabase,
mongoAdminDatabase;
    private MongoCollection<Document> usersCollection, logsCollection,
feedbackCollection, roomsCollection, guestsCollection, billsCollection;

    public MongoModifier() {
        /*MongoClientSettings settings = MongoClientSettings.builder()
                .applyToConnectionPoolSettings(builder ->
                        builder.maxSize(1) // Adjust this value based on
your needs
                )
                .applyConnectionString(new
ConnectionString(connectionString))
                // other settings...
                .build();
        mongoClient = MongoClients.create(settings);*/
        mongoClient = MongoClients.create("mongodb://localhost:27017");
        mongoUserDatabase = mongoClient.getDatabase("Userbase");;
        mongoHotelDatabase = mongoClient.getDatabase("Hotelbase");
        mongoAdminDatabase = mongoClient.getDatabase("Adminbase");
    }

    public void addUser(User user, String password) throws
PreexisistingUserException {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Users");
        try {
            getUser(user.getUsername());
            throw new PreexisistingUserException();
        }
        catch (UserNotFoundException e) {
```

```
            Document document = new Document("username",
user.getUsername())
                    .append("name", user.getName())
                    .append("phonenumber", user.getPhoneNumber())
                    .append("email", user.getEmailId())
                    .append("password", password);
            collection.insertOne(document);
        }
    }
    public boolean removeUser(User user) {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Users");
        try {
            getUser(user.getUsername());
            collection.deleteOne(Filters.eq("username",
user.getUsername()));
            return true;
        }
        catch (UserNotFoundException e) {
            return false;
        }
    }
    public User getUser(String username) throws UserNotFoundException {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Users");
        Document query = new Document("username", username);
        Document result = collection.find(query).first();
        if (result == null)
            throw new UserNotFoundException();
        return new User(result);
    }

    public HashSet<User> getAllUsers() {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Users");
        MongoCursor<Document> result = collection.find().iterator();
        HashSet<User> users = new HashSet<>();
```

```
        while(result.hasNext()) {
            users.add(new User(result.next()));
        }
        return users;
    }
    public String getPassword(String username, String
passwordAccessToken) throws IllegalPasswordAccessException,
UserNotFoundException {
        if (!(passwordAccessToken.equals(PASSWORD_ACCESS_TOKEN)))
            throw new IllegalPasswordAccessException();
        else {
            MongoDatabase database = mongoUserDatabase;
            MongoCollection<Document> collection =
database.getCollection("Users");
            Document query = new Document("username", username);
            Document result = collection.find(query).first();
            if (result == null)
                throw new UserNotFoundException();
            return result.getString("password");
        }
    }
    public HashSet<Room> getAllRooms() {
        return getRooms("");
    }

    public HashSet<Room> getRooms(String guestUsername) {
        try {
            MongoDatabase database = mongoHotelDatabase;
            MongoCollection<Document> collection =
database.getCollection("Rooms");
            MongoCursor<Document> results;
            if (guestUsername.equals(""))
                results = collection.find().iterator();
            else
                results = collection.find(new Document("guest",
guestUsername)).iterator();
            HashSet<Room> rooms = new HashSet<>();
            while(results.hasNext()) {
                Document roomDoc = results.next();
```

```
                //System.out.println("HELLO THEREEEEEEE: " +
roomDoc.get("roomId"));
                try {
                    rooms.add((Room)
Class.forName("hotelcomponents.rooms." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc,
getRoomlessGuest(roomDoc.getString("guest"))));
                }
                catch (GuestNotFoundException e) {
                    System.out.println(rooms);
                    rooms.add((Room)
Class.forName("hotelcomponents.rooms." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc, null));
                }
            }
            return rooms;
        }
        catch (ClassNotFoundException | NoSuchMethodException e) {
            e.printStackTrace();
            System.out.println("The given user was not found");
        }
        catch (IllegalAccessException | InstantiationException |
InvocationTargetException e) {
            e.printStackTrace();
        }

        return null;
    }

    public Bill getBill(String guestUsername) {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Bills");
        Document result = collection.find(new Document("guest",
guestUsername)).first();
        try {
            if (result == null)
                throw new BillNotFoundException();
```

```
            Bill bill = new Bill();
            for (String key : result.keySet()) {
                if (!(key.equals("_id") || key.equals("guest"))) {
                    bill.addPayment(key, result.getDouble(key));
                }
            }
            return bill;
        }
        catch (BillNotFoundException e) {
            createBill(guestUsername);
            return new Bill();
        }

    }

    public Guest getGuest(String guestUsername) throws
GuestNotFoundException {
        MongoDatabase database = mongoUserDatabase;
        MongoCollection<Document> collection =
database.getCollection("Guests");
        Document result = collection.find(new
Document("originalUsername", guestUsername)).first();
        if (result == null)
            throw new GuestNotFoundException();
        Guest guest = null;
        try {
            HashSet<Room> rooms = getRooms(guestUsername);
            guest = new Guest(result, rooms);
            for (Room room : rooms)
                room.addGuest(guest);
        }
        catch (UserNotFoundException e) {
            System.out.println("The given user was not found");
        }
        return guest;
    }

    public Guest getRoomlessGuest(String guestUsername) throws
GuestNotFoundException{
        MongoDatabase database = mongoUserDatabase;
```

```
        MongoCollection<Document> collection =
database.getCollection("Guests");
        Document result = collection.find(new
Document("originalUsername", guestUsername)).first();
        if (result == null)
            throw new GuestNotFoundException();
        Guest guest = null;
        try {
            guest = new Guest(result, null);
        }
        catch (UserNotFoundException e) {
            System.out.println("The given user was not found");
        }
        return guest;
    }

    //

    public void addGuest(Guest guest){
        MongoDatabase database = mongoClient.getDatabase("Userbase");
        MongoCollection<Document> collection =
database.getCollection("Guests");

        Document document = new Document("checkInTime",
guest.getCheckInTime().toString())
                .append("checkOutTime",
guest.getCheckOutTime().toString())
                .append("originalUsername", guest.getUsername());
        collection.insertOne(document);

    }
    public boolean removeGuest(Guest guest){
        MongoDatabase database = mongoClient.getDatabase("Userbase");
        MongoCollection<Document> collection =
database.getCollection("Guests");

        Document query = new Document("originalUsername",
guest.getUsername());
        Document result = collection.find(query).first();
        if(result == null){
```

```
            return false;
        }
        collection.deleteOne(query);
        return true;
    }

    public void createBill(String username) {
        MongoDatabase database = mongoClient.getDatabase("Userbase");
        MongoCollection<Document> collection =
database.getCollection("Bills");
        collection.insertOne(new Document().append("guest", username));
    }
    public void addBill(Guest guest){
        MongoDatabase database = mongoClient.getDatabase("Userbase");
        MongoCollection<Document> collection =
database.getCollection("Bills");

        collection.findOneAndDelete(new Document("guest",
guest.getUsername()));

        Bill bill = guest.getBill();
        Document document = new Document("guest", guest.getUsername());
        for(String key: bill.getPurchases().keySet()){
            document.append(key, bill.getPurchases().get(key));
        }
        collection.insertOne(document);
    }

    public boolean removeBill(Guest guest){
        MongoDatabase database = mongoClient.getDatabase("Userbase");
        MongoCollection<Document> collection =
database.getCollection("Bills");

        Document query = new Document("guest", guest.getUsername());
        Document result = collection.find(query).first();
        if(result == null){
            return false;
        }
        collection.deleteOne(query);
        return true;
```

```
    }
    public void updateBill(Guest guest){
        removeBill(guest);
        addBill(guest);
    }
    public Room getRoom(int roomID) throws RoomNotFoundException {
        MongoCollection<Document> collection =
mongoHotelDatabase.getCollection("Rooms");

        try{
            Document query = new Document("roomID", roomID);
            Document result = collection.find(query).first();

            if (result == null){
                throw new RoomNotFoundException();
            }
            try {
                return (Room) Class.forName("hotelcomponents.rooms." +
result.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(result,
getRoomlessGuest(result.getString("guest")));
            }
            catch (GuestNotFoundException ex) {
                return (Room) Class.forName("hotelcomponents.rooms." +
result.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(result, null);
            }
        } catch (ClassNotFoundException | InvocationTargetException |
InstantiationException | IllegalAccessException | NoSuchMethodException
e) {
            throw new RuntimeException(e);
        }
    }
    public void addRoom(Room room) throws PreexisistingRoomException {
        MongoDatabase database = mongoClient.getDatabase("Hotelbase");
        MongoCollection<Document> collection =
database.getCollection("Rooms");

        try {
            getRoom(room.getRoomId());
```

```
            throw new PreexisistingRoomException();
        }
        catch (RoomNotFoundException e) {
            try {
                Document document = new Document("roomId",
room.getRoomId())
                        .append("capacity", room.getCapacity())
                        .append("cost", (int)room.getCost())
                        .append("area", room.getArea())
                        .append("roomType", room.getType())
                        .append("guest", room.getGuest().getUsername());
                collection.insertOne(document);
            }
            catch (NullPointerException ex) {
                Document document = new Document("roomId",
room.getRoomId())
                        .append("capacity", room.getCapacity())
                        .append("cost", (int)room.getCost())
                        .append("area", room.getArea())
                        .append("roomType", room.getType())
                        .append("guest", "");
                collection.insertOne(document);
            }
        }
    }

    public boolean removeRoom(Room room){
        MongoDatabase database = mongoClient.getDatabase("Hotelbase");
        MongoCollection<Document> collection =
database.getCollection("Rooms");

        Document query = new Document("roomId", room.getRoomId());
        Document result = collection.find(query).first();
        if(result == null){
            System.out.println(query);
            return false;
        }
        collection.deleteOne(query);
        return true;
    }
```

```
    void updateRoom(Room room) {
        removeRoom(room);
        try{
            addRoom(room);
        }
        catch(PreexisistingRoomException e){}
    }

    public HashSet<Room> getRoomsByType(String roomType){
        try {
            MongoDatabase database =
mongoClient.getDatabase("Hotelbase");
            MongoCollection<Document> collection =
database.getCollection("Rooms");
            MongoCursor<Document> results;
            if (roomType.equals(""))
                results = collection.find().iterator();
            else
                results = collection.find(new Document("roomType",
roomType)).iterator();
            HashSet<Room> rooms = new HashSet<>();
            while(results.hasNext()) {
                Document roomDoc = results.next();
                rooms.add((Room)Class.forName("hotelcomponents." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc, null));
            }
            return rooms;
        }
        catch (ClassNotFoundException | NoSuchMethodException e) {
            System.out.println("The given user was not found");
        }
        catch (IllegalAccessException | InstantiationException |
InvocationTargetException e) {
            e.printStackTrace();
        }
        return null;
    }
```

```
    public HashSet<Room> getRooms(int capacity){
        try {
            MongoDatabase database =
mongoClient.getDatabase("Hotelbase");
            MongoCollection<Document> collection =
database.getCollection("Rooms");
            MongoCursor<Document> results;
            results = collection.find(new Document("capacity",
capacity)).iterator();
            HashSet<Room> rooms = new HashSet<>();
            while(results.hasNext()) {
                Document roomDoc = results.next();
                rooms.add((Room)Class.forName("hotelcomponents." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc, null));
            }
            return rooms;
        }
        catch (ClassNotFoundException | NoSuchMethodException e) {
            System.out.println("The given user was not found");
        }
        catch (IllegalAccessException | InstantiationException |
InvocationTargetException e) {
            e.printStackTrace();
        }
        return null;
    }
    public HashSet<Room> getRooms(int min, int max){
        try {
            MongoDatabase database =
mongoClient.getDatabase("Hotelbase");
            MongoCollection<Document> collection =
database.getCollection("Rooms");
            MongoCursor<Document> results;
            results = collection.find().iterator();
            HashSet<Room> rooms = new HashSet<>();
            while(results.hasNext()) {
                Document roomDoc = results.next();
                int cap = (int)roomDoc.getInteger("capacity");
                if ((min <= cap) && (cap <= max))
```

```
                    rooms.add((Room)Class.forName("hotelcomponents." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc, null));
            }
            return rooms;
        }
        catch (ClassNotFoundException | NoSuchMethodException e) {
            System.out.println("The given user was not found");
        }
        catch (IllegalAccessException | InstantiationException |
InvocationTargetException e) {
            e.printStackTrace();
        }
        return null;
    }

    //

    public void addFeedback(Feedback feedback) {
        MongoDatabase database = mongoAdminDatabase;
        MongoCollection<Document> collection =
database.getCollection("Feedback");

        collection.insertOne(new Document()
                .append("username", feedback.getAssociatedUsername())
                .append("type", feedback.getType())
                .append("rating", feedback.getRating())
                .append("text", feedback.getFeedbackString())
                .append("date", feedback.getDate().toString()));
    }

    public ArrayList<Feedback> getFeedback(String targetType) {
        MongoDatabase database = mongoAdminDatabase;
        MongoCollection<Document> collection =
database.getCollection("Feedback");
        MongoCursor<Document> result = collection.find(new
Document("type", targetType)).iterator();
        ArrayList<Feedback> list = new ArrayList<>();

        while (result.hasNext()) {
```

```
            Document doc = result.next();
            list.add(new Feedback(doc));
        }

        return list;
    }

    public ArrayList<Feedback> getFeedback() {
        MongoDatabase database = mongoAdminDatabase;
        MongoCollection<Document> collection =
database.getCollection("Feedback");
        MongoCursor<Document> result = collection.find().iterator();
        ArrayList<Feedback> list = new ArrayList<>();

        while (result.hasNext()) {
            Document doc = result.next();
            list.add(new Feedback(doc));
        }

        return list;
    }

    public static void main(String[] args) {
        MongoModifier mod = new MongoModifier();
        /*try {
            mod.addUser(new User("vishalsairam01", "Vishal Sairam",
"vishalsairam01@gmail.com", "+91 9384851772"));
        }
        catch (PreexisistingUserException e) {
            System.out.println(e);
        }*/

        mod.removeUser(new User("vishalsairam", "Vishal Sairam",
"vishalsairam01@gmail.com", "+91 9384851772"));
    }
}
```

## PACKAGE FEEDBACK COMPONENTS :

Java

```java
//PACKAGE FEEDBACK COMPONENTS:

// FEEDBACK CLASS:

package feedbackcomponents;
import hotelcomponents.HotelComponent;
import org.bson.Document;
import java.time.LocalDate;

public class Feedback implements Comparable<Feedback> {
    private String feedbackString;
    private int rating;
    private String username;
    private LocalDate date;
    private String type;

    public Feedback(String username, String type, int rating, String
feedbackString) {
        this.username = username;
        this.feedbackString = feedbackString;
        this.rating = rating;
```

```
        this.type = type;
        date = LocalDate.now();
    }

    public Feedback(Document feedbackDoc) {
        this.username = feedbackDoc.getString("username");
        this.feedbackString = feedbackDoc.getString("text");
        this.rating = feedbackDoc.getInteger("rating");
        this.type = feedbackDoc.getString("type");
        date = LocalDate.parse(feedbackDoc.getString("date"));
    }

    public int getRating() {
        return rating;
    }

    public String getFeedbackString() {
        return feedbackString;
    }

    public String getAssociatedUsername() {
        return username;
    }
    public LocalDate getDate() {
        return date;
    }
    public String getType() {
        return type;
    }
    @Override
    public int compareTo(Feedback feedback) {
        return date.compareTo(feedback.date);
    }
}
```

**//FEEDBACK GIVER INTERFACE:**

```
package feedbackcomponents;

public interface FeedbackGiver {
    void giveFeedback(Feedback feedback);
}
```

```
//FEEDBACK MANAGER CLASS:

package feedbackcomponents;
import hotelcomponents.HotelComponent;
import java.util.ArrayList;
public class FeedbackManager<T extends HotelComponent> {
    /*private ArrayList<Feedback> feedbacks;

    public ArrayList<Feedback> getFeedBack() {

        if ();
    }

    public ArrayList<Feedback> getAllFeedback() {
        return null;
    }*/
}
```

## PACKAGE HOTEL COMPONENTS:

Java

```
// PACKAGE HOTEL COMPONENTS:

// PACKAGE ROOMS:

// APARTMENT ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class ApartmentRoom extends Room {

    public ApartmentRoom(int roomId, int capacity, double cost, int
area, Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public ApartmentRoom(Document roomDoc, Guest guest) {
```

```
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "ApartmentRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return true;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}


// CABANA ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class CabanaRoom extends Room {
```

```
    public CabanaRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public CabanaRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "CabanaRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return false;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return false;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}
```

```
// DELUXE ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class DeluxeRoom extends Room {

    public DeluxeRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public DeluxeRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "DeluxeRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return false;
    }

    @Override
```

```
    public  String getDescription(){
        return " ";
    }
}


// DOUBLE ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class DoubleRoom extends Room {

    public DoubleRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public DoubleRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "DoubleRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return false;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return false;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return false;
    }
```

```
    @Override
    public boolean hasFreeGameRoomAccess() {
        return false;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}


// ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public abstract class Room implements Comparable<Room> {
    private int roomId;
    private int capacity;
    private Guest guest;
    private double cost;
    private int area;

    public Room(int roomId, int capacity, double cost, int area, Guest
guest) {
        this.roomId = roomId;
        this.capacity = capacity;
        this.cost = cost;
        this.area = area;
        this.guest = guest;
    }

    public Room(Document roomDoc, Guest guest) {
        this.roomId = roomDoc.getInteger("roomId");
        this.capacity = roomDoc.getInteger("capacity");
        this.cost = roomDoc.getInteger("cost");
        this.area = roomDoc.getInteger("area");
        this.guest = guest;
    }
```

```
public Room() {

}

public int getRoomId() {
    return roomId;
}

public int getCapacity() {
    return capacity;
}

public double getCost() {
    return cost;
}

public int getArea() {
    return area;
}

public Guest getGuest() {
    return guest;
}

public boolean isOccupied() {
    if (guest == null)
        return false;
    return true;
}

public void setCost(double cost) {
    this.cost = cost;
}

public void addGuest(Guest guest) {
    this.guest = guest;
}

public void removeGuest() {
```

```
        this.guest = null;
    }

    @Override
    public int compareTo(Room r) {
        return roomId - r.roomId;
    }

    public abstract String getType();
    public abstract boolean hasFreeGymAccess();
    public abstract boolean hasFreeSwimmingPoolAccess();
    public abstract boolean hasMiniBarAccess();
    public abstract boolean hasFreeGameRoomAccess();

    public abstract String getDescription();

}

// SINGLE ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class SingleRoom extends Room {

    public SingleRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public SingleRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }

    @Override
    public String getType() {
        return "SingleRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
```

```
        return false;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return false;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return false;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return false;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}


// SUITE ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class SuiteRoom extends Room {

    public SuiteRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public SuiteRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
```

```
    public String getType() {
        return "SuiteRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return true;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}


// KING ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class KingRoom extends Room {

    public KingRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
```

```
        super(roomId, capacity, cost, area, guest);
    }
    public KingRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "KingRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return true;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
}


// QUEEN ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
```

```
import org.bson.Document;

public class QueenRoom extends Room {

    public QueenRoom(int roomId, int capacity, double cost, int area,
Guest guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public QueenRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "QueenRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return false;
    }

    @Override
    public  String getDescription(){
        return " ";
    }
```

```
}


// EXECUTIVE ROOM CLASS:
package hotelcomponents.rooms;
import accessors.Guest;
import org.bson.Document;

public class ExecutiveRoom extends Room {

    public ExecutiveRoom(int roomId, int capacity, double cost, int area, Guest
guest) {
        super(roomId, capacity, cost, area, guest);
    }
    public ExecutiveRoom(Document roomDoc, Guest guest) {
        super(roomDoc, guest);
    }
    @Override
    public String getType() {
        return "ExecutiveRoom";
    }

    @Override
    public boolean hasFreeGymAccess() {
        return true;
    }

    @Override
    public boolean hasFreeSwimmingPoolAccess() {
        return true;
    }

    @Override
    public boolean hasMiniBarAccess() {
        return true;
    }

    @Override
    public boolean hasFreeGameRoomAccess() {
        return true;
    }

    @Override
    public  String getDescription(){
```

```
        return " ";
    }
}
```

## PACKAGE HOTEL COMPONENTS:

Java

```java
//PACKAGE HOTEL COMPONENTS:

// AMENITY CLASS:
package hotelcomponents;

import accessors.Guest;
import accessors.User;

import java.time.LocalTime;

public abstract class Amenity extends HotelComponent {
    private LocalTime openingTime;
    private LocalTime closingTime;
    private int maximumCapacity;
    private int currentCapacity;

    public Amenity(LocalTime openingTime, LocalTime closingTime, int
maximumCapacity) {
        this.openingTime = openingTime;
        this.closingTime = closingTime;
        this.maximumCapacity = maximumCapacity;
    }


    public boolean isOpen(LocalTime time) {

        int t1 = time.compareTo(openingTime);

        int t2 = time.compareTo(closingTime);
```

```
        return (t1 > 0) && (t2 < 0);

    }
    public boolean addUser() {
        if(!isFull(currentCapacity)){
            currentCapacity++;
            return true;
        }

        return false ;
    }

    public LocalTime getOpeningTime() {
        return openingTime;
    }

    public LocalTime getClosingTime() {
        return closingTime;
    }

    // if added true, else false
    public boolean isFull(int currentCapacity) {
        return currentCapacity == maximumCapacity;    }
    public abstract void utilize(Guest guest,int hoursUsed);
    // modify the bill ( )
}

// GAME ROOM CLASS:
package hotelcomponents;

import accessors.Guest;
import database.MongoModifier;
import stringoperations.StringModifier;

import java.time.LocalTime;

public class GameRoom extends Amenity implements RateBased {
    private int ratePerHour;
```

```
    public GameRoom(LocalTime openingTime, LocalTime closingTime, int
maximumCapacity, int rate) {
        super(openingTime, closingTime, maximumCapacity);
        this.ratePerHour = rate;
    }

    @Override
    public void utilize(Guest guest, int hoursUsed) {
        MongoModifier mongo = new MongoModifier();

guest.purchase(StringModifier.addSpace(getClass().getSimpleName()) + "
x" + hoursUsed + " hours", getRate() * hoursUsed);
        mongo.addBill(guest);
    }


    @Override
    public int getRate() {
        return this.ratePerHour;
    }

    @Override
    public void setRate(int rate) {
        this.ratePerHour = rate;
    }
}

// GYM CLASS:
package hotelcomponents;

import accessors.Guest;
import database.MongoModifier;
import hotelexceptions.GuestNotFoundException;
import screencomponents.pages.UtilizeAmenitiesPagePanel;
import stringoperations.StringModifier;

import java.awt.*;
import java.time.LocalTime;

public class Gym extends Amenity implements RateBased{
```

```
    private final int rate = 250;
    private int ratePerHour;
    public Gym(LocalTime openingTime, LocalTime closingTime, int
maximumCapacity, int rate) {
        super(openingTime, closingTime, maximumCapacity);
        this.ratePerHour = rate;
    }

    @Override
    public void utilize(Guest guest, int hoursUsed) {
        MongoModifier mongo = new MongoModifier();

guest.purchase(StringModifier.addSpace(getClass().getSimpleName()) + "
x" + hoursUsed + " hours", getRate() * hoursUsed);
        mongo.addBill(guest);
    }

    @Override
    public int getRate() {
        return ratePerHour ;
    }

    @Override
    public void setRate(int rate) {
        this.ratePerHour = rate;
    }
}
// HOTEL CLASS:
package hotelcomponents;

import hotelcomponents.rooms.Room;

import java.util.HashSet;

public class Hotel {
    private Restaurant res;
    private Gym gym;
    private HashSet<Room> rooms;
}

// HOTELCOMPONENT CLASS:
```

```java
package hotelcomponents;
public abstract class HotelComponent {
}

// RATE BASED INTERFACE:
package hotelcomponents;
public interface RateBased {
    int getRate();
    void setRate(int rate);
}
// RESTAURANT CLASS:
package hotelcomponents;

import accessors.Guest;
import database.MongoModifier;

import java.time.LocalTime;
import java.util.HashMap;

public class Restaurant extends Amenity {
    private HashMap<String, Integer> menu;

    public Restaurant(LocalTime openingTime, LocalTime closingTime, int
maximumCapacity) {
        super(openingTime, closingTime, maximumCapacity);
        setMenu();
    }

    private void setMenu() {
        menu = new HashMap<>();
        menu.put("Paneer Butter Masala",250);
        menu.put("Dal Tadka",200);
        menu.put("Malai Kofta",250);
        menu.put("Roti",50);
        menu.put("Naan",70);
        menu.put("Idlis",32);
        menu.put("Dosa",65);
        menu.put("Poori masala",80);
        menu.put("Parotta",55);
        menu.put("Fried Rice",120);
        menu.put("Pav Bhaji",90);

    }
```

```
    public void purchaseFoodItems(Guest guest, HashMap<String,Integer> orders){
        for (String key : orders.keySet()){
            guest.purchase (key, orders.get(key));
        }
    }
    public HashMap<String,Integer> getMenu(){
        return menu;
    }
    @Override
    public void utilize(Guest guest, int hoursUsed) {
        if (hoursUsed!=0){
            guest.purchase ("RestaurantServices x " + hoursUsed ,hoursUsed);
        }
        MongoModifier mongo = new MongoModifier();
       // mongo.updateBill(guest);
    }
}
// SWIMMINGPOOL CLASS:
package hotelcomponents;

import accessors.Guest;
import database.MongoModifier;
import stringoperations.StringModifier;

import java.time.LocalTime;

public class SwimmingPool extends Amenity implements RateBased {
    private int ratePerHour;

    public SwimmingPool(LocalTime openingTime, LocalTime closingTime, int
maximumCapacity, int rate) {
        super(openingTime, closingTime, maximumCapacity);
        this.ratePerHour = rate;
    }

    @Override
    public void utilize(Guest guest, int hoursUsed) {
        MongoModifier mongo = new MongoModifier();
        guest.purchase(StringModifier.addSpace(getClass().getSimpleName()) + "
x" + hoursUsed + " hours", getRate() * hoursUsed);
        mongo.addBill(guest);
    }

    @Override
```

```
    public int getRate() {
        return ratePerHour;
    }

    @Override
    public void setRate(int rate) {
        this.ratePerHour = rate;
    }
}
```

## PACKAGE HOTEL EXCEPTIONS :

Java

```
// PACKAGE HOTEL EXCEPTIONS:

// BILLNOTFOUNDEXCEPTION CLASS:

package hotelexceptions;

public class BillNotFoundException extends Exception {
}

// GUESTNOTFOUNDEXCEPTION CLASS:

package hotelexceptions;

public class GuestNotFoundException extends Exception {
}

// ILLEGALPASSWORDACCESSEXCEPTION CLASS:

package hotelexceptions;

public class IllegalPasswordAccessException extends Exception {
    @Override
    public String toString() {
        return "Password Access Token is Invalid";
    }
}

// PREEXISISTINGROOMEXCEPTION CLASS:
```

```java
package hotelexceptions;

public class PreexisistingRoomException extends Exception{
    @Override
    public String toString() {
        return "Room already exists";
    }
}

// PREEXISISTINGUSEREXCEPTION CLASS:

package hotelexceptions;

public class PreexisistingUserException extends Exception{
    @Override
    public String toString() {
        return "This username already Exists";
    }
}

// ROOMNOTFOUNDEXCEPTION CLASS:

package hotelexceptions;

public class RoomNotFoundException extends Exception{
    @Override
    public String toString() {
        return "Room does not exist";
    }
}

// USERNOTFOUNDEXCEPTION CLASS:

package hotelexceptions;

public class UserNotFoundException extends Exception {
    @Override
    public String toString() {
        return "User does not exist";
    }
}
```

## PACKAGE LAYOUTS :

Java

```java
//PACKAGE LAYOUTS:

// WRAPLAYOUT CLASS:

package layouts;

import java.awt.*;
import javax.swing.JScrollPane;
import javax.swing.SwingUtilities;

/**
 *  FlowLayout subclass that fully supports wrapping of components.
 */
public class WrapLayout extends FlowLayout
{
    private Dimension preferredLayoutSize;

    /**
     * Constructs a new <code>WrapLayout</code> with a left
     * alignment and a default 5-unit horizontal and vertical gap.
     */
    public WrapLayout()
    {
        super();
```

```
}

/**
 * Constructs a new <code>FlowLayout</code> with the specified
 * alignment and a default 5-unit horizontal and vertical gap.
 * The value of the alignment argument must be one of
 * <code>WrapLayout</code>, <code>WrapLayout</code>,
 * or <code>WrapLayout</code>.
 * @param align the alignment value
 */
public WrapLayout(int align)
{
    super(align);
}

/**
 * Creates a new flow layout manager with the indicated alignment
 * and the indicated horizontal and vertical gaps.
 * <p>
 * The value of the alignment argument must be one of
 * <code>WrapLayout</code>, <code>WrapLayout</code>,
 * or <code>WrapLayout</code>.
 * @param align the alignment value
 * @param hgap the horizontal gap between components
 * @param vgap the vertical gap between components
 */
public WrapLayout(int align, int hgap, int vgap)
{
    super(align, hgap, vgap);
}

/**
 * Returns the preferred dimensions for this layout given the
 * <i>visible</i> components in the specified target container.
 * @param target the component which needs to be laid out
 * @return the preferred dimensions to lay out the
 * subcomponents of the specified container
 */
@Override
public Dimension preferredLayoutSize(Container target)
{
    return layoutSize(target, true);
}
```

```
    /**
     * Returns the minimum dimensions needed to layout the <i>visible</i>
     * components contained in the specified target container.
     * @param target the component which needs to be laid out
     * @return the minimum dimensions to lay out the
     * subcomponents of the specified container
     */
    @Override
    public Dimension minimumLayoutSize(Container target)
    {
        Dimension minimum = layoutSize(target, false);
        minimum.width -= (getHgap() + 1);
        return minimum;
    }

    /**
     * Returns the minimum or preferred dimension needed to layout the target
     * container.
     *
     * @param target target to get layout size for
     * @param preferred should preferred size be calculated
     * @return the dimension to layout the target container
     */
    private Dimension layoutSize(Container target, boolean preferred)
    {
        synchronized (target.getTreeLock())
        {
            //  Each row must fit with the width allocated to the containter.
            //  When the container width = 0, the preferred width of the
container
            //  has not yet been calculated so lets ask for the maximum.

            int targetWidth = target.getSize().width;
            Container container = target;

            while (container.getSize().width == 0 && container.getParent() !=
null)
            {
                container = container.getParent();
            }

            targetWidth = container.getSize().width;

            if (targetWidth == 0)
```

```
                targetWidth = Integer.MAX_VALUE;

            int hgap = getHgap();
            int vgap = getVgap();
            Insets insets = target.getInsets();
            int horizontalInsetsAndGap = insets.left + insets.right + (hgap *
2);
            int maxWidth = targetWidth - horizontalInsetsAndGap;

            //  Fit components into the allowed width

            Dimension dim = new Dimension(0, 0);
            int rowWidth = 0;
            int rowHeight = 0;

            int nmembers = target.getComponentCount();

            for (int i = 0; i < nmembers; i++)
            {
                Component m = target.getComponent(i);

                if (m.isVisible())
                {
                    Dimension d = preferred ? m.getPreferredSize() :
m.getMinimumSize();

                    //  Can't add the component to current row. Start a new
row.

                    if (rowWidth + d.width > maxWidth)
                    {
                        addRow(dim, rowWidth, rowHeight);
                        rowWidth = 0;
                        rowHeight = 0;
                    }

                    //  Add a horizontal gap for all components after the first

                    if (rowWidth != 0)
                    {
                        rowWidth += hgap;
                    }

                    rowWidth += d.width;
```

```
                    rowHeight = Math.max(rowHeight, d.height);
                }
            }

            addRow(dim, rowWidth, rowHeight);

            dim.width += horizontalInsetsAndGap;
            dim.height += insets.top + insets.bottom + vgap * 2;

            //      When using a scroll pane or the DecoratedLookAndFeel we
need to
            //  make sure the preferred size is less than the size of the
            //  target containter so shrinking the container size works
            //  correctly. Removing the horizontal gap is an easy way to do
this.

            Container scrollPane =
SwingUtilities.getAncestorOfClass(JScrollPane.class, target);

            if (scrollPane != null && target.isValid())
            {
                dim.width -= (hgap + 1);
            }

            return dim;
        }
    }

    /*
     *  A new row has been completed. Use the dimensions of this row
     *  to update the preferred size for the container.
     *
     *  @param dim update the width and height when appropriate
     *  @param rowWidth the width of the row to add
     *  @param rowHeight the height of the row to add
     */
    private void addRow(Dimension dim, int rowWidth, int rowHeight)
    {
        dim.width = Math.max(dim.width, rowWidth);

        if (dim.height > 0)
        {
            dim.height += getVgap();
        }
```

```
        dim.height += rowHeight;
    }
}
```

## PACKAGE PAYMENTS :

Java

```java
// PACKAGE PAYMENTS:

// BILL CLASS:

package payments;

import java.util.HashMap;
import java.util.LinkedHashMap;

public class Bill {
    private LinkedHashMap<String, Double> purchases = new LinkedHashMap<>();
    private double totalAmount;

    public void addPayment(String reason, double amount) {
        purchases.put(reason, amount);
        totalAmount += amount;
    }
    public double getTotalAmount() {
        return totalAmount;
    }

    public LinkedHashMap<String, Double> getPurchases() {
        return purchases;
    }

    public void printInvoice() {

    }
}
```

```
// PDFCREATOR CLASS:

package payments;

import accessors.Guest;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.layout.Document;
import com.itextpdf.kernel.geom.PageSize;
import com.itextpdf.layout.borders.Border;
import com.itextpdf.layout.borders.SolidBorder;
import com.itextpdf.layout.element.Cell;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;
import com.itextpdf.layout.property.TextAlignment;
import database.MongoModifier;

import java.time.LocalDate;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class PDFCreator {

    public void createPDF(Guest guest) throws FileNotFoundException {
        LocalDate date = LocalDate.now();
        SubBill b = new SubBill(); // Create an instance
        HashMap<String, Double> purchase = b.getPurchases();
        double total;

        String path = "invoice.pdf";
        PdfWriter pdfWriter = new PdfWriter(path);
        PdfDocument pdfDocument = new PdfDocument(pdfWriter);
        Document document = new Document(pdfDocument, PageSize.A4);
        Paragraph producPara = new Paragraph("CHECKINNMATE");

document.add(producPara.setBold().setFontSize(30f).setTextAlignment(TextAlignme
nt.CENTER));

        float threecol = 190f;
        float threecol1 = 520f;
```

```
        float width[] = {threecol1};
        float col[] = {threecol, threecol, threecol};

        Border gb = new SolidBorder(2f);
        Table divider = new Table(width);
        divider.setBorder(gb);
        document.add(divider);

        Paragraph para1 = new  Paragraph("Date: " +
String.valueOf(date)).setTextAlignment(TextAlignment.RIGHT);
        document.add(para1);

        Paragraph para = new Paragraph("INVOICE");

document.add(para.setBold().setFontSize(20f).setTextAlignment(TextAlignment.CEN
TER));

        Table table = new Table(col);
        table.addCell(new Cell().add(new Paragraph("Description")).setBold());
        table.addCell(new Cell().add(new
Paragraph("Quantity")).setTextAlignment(TextAlignment.CENTER).setBold());
        table.addCell(new Cell().add(new
Paragraph("Price")).setTextAlignment(TextAlignment.RIGHT).setBold().setMarginRi
ght(15f));
// m has been created only as a sample
        //HashMap<String,Double> m = (new
MongoModifier()).getBill(guest.getUsername()).getPurchases();
        HashMap<String,Double> m = guest.getBill().getPurchases();
        total = guest.getBill().getTotalAmount();
        //m.put("Swimming",500.0);
        //m.put("Gym",600.0);

// used for the main one
        List<Product> prodL = new ArrayList<>();
        for (Map.Entry<String, Double> entry : m.entrySet()) {
            //System.out.println();
            String k = entry.getKey();
            Double v = entry.getValue();
            prodL.add(new Product(k, v));
        }
        for (Product prod : prodL) {
            table.addCell(new Cell().add(new
Paragraph(prod.getPname()))).setBorder(Border.NO_BORDER).setMarginLeft(10f));
```

```
            table.addCell(new Cell().add(new
Paragraph(String.valueOf(1))).setTextAlignment(TextAlignment.CENTER).setBorder(
Border.NO_BORDER));
            table.addCell(new Cell().add(new
Paragraph(String.valueOf(prod.getPriceperpiece()))).setTextAlignment(TextAlignm
ent.RIGHT).setBorder(Border.NO_BORDER).setMarginRight(15f));
        }

        document.add(table.setMarginBottom(30f));
        Paragraph tot = new Paragraph("Total: "+ total);

document.add(tot.setFontSize(15f).setBold().setTextAlignment(TextAlignment.RIGH
T));
        Paragraph gst = new Paragraph("GST : 18% ");
        Double final_value = total * 118 / (double)100;

document.add(gst.setFontSize(16f).setTextAlignment(TextAlignment.RIGHT));
        Paragraph final_val = new Paragraph("Final Bill "+ final_value);

document.add(final_val.setFontSize(20f).setTextAlignment(TextAlignment.RIGHT).s
etBold());
        document.close();
    }
}
// SUBBILL CLASS:

package payments;

import java.util.HashMap;

public class Product {
    private String pname;
    private final int quantity = 1;
    private Double priceperpiece;
    public Product(String pname, Double priceperpiece) {
        this.pname = pname;
        this.priceperpiece = priceperpiece;
    }
    public String getPname() {
        return pname;
    }
    public void setPname(String pname) {
        this.pname = pname;
    }
```

```
    public Double getPriceperpiece() {
        return priceperpiece;
    }
    public void setPriceperpiece(Double priceperpiece) {
        this.priceperpiece = priceperpiece;
    }
}
// PRODUCT CLASS:

package payments;

import java.util.HashMap;

public class SubBill {
    private HashMap<String, Double> purchases = new HashMap<>();
    private double totalAmount;

    public SubBill() {
        this.purchases = new HashMap<>();
        this.totalAmount = 0.0;
    }

    public void addPayment(String reason, double amount) {
        purchases.put(reason, amount);
        totalAmount += amount;
    }

    public double getTotalAmount() {
        return totalAmount;
    }

    public HashMap<String, Double> getPurchases() {
        return purchases;
    }

    public String printInvoice() {
        return ""; // Yet to implement
    }
}
```

## PACKAGE SCREEN COMPONENTS :

Java

```
// PACKAGE SCREENCOMPONENTS:


// PACKAGE ADMIN COMPONENTS:


// ADDROOMPAGEPANEL CLASS:
package screencomponents.admincomponents;

import javax.swing.*;
import java.awt.*;

public class AddRoomPagePanel extends JPanel {
    public AddRoomPagePanel(){
        setBackground(Color.white);
        setLayout(new GridBagLayout());

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        add(new AddRoomPanel(), new GridBagConstraints());

    }
}
// ADDROOMPANEL CLASS:
package screencomponents.admincomponents;
```

```java
import accessors.Guest;
import database.MongoModifier;
import hotelcomponents.rooms.Room;
import hotelexceptions.PreexisistingRoomException;
import org.bson.Document;
import screencomponents.constraints.DropDownPanel;
import stringoperations.StringModifier;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.lang.reflect.InvocationTargetException;

public class AddRoomPanel extends JPanel{
    protected JTextField roomIDField;
    protected JComboBox<Integer> capacityDropDown;

    protected JTextField costField;
    protected JTextField areaField;
    protected JComboBox<String> roomTypeDropDown;
    protected JLabel loginTitle;
    protected JLabel errorLabel;
    protected JButton enterButton;

    protected GridBagConstraints c = new GridBagConstraints();
    public AddRoomPanel(){
        setLayout(new GridBagLayout());
        setBorder(BorderFactory.createLineBorder(Color.black, 5));
        setBackground(Color.white);

        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.insets = new Insets(20, 40, 0, 40);
        c.anchor = GridBagConstraints.CENTER;

        loginTitle = new JLabel("ADD ROOM");
        loginTitle.setBackground(Color.white);
```

```
loginTitle.setForeground(Color.black);
loginTitle.setHorizontalAlignment(JLabel.CENTER);
loginTitle.setVerticalAlignment(JLabel.CENTER);
loginTitle.setFont(new Font("Times New Roman", Font.BOLD, 50));
add(loginTitle, c);

c.gridx = 0;
c.gridy = 1;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.EAST;

JLabel usernameText = new JLabel("RoomID :");
usernameText.setForeground(Color.black);
usernameText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(usernameText, c);

c.gridx = 1;
c.gridy = 1;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.WEST;

roomIDField = new JTextField(4);
roomIDField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
//roomIDField.addActionListener(new CredentialsHandler());
add(roomIDField, c);

c.gridx = 0;
c.gridy = 2;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.EAST;

JLabel passwordText = new JLabel("Capacity :");
passwordText.setForeground(Color.black);
passwordText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(passwordText, c);
```

```
c.gridx = 1;
c.gridy = 2;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.WEST;

capacityDropDown = new JComboBox<>(new Integer[]{1, 2, 3, 4, 5, 6});
add(capacityDropDown, c);

c.gridx = 0;
c.gridy = 3;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.EAST;

JLabel costLabel = new JLabel("Cost :");
costLabel.setForeground(Color.black);
costLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(costLabel, c);

c.gridx = 1;
c.gridy = 3;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
c.anchor = GridBagConstraints.WEST;

costField = new JTextField(4);
costField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
//capacityField.addActionListener(new CredentialsHandler());
add(costField, c);

c.gridx = 0;
c.gridy = 4;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(15, 15, 0, 15);
```

```
        c.anchor = GridBagConstraints.EAST;

        JLabel areaLabel = new JLabel("Area :");
        areaLabel.setForeground(Color.black);
        areaLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        add(areaLabel, c);

        c.gridx = 1;
        c.gridy = 4;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(15, 15, 0, 15);
        c.anchor = GridBagConstraints.WEST;

        areaField = new JTextField(4);
        areaField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        //capacityField.addActionListener(new CredentialsHandler());
        add(areaField, c);


        c.gridx = 0;
        c.gridy = 5;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(15, 15, 0, 15);
        c.anchor = GridBagConstraints.EAST;

        JLabel roomTypeLabel = new JLabel("Room Type :");
        roomTypeLabel.setForeground(Color.black);
        roomTypeLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        add(roomTypeLabel, c);

        c.gridx = 1;
        c.gridy = 5;
        c.insets = new Insets(15, 15, 0, 15);
        c.anchor = GridBagConstraints.WEST;

        roomTypeDropDown = new JComboBox<>(new String[]{"Single Room", "Deluxe
Room", "Double Room", "Executive Room", "Apartment Room", "Cabana Room", "King
Room", "Queen Room"});
        add(roomTypeDropDown, c);
```

```
        c.gridx = 0;
        c.gridy = 7;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.insets = new Insets(22, 20, 20, 20);
        c.anchor = GridBagConstraints.CENTER;

        enterButton = new JButton("ENTER");
        enterButton.setUI(new BasicButtonUI());
        enterButton.setForeground(Color.black);
        enterButton.setBackground(Color.white);
        enterButton.setFocusPainted(false);

        enterButton.setFont((new Font("Times New Roman", Font.PLAIN, 20)));
        //enterButton.addActionListener(new CredentialsHandler());
        enterButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                setCursor(Cursor.getDefaultCursor());
            }
        });
        enterButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                MongoModifier mongo = new MongoModifier();
                //Room(int roomId, int capacity, double cost, int area, Guest
guest)
                int roomID = (int)Integer.parseInt(roomIDField.getText());
                int capacity = (int)capacityDropDown.getSelectedItem();
                double cost = (double)Double.parseDouble(costField.getText());
                int area = (int)Integer.parseInt(areaField.getText());
                String roomType = (String)roomTypeDropDown.getSelectedItem();
                Room room;
                try {
                    room = ((Room)Class.forName("hotelcomponents.rooms." +
StringModifier.removeSpace(roomType)).getConstructor(int.class, int.class,
```

```
double.class, int.class, Guest.class).newInstance(roomID, capacity, cost, area,
null));
                } catch (InstantiationException ex) {
                    throw new RuntimeException(ex);
                } catch (IllegalAccessException ex) {
                    throw new RuntimeException(ex);
                } catch (InvocationTargetException ex) {
                    throw new RuntimeException(ex);
                } catch (NoSuchMethodException ex) {
                    throw new RuntimeException(ex);
                } catch (ClassNotFoundException ex) {
                    throw new RuntimeException(ex);
                }

                try {
                    mongo.addRoom(room);
                    /*roomIDField.setText("");
                    capacityDropDown.setSelectedIndex(0);
                    costField.setText("");
                    areaField.setText("");
                    roomTypeDropDown.setSelectedIndex(0);*/
                    errorLabel.setVisible(true);
                } catch (PreexisistingRoomException ex) {
                    throw new RuntimeException(ex);
                }
            }
        });
        //rooms.add((Room)Class.forName("hotelcomponents." +
roomDoc.getString("roomType")).getConstructor(Document.class,
Guest.class).newInstance(roomDoc, null));

        add(enterButton, c);

        c.gridx = 0;
        c.gridy = 8;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.insets = new Insets(0, 20, 5, 20);
        c.anchor = GridBagConstraints.CENTER;

        errorLabel = new JLabel("Room Added Successfully!!");
        errorLabel.setForeground(Color.green);
        errorLabel.setFont(new Font("Times New Roman", Font.PLAIN, 15));
```

```
        errorLabel.setHorizontalAlignment(JLabel.CENTER);
        errorLabel.setVerticalAlignment(JLabel.CENTER);
        errorLabel.setVisible(false);
        add(errorLabel, c);
    }
}
// CHECKFEEDBACKPAGEPANEL CLASS:
package screencomponents.admincomponents;

import database.MongoModifier;
import feedbackcomponents.Feedback;
import layouts.WrapLayout;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class CheckFeedbackPagePanel extends JPanel {

    public CheckFeedbackPagePanel() {
        setBackground(Color.white);
        setLayout(new WrapLayout(FlowLayout.CENTER, 20, 20));

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));

        MongoModifier mongo = new MongoModifier();
        ArrayList<Feedback> feedbacks = mongo.getFeedback();

        for (Feedback feedback : feedbacks) {
            add(new FeedbackPanel(feedback));
        }
    }
}
// FEEDBACKPANEL CLASS:
package screencomponents.admincomponents;

import feedbackcomponents.Feedback;
import stringoperations.StringModifier;

import javax.swing.*;
import java.awt.*;
```

```
public class FeedbackPanel extends JPanel{

    public FeedbackPanel(Feedback feedback) {
        GridBagConstraints c = new GridBagConstraints();

        setBackground(Color.white);
        setLayout(new GridBagLayout());
        //setPreferredSize(new Dimension(500, 300));
        setBorder(BorderFactory.createLineBorder(Color.black, 4));

        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.anchor = GridBagConstraints.WEST;
        c.insets = new Insets(0, 15, 0, 120);

        add(new JPanel() {{
            setBackground(Color.white);
            add(new JLabel("Username: " + feedback.getAssociatedUsername()) {{
                setFont(new Font("Times New Roman", Font.PLAIN, 32));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.black);
            }});
        }}, c);

        c.gridx = 1;
        c.anchor = GridBagConstraints.EAST;
        c.insets = new Insets(0, 120, 0, 15);

        add(new JPanel() {{
            setBackground(Color.white);
            add(new JLabel("Date: " + feedback.getDate()) {{
                setFont(new Font("Times New Roman", Font.PLAIN, 32));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.black);
            }});
        }}, c);



        c.gridx = 0;
```

```
        c.gridy = 1;
        c.anchor = GridBagConstraints.WEST;
        c.insets = new Insets(0, 15, 0, 120);

        add(new JPanel() {{
            setBackground(Color.white);
            add(new JLabel("Topic: " +
StringModifier.addSpace(feedback.getType())) {{
                setFont(new Font("Times New Roman", Font.PLAIN, 32));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.black);
            }});
        }}, c);



        c.gridx = 1;
        c.anchor = GridBagConstraints.EAST;
        c.insets = new Insets(0, 120, 0, 15);

        add(new JPanel() {{
            setBackground(Color.white);
            add(new JLabel("Rating : " + feedback.getRating() + "/5") {{
                setFont(new Font("Times New Roman", Font.PLAIN, 32));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.black);
            }});
        }}, c);

        c.gridx = 0;
        c.gridy = 2;
        c.gridwidth = 2;
        c.anchor = GridBagConstraints.CENTER;
        c.fill = GridBagConstraints.BOTH;
        c.insets = new Insets(0, 20, 0, 15);

        add(new JTextArea("Comment:\n" + feedback.getFeedbackString()) {{
            setLineWrap(true);
            setWrapStyleWord(true);
            setBackground(Color.white);
            setForeground(Color.black);
            setEditable(false);
```

```
            setFont(new Font("Times New Roman", Font.PLAIN, 40));
        }}, c);
    }
}
// REMOVEROOMPAGEPANEL CLASS:
package screencomponents.admincomponents;

import database.MongoModifier;
import hotelcomponents.rooms.Room;
import layouts.WrapLayout;
import screencomponents.RoomPanel;
import screencomponents.pages.SearchPagePanel;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

public class RemoveRoomPagePanel extends JPanel{

    private MongoModifier mongo = new MongoModifier();
    private ArrayList<Room> allRooms;
    public RemoveRoomPagePanel(){
        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new WrapLayout(FlowLayout.CENTER, 33, 33));

        ArrayList<Room> roomsList = new ArrayList<>();
        for (Room room : mongo.getAllRooms()) {
            if (room.getGuest() == null)
                roomsList.add(room);
            else if (room.getGuest() != null &&
room.getGuest().getUsername().equals(""))
                roomsList.add(room);
        }
        Collections.sort(roomsList);
        for (Room room : roomsList) {
            add((new RemoveRoomPanel(room, new RemoveClicked(room))));
```

```
        }
        allRooms = roomsList;
    }
    public void reDisplayRooms() {
        removeAll();


        ArrayList<Room> roomsList = new ArrayList<>();
        for (Room room : allRooms) {
            if (room.getGuest() == null)
                roomsList.add(room);
            else if (room.getGuest() != null &&
room.getGuest().getUsername().equals(""))
                roomsList.add(room);
        }
        Collections.sort(roomsList);
        for (Room room : roomsList) {
            add((new RemoveRoomPanel(room, new RemoveClicked(room))));
        }

        revalidate();
        repaint();
    }
    private class RemoveClicked implements ActionListener {
        Room room;

        RemoveClicked(Room room) {
            this.room = room;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            JButton srcButton = ((JButton)e.getSource());
            if (srcButton.getText().equals("REMOVE")){
                srcButton.setForeground(Color.RED);
                srcButton.setText("CONFIRM");
            } else if (srcButton.getText().equals("CONFIRM")) {
                mongo.removeRoom(room);
                reDisplayRooms();
            }

        }
    }
    public static void main(String[] args) {
```

```
        JFrame window = new JFrame();
        window.setSize(1200, 800);
        window.setTitle("CheckInnMate");
        //window.setResizable(false);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setLocationRelativeTo(null);
        window.getContentPane().setBackground(Color.white);

        window.add(new RemoveRoomPagePanel());
        window.setVisible(true);
    }


}
// REMOVEROOMPANEL CLASS:
package screencomponents.admincomponents;

import database.MongoModifier;
import hotelcomponents.rooms.Room;
import layouts.WrapLayout;
import screencomponents.RoomPanel;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class RemoveRoomPanel extends JPanel{
    private final MongoModifier mongo = new MongoModifier();

    private Font roomFont = new Font("Times New Roman", Font.PLAIN, 25);
    private JButton removeButton, detailsButton;
    private JLabel imageLabel, capacityLabel, areaLabel;
    private JPanel gapPanel1, gapPanel2, gapPanel3;

    public RemoveRoomPanel(Room room, ActionListener action) {
        setBackground(Color.white);
        setLayout(new GridBagLayout());
        setBorder(BorderFactory.createLineBorder(Color.black, 5));
        //setPreferredSize(new Dimension(0, 400));
        GridBagConstraints c = new GridBagConstraints();
```

```
        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.gridheight = 1;
        c.insets = new Insets(-1,-1, 1, -1);
        c.anchor = GridBagConstraints.CENTER;

        System.out.println(room.getType());
        Image img = new ImageIcon(RoomPanel.class.getResource("/RoomImages/" +
room.getType() + ".jpg")).getImage();
        add(imageLabel = new JLabel(new ImageIcon(img.getScaledInstance(400,
200, Image.SCALE_SMOOTH))), c);

        c.gridy = 1;
        c.anchor = GridBagConstraints.WEST;
        c.insets = new Insets(-1,-1, 0, -1);

        add(gapPanel1 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 2;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel typeLabel = new JLabel("Room: " +
room.getClass().getSimpleName());
        typeLabel.setForeground(Color.black);
        typeLabel.setFont(roomFont);
        typeLabel.setHorizontalAlignment(JLabel.LEFT);
        typeLabel.setVerticalAlignment(JLabel.CENTER);

        add(typeLabel, c);

        c.gridy = 3;
        c.insets = new Insets(-1, -1, 0, -1);

        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);
```

```
        c.gridx = 0;
        c.gridy = 4;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel roomNoLabel = new JLabel("Room No: " + room.getRoomId());
        roomNoLabel.setForeground(Color.black);
        roomNoLabel.setFont(roomFont);
        roomNoLabel.setHorizontalAlignment(JLabel.LEFT);
        roomNoLabel.setVerticalAlignment(JLabel.CENTER);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(roomNoLabel, c);

        c.gridy = 5;
        c.insets = new Insets(-1, -1, 0, -1);

        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 6;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel roomCostLabel = new JLabel("Cost Per Night: \u20B9" +
(int)room.getCost());
        roomCostLabel.setForeground(Color.black);
        roomCostLabel.setFont(roomFont);
        roomCostLabel.setHorizontalAlignment(JLabel.LEFT);
        roomCostLabel.setVerticalAlignment(JLabel.CENTER);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(roomCostLabel, c);

        c.gridy = 7;
        c.insets = new Insets(-1, -1, 0, -1);

        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);
```

```
        c.gridx = 0;
        c.gridy = 8;
        c.insets = new Insets(-1, 5, 0, -1);

        capacityLabel = new JLabel("Capacity: " + room.getCapacity());
        capacityLabel.setForeground(Color.black);
        capacityLabel.setFont(roomFont);
        capacityLabel.setHorizontalAlignment(JLabel.LEFT);
        capacityLabel.setVerticalAlignment(JLabel.CENTER);
        capacityLabel.setVisible(false);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(capacityLabel, c);

        c.gridy = 9;
        c.insets = new Insets(-1, -1, 0, -1);

        add(gapPanel2 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
            setVisible(false);
        }}, c);

        c.gridx = 0;
        c.gridy = 10;
        c.insets = new Insets(-1, 5, 0, -1);

        areaLabel = new JLabel("Area: " + room.getArea() + "m\u00B2");
        areaLabel.setForeground(Color.black);
        areaLabel.setFont(roomFont);
        areaLabel.setHorizontalAlignment(JLabel.LEFT);
        areaLabel.setVerticalAlignment(JLabel.CENTER);
        areaLabel.setVisible(false);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(areaLabel, c);

        c.gridy = 11;
        c.insets = new Insets(-1, -1, 0, -1);

        add(gapPanel3 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
            setVisible(false);
```

```
        }}, c);


        c.gridx = 0;
        c.gridy = 12;
        c.gridwidth = 1;
        c.insets = new Insets(-1, -1, -1, -1);
        c.fill = GridBagConstraints.BOTH;

        removeButton = new JButton("REMOVE");
        removeButton.setUI(new BasicButtonUI());
        removeButton.setBackground(Color.white);
        removeButton.setForeground(Color.black);
        //removeButton.setBorderPainted(false);
        removeButton.setFocusPainted(false);
        removeButton.setFont(roomFont);
        removeButton.setBorder(BorderFactory.createMatteBorder(0, 0, 0, 4,
Color.black));
        removeButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {

removeButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                removeButton.setCursor(Cursor.getDefaultCursor());
            }
        });
        removeButton.addActionListener(action);
        add(removeButton, c);

        c.gridx = 1;
        c.gridy = 12;
        c.gridwidth = 1;
        c.insets = new Insets(-1, -1, -1, -1);
        c.fill = GridBagConstraints.BOTH;

        detailsButton = new JButton("GET DETAILS");
        detailsButton.setUI(new BasicButtonUI());
        detailsButton.setBackground(Color.white);
        detailsButton.setForeground(Color.black);
        detailsButton.setBorderPainted(false);
```

```
        detailsButton.setFocusPainted(false);
        detailsButton.setFont(roomFont);
        detailsButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {

detailsButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                detailsButton.setCursor(Cursor.getDefaultCursor());
            }
        });
        detailsButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (((JButton)e.getSource()).getText().equals("GET DETAILS")) {
                    imageLabel.setVisible(false);
                    capacityLabel.setVisible(true);
                    areaLabel.setVisible(true);
                    gapPanel1.setVisible(false);
                    gapPanel2.setVisible(true);
                    gapPanel3.setVisible(true);
                    ((JButton)e.getSource()).setText("CLOSE");
                }
                else {
                    imageLabel.setVisible(true);
                    capacityLabel.setVisible(false);
                    areaLabel.setVisible(false);
                    gapPanel1.setVisible(true);
                    gapPanel2.setVisible(false);
                    gapPanel3.setVisible(false);
                    ((JButton)e.getSource()).setText("GET DETAILS");
                }
            }
        });

        add(detailsButton, c);
        //setPreferredSize(new Dimension(500, 0));

    }
}
```

```
// PACKAGE CONSTRAINTS:

// DROPDOWNPANEL CLASS:
package screencomponents.constraints;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class DropDownPanel<T> extends JPanel {
    private JComboBox<T> dropDownMenu;

    public DropDownPanel(String text, T[] arr) {
        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createMatteBorder(0,
0, 4, 0, Color.black), BorderFactory.createEmptyBorder(5, 0, 5, 0)));
        setLayout(new GridBagLayout());

        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        c.gridwidth = 1;
        c.gridheight = 1;
        c.anchor = GridBagConstraints.CENTER;
        c.insets = new Insets(0, 0, 15, 0);


        JLabel textLabel = new JLabel(text);
        textLabel.setBackground(Color.white);
        textLabel.setForeground(Color.black);
        textLabel.setHorizontalAlignment(JLabel.CENTER);
        textLabel.setVerticalAlignment(JLabel.CENTER);
        textLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));

        add(textLabel, c);

        c.gridy = 1;
        c.insets = new Insets(0, 0, 0, 0);

        dropDownMenu = new JComboBox<T>(arr);

((JLabel)dropDownMenu.getRenderer()).setHorizontalAlignment(SwingConstants.CENT
ER);
```

```
        add(dropDownMenu, c);
    }

    public String getCurrentSelection() {
        return (String)dropDownMenu.getSelectedItem();
    }
}
// PRICEBAR CLASS:
package screencomponents.constraints;

import screencomponents.ScreenPanel;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;

public class PriceBar extends JPanel {
    private JSlider costSlider;
    private JSlider associatedSlider;
    private int bound;
    private int type;

    public static final int LOWER = 1;
    public static final int UPPER = 2;

    public PriceBar(int type) {
        this.associatedSlider = associatedSlider;
        this.type = type;

        setBackground(Color.white);
        setLayout(new GridBagLayout());

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createMatteBorder(0,
0, 4, 0, Color.black), BorderFactory.createEmptyBorder(5, 0, 5, 0)));

        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(0, 0, 0, 0);
        c.anchor = GridBagConstraints.CENTER;
```

```
c.fill = GridBagConstraints.NONE;

JLabel costTitle = new JLabel();
if (type == 1)
     costTitle.setText("COST LOWER BOUND");
else if (type == 2)
    costTitle.setText("COST UPPER BOUND");
costTitle.setHorizontalAlignment(JLabel.CENTER);
costTitle.setVerticalAlignment(JLabel.CENTER);
costTitle.setForeground(Color.black);
costTitle.setBackground(Color.white);
costTitle.setFont(new Font("Times New Roman", Font.PLAIN, 10));

add(new JPanel() {{
    setBackground(Color.white);
    setLayout(new BorderLayout());
    add(costTitle);
}}, c);

c.gridx = 0;
c.gridy = 1;
c.fill = GridBagConstraints.BOTH;
c.weightx = 1;
c.weighty = 1;

costSlider = new JSlider(0, 10000);
costSlider.setBackground(Color.white);
costSlider.setMinorTickSpacing(500);
costSlider.setMajorTickSpacing(2500);
if (type == LOWER)
    costSlider.setValue(0);
else if (type == UPPER)
    costSlider.setValue(10000);
costSlider.setPaintTicks(true);
costSlider.setPaintLabels(true);
costSlider.setSnapToTicks(true);
//costSlider.setBorder(BorderFactory.createEmptyBorder(2, 2, 2, 2));
costSlider.addChangeListener(new ChangeListener() {
    @Override
    public void stateChanged(ChangeEvent e) {
        int selectedValue;
        switch(type) {
            case LOWER:
                selectedValue = costSlider.getValue();
```

```
                        System.out.println("Selected Value: " + selectedValue);
                        if (selectedValue > associatedSlider.getValue())
                            costSlider.setValue(associatedSlider.getValue());
                        break;
                    case UPPER:
                        selectedValue = costSlider.getValue();
                        System.out.println("Selected Value: " + selectedValue);
                        if (selectedValue < associatedSlider.getValue())
                            costSlider.setValue(associatedSlider.getValue());
                        break;
                }
            }
        });

        add(costSlider, c);
    }
    public void addAssociatedSlider(JSlider slider) {
        associatedSlider = slider;
    }
    public JSlider getCostSlider() {
        return costSlider;
    }
    public JSlider getAssociatedSlider() {
        return associatedSlider;
    }

    public void setAssociatedSlider(JSlider associatedSlider) {
        this.associatedSlider = associatedSlider;
    }

    public static void main(String[] args) {
        JFrame window = new JFrame();
        window.setSize(150, 150);
        window.setTitle("CheckInnMate");
        //window.setResizable(false);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setLocationRelativeTo(null);
        window.getContentPane().setBackground(Color.white);

        //window.add(new PriceBar());
        window.setVisible(true);
    }

}
```

```
// PACKAGE PAGES:

// CONFIRMBOOKINGPANEL CLASS:
package screencomponents.pages;

import accessors.Guest;
import database.MongoModifier;
import hotelcomponents.rooms.Room;
import hotelexceptions.PreexisistingRoomException;
import hotelexceptions.UserNotFoundException;
import layouts.WrapLayout;
import screencomponents.NavButton;
import screencomponents.RoomPanel;
import screencomponents.ScreenPanel;
import stringoperations.StringModifier;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDate;
import java.util.HashSet;

public class ConfirmBookingPanel extends JPanel {
    private JPanel roomsPanel;
    public ConfirmBookingPanel(HashSet<Room> rooms) {
        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new BorderLayout());

        add(new JPanel() {{
            setBackground(Color.white);
            add(new JLabel("WOULD YOU LIKE TO CONFIRM YOUR BOOKING?") {{
                setForeground(Color.black);
                setFont(new Font("Times New Roman", Font.BOLD, 35));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
            }});
        }}, BorderLayout.NORTH);
```

```
        add(new JScrollPane(roomsPanel = new JPanel() {{
            setBackground(Color.white);
            setLayout(new WrapLayout(FlowLayout.CENTER, 33, 33));
        }}, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{
            getVerticalScrollBar().setUnitIncrement(10);
        }}, BorderLayout.CENTER);

        //setPreferredSize(new Dimension(500, 0));
        for (Room room : rooms) {
            roomsPanel.add((new RoomPanel(room, null) {{
                disableButtons();
            }}));
        }

        JComboBox<LocalDate> checkInDate = new JComboBox<>();

((JLabel)checkInDate.getRenderer()).setHorizontalAlignment(SwingConstants.CENTE
R);
        for (LocalDate date = LocalDate.now();
date.isBefore(LocalDate.now().plusDays(60)); date = date.plusDays(1)) {
            checkInDate.addItem(date);
        }

        JComboBox<LocalDate> checkOutDate = new JComboBox<>();

((JLabel)checkOutDate.getRenderer()).setHorizontalAlignment(SwingConstants.CENT
ER);
        for (LocalDate date = LocalDate.now();
date.isBefore(LocalDate.now().plusDays(60)); date = date.plusDays(1)) {
            checkOutDate.addItem(date);
        }
        checkOutDate.setSelectedItem(1);

        checkInDate.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if
(((LocalDate)checkInDate.getSelectedItem()).isAfter(((LocalDate)checkOutDate.ge
tSelectedItem())))
```

```
checkInDate.setSelectedItem(((LocalDate)checkOutDate.getSelectedItem()).minusDa
ys(1));
            }
        });

        checkOutDate.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if
(((LocalDate)checkOutDate.getSelectedItem()).isBefore(((LocalDate)checkInDate.g
etSelectedItem())))

checkOutDate.setSelectedItem(((LocalDate)checkInDate.getSelectedItem()).plusDay
s(1));
            }
        });

        JLabel checkInLabel = new JLabel("Check In Date: ") {{
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.BOLD, 20));
            setHorizontalAlignment(JLabel.CENTER);
            setVerticalAlignment(JLabel.CENTER);
        }};

        JLabel checkOutLabel = new JLabel("Check Out Date: ") {{
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.BOLD, 20));
            setHorizontalAlignment(JLabel.CENTER);
            setVerticalAlignment(JLabel.CENTER);
        }};

        JPanel dateAndBookingPanel = new JPanel();
        dateAndBookingPanel.setBackground(Color.white);
        dateAndBookingPanel.setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();

        dateAndBookingPanel.add(checkInLabel, c);

        c.gridx = 1;
        c.insets = new Insets(0, 0, 0, 50);
        dateAndBookingPanel.add(checkInDate, c);

        c.gridx = 2;
```

```
        c.insets = new Insets(0, 0, 0, 0);
        dateAndBookingPanel.add(checkOutLabel, c);

        c.gridx = 3;
        c.insets = new Insets(0, 0, 0, 50);
        dateAndBookingPanel.add(checkOutDate, c);

        c.gridx = 4;
        c.insets = new Insets(0, 0, 0, 50);

        dateAndBookingPanel.add(new JButton("  BOOK  ") {{
            setUI(new BasicButtonUI());
            setBackground(Color.white);
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.PLAIN, 30));
            setFocusPainted(false);
            setBorder(BorderFactory.createMatteBorder(2, 2, 2, 2,
Color.black));
            addMouseListener(new MouseAdapter() {
                public void mouseEntered(MouseEvent e) {
                    setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
                }
                @Override
                public void mouseExited(MouseEvent e) {
                    setCursor(Cursor.getDefaultCursor());
                }
            });
            addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    if (rooms.isEmpty())
                        return;
                    MongoModifier mongo = new MongoModifier();
                    try {
                        Guest guest = new
Guest(mongo.getUser(((ScreenPanel)getParent().getParent().getParent()).getAssoc
iatedUsername()), (LocalDate)checkInDate.getSelectedItem(),
(LocalDate)checkOutDate.getSelectedItem(), rooms);
                        for (Room room : rooms) {
                            mongo.removeRoom(room);
                            room.addGuest(guest);
                            mongo.addRoom(room);

guest.purchase(StringModifier.addSpace(room.getType()), room.getCost());
```

```
                        }
                        System.out.println("Hi");
                        mongo.addGuest(guest);
                        mongo.addBill(guest);

                    }
                    catch (UserNotFoundException | PreexisistingRoomException
userNotFoundException) {
                        userNotFoundException.printStackTrace();
                    }

System.out.println(getParent().getParent().getParent().getParent().getParent().
getParent().getParent());
                    JFrame frame =
(JFrame)(getParent().getParent().getParent().getParent().getParent().getParent(
).getParent());
                    frame.getContentPane().removeAll();
                    frame.add(new ScreenPanel(), BorderLayout.CENTER);
                    frame.revalidate();
                    frame.repaint();
                }
            });
        }}, c);
        add(dateAndBookingPanel, BorderLayout.SOUTH);
    }
}
// FEEDBACKPAGEPANEL CLASS:
package screencomponents.pages;

import database.MongoModifier;
import feedbackcomponents.Feedback;
import hotelexceptions.GuestNotFoundException;
import screencomponents.AppDefaultButton;
import screencomponents.ScreenPanel;
import stringoperations.StringModifier;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class FeedbackPagePanel extends JPanel {
```

```
    private JComboBox<String> componentTypeBox;
    private JComboBox<Integer> ratingBox;
    private JTextArea feedbackArea;

    public FeedbackPagePanel() {
        setBackground(Color.white);
        setLayout(new GridBagLayout());

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));

        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(0, 0, 30, 0);

        add(new JLabel("Feedback subject: ") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 40));
            setHorizontalAlignment(JLabel.CENTER);
            setVerticalAlignment(JLabel.CENTER);
            setForeground(Color.black);
        }}, c);

        c.gridx = 1;

        componentTypeBox = new JComboBox<String>(new String[]{"General",
"Room", "Game Room", "Gym", "Swimming Pool"});
        componentTypeBox.setFont(new Font("Calibri", Font.PLAIN, 20));

((JLabel)componentTypeBox.getRenderer()).setHorizontalAlignment(SwingConstants.
CENTER);

((JLabel)componentTypeBox.getRenderer()).setVerticalAlignment(SwingConstants.CE
NTER);
        add(componentTypeBox, c);

        c.gridx = 0;
        c.gridy = 1;

        add(new JLabel("Rating: ") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 40));
            setHorizontalAlignment(JLabel.CENTER);
            setVerticalAlignment(JLabel.CENTER);
            setForeground(Color.black);
```

```
        }}, c);

        c.gridx = 1;

        ratingBox = new JComboBox<Integer>(new Integer[]{1, 2, 3, 4, 5});
        ratingBox.setFont(new Font("Calibri", Font.PLAIN, 20));

((JLabel)ratingBox.getRenderer()).setHorizontalAlignment(SwingConstants.CENTER)
;

((JLabel)ratingBox.getRenderer()).setVerticalAlignment(SwingConstants.CENTER);

        add(ratingBox, c);

        c.gridx = 0;
        c.gridy = 2;
        c.gridwidth = 2;

        feedbackArea = new JTextArea("Type here");
        feedbackArea.setBackground(new Color(240, 240, 240));
        feedbackArea.setBorder(BorderFactory.createLineBorder(Color.black, 4));
        feedbackArea.setPreferredSize(new Dimension(600, 400));
        feedbackArea.setWrapStyleWord(true);
        feedbackArea.setLineWrap(true);
        feedbackArea.setFont(new Font("Times New Roman", Font.PLAIN, 30));
        feedbackArea.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                feedbackArea.setText("");
            }
        });

        add(feedbackArea, c);

        c.gridy = 3;

        JButton sendFeedbackButton = new AppDefaultButton("ENTER");
        sendFeedbackButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                MongoModifier mongo = new MongoModifier();
                try {
                    String username =
((ScreenPanel)FeedbackPagePanel.this.getParent()).getAssociatedUsername();
```

```
                    mongo.getGuest(username)
                            .giveFeedback(new Feedback(username,
StringModifier.removeSpace((String)componentTypeBox.getSelectedItem()),
(int)ratingBox.getSelectedItem(), feedbackArea.getText()));
                    feedbackArea.setText("");
                    ratingBox.setSelectedItem("1");
                    componentTypeBox.setSelectedItem(("General"));
                }
                catch (GuestNotFoundException guestNotFoundException) {
                    guestNotFoundException.printStackTrace();
                }
            }
        });

        add(sendFeedbackButton, c);
    }
}
// LOGINPAGEPANEL CLASS:
package screencomponents.pages;

import screencomponents.LoginPanel;
import screencomponents.SignUpPanel;

import javax.swing.*;
import java.awt.*;

public class LoginPagePanel extends JPanel{
    private LoginPanel loginPanel;
    private SignUpPanel signUpPanel;

    public LoginPagePanel() {

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new GridBagLayout());
        setBackground(Color.white);

        loginPanel = new LoginPanel();
        add(loginPanel, new GridBagConstraints());

        signUpPanel = new SignUpPanel();
        signUpPanel.setVisible(false);
        add(signUpPanel, new GridBagConstraints());
```

```
    }
}

// SEARCHPAGEPANEL CLASS:
package screencomponents.pages;

import database.MongoModifier;
import hotelcomponents.rooms.Room;
import layouts.WrapLayout;
import screencomponents.NavPanel;
import screencomponents.RoomPanel;
import screencomponents.constraints.DropDownPanel;
import screencomponents.constraints.PriceBar;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.Set;

public class SearchPagePanel extends JPanel {
    private final MongoModifier mongo = new MongoModifier();
    private HashSet<Room> bookedRooms = new HashSet<>();
    private ArrayList<Room> allRooms;
    public SearchPagePanel() {
        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new WrapLayout(FlowLayout.CENTER, 33, 33));

        ArrayList<Room> roomsList = new ArrayList<>();
        for (Room room : mongo.getAllRooms()) {
            if (room.getGuest() == null)
                roomsList.add(room);
            else if (room.getGuest() != null &&
room.getGuest().getUsername().equals(""))
                roomsList.add(room);
        }
        Collections.sort(roomsList);
```

```
        for (Room room : roomsList) {
            add((new RoomPanel(room, new BookingClicked())));
        }
        allRooms = roomsList;
    }
    public void reDisplayRooms() {
        removeAll();
        NavPanel nav =
(NavPanel)((JComponent)getParent().getParent().getParent().getParent().getCompo
nent(0)).getComponent(0);
        JSlider lowerCostSlider =
((PriceBar)nav.getComponent(1)).getCostSlider();
        JSlider upperCostSlider =
((PriceBar)nav.getComponent(1)).getAssociatedSlider();
        DropDownPanel capacityPanel = (DropDownPanel)nav.getComponent(3);
        DropDownPanel roomTypePanel = (DropDownPanel)nav.getComponent(4);

        ArrayList<Room> rooms = new ArrayList<>(allRooms);
        ArrayList<Room> removeList = new ArrayList<>();
        for (Room room : rooms) {
            //System.out.println(roomTypePanel.getCurrentSelection().replace("
", "") +"??" + room.getType() + " " +
capacityPanel.getCurrentSelection().equals("ALL"));
            if (!(lowerCostSlider.getValue() <= room.getCost() &&
room.getCost() <= upperCostSlider.getValue()))
                removeList.add(room);
            if (!(capacityPanel.getCurrentSelection().equals("ALL")) &&
(Integer.parseInt(capacityPanel.getCurrentSelection()) != room.getCapacity()))
                removeList.add(room);
            if (!(roomTypePanel.getCurrentSelection().replace(" ",
"").equals(room.getType())) &&
!(roomTypePanel.getCurrentSelection().equals("ALL")))
                removeList.add(room);
            /*else if (*/
        }

        rooms.removeAll(removeList);

        ArrayList<Room> roomsList = new ArrayList<>();
        for (Room room : rooms) {
            if (room.getGuest() == null)
                roomsList.add(room);
            else if (room.getGuest() != null &&
room.getGuest().getUsername().equals(""))
```

```
                roomsList.add(room);
            }
            Collections.sort(roomsList);
            for (Room room : roomsList) {
                add((new RoomPanel(room, new BookingClicked())));
            }

            revalidate();
            repaint();
        }

        public HashSet<Room> getBookedRooms() {
            return bookedRooms;
        }

        private class BookingClicked implements ActionListener {

            @Override
            public void actionPerformed(ActionEvent e) {
                if (((JButton)e.getSource()).getForeground().equals(Color.black)) {

bookedRooms.add(((RoomPanel)((JButton)e.getSource()).getParent()).getRoom());
                    ((JButton) e.getSource()).setForeground(Color.green);
                }
                else if
(((JButton)e.getSource()).getForeground().equals(Color.green)) {

bookedRooms.remove(((RoomPanel)((JButton)e.getSource()).getParent()).getRoom())
;
                    ((JButton) e.getSource()).setForeground(Color.black);
                }
            }
        }
    }

// USEAMENITYPAGEPANEL CLASS:
package screencomponents.pages;

import accessors.Guest;
import database.MongoModifier;
import hotelcomponents.Amenity;
import hotelcomponents.RateBased;
import hotelexceptions.GuestNotFoundException;
import screencomponents.AppDefaultButton;
```

```
import screencomponents.ScreenPanel;
import stringoperations.StringModifier;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalTime;

public class UseAmenityPagePanel<T extends Amenity & RateBased> extends JPanel
{
    private T amenity;
    private JComboBox<Integer> hoursBox;
    public UseAmenityPagePanel(T amenity) {

        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new GridBagLayout());

        this.amenity = amenity;
    }

    @Override
    public void addNotify() {
        super.addNotify();

        ScreenPanel screenPanel = (ScreenPanel) getParent();
        boolean isFree = false;
        boolean invalidTime = false;
        switch(StringModifier.addSpace(amenity.getClass().getSimpleName())) {
            case "Gym" :
                if
(screenPanel.getAssociatedGuest().getMostValubleRoom().hasFreeGymAccess())
                    isFree = true;
                break;
            case "Game Room" :
                if
(screenPanel.getAssociatedGuest().getMostValubleRoom().hasFreeGameRoomAccess())
                    isFree = true;
                break;
            case "Swimming Pool" :
```

```
                if
(screenPanel.getAssociatedGuest().getMostValubleRoom().hasFreeSwimmingPoolAcces
s())
                    isFree = true;
                break;
        }

        if (LocalTime.now().isBefore(amenity.getOpeningTime()) ||
LocalTime.now().isAfter(amenity.getClosingTime())) {
            add(new JLabel("This Amenity is closed at this hour.") {{
                setFont(new Font("Times New Roman", Font.PLAIN, 40));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.red);
            }});
            add(new JLabel("Access Times are from : " +
amenity.getOpeningTime() + " to " + amenity.getClosingTime()) {{
                setFont(new Font("Times New Roman", Font.PLAIN, 30));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.red);
            }}, new GridBagConstraints() {{
                gridy = 1;
            }});
        }
        else if(isFree) {
            add(new JLabel("YOU MAY FREELY ACCESS THIS AMENITY!") {{
                setFont(new Font("Times New Roman", Font.PLAIN, 40));
                setHorizontalAlignment(JLabel.CENTER);
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.green);
            }});
        }
        else {
            JPanel centerPanel = new JPanel();
            centerPanel.setBackground(Color.white);
            centerPanel.setBorder(BorderFactory.createLineBorder(Color.black,
4));
            centerPanel.setLayout(new GridLayout(3, 1));

            centerPanel.add(new
JLabel(StringModifier.addSpace(amenity.getClass().getSimpleName())) {{
                setFont(new Font("Times New Roman", Font.BOLD, 40));
                setHorizontalAlignment(JLabel.CENTER);
```

```
                setVerticalAlignment(JLabel.CENTER);
                setForeground(Color.black);
            }});

            centerPanel.add(new JPanel() {{
                setBackground(Color.white);
                setLayout(new GridBagLayout());
                add(new JLabel("Hours of Use: ") {{
                    setFont(new Font("Times New Roman", Font.PLAIN, 25));
                    setHorizontalAlignment(JLabel.CENTER);
                    setVerticalAlignment(JLabel.CENTER);
                    setForeground(Color.black);
                }}, new GridBagConstraints());


                add(hoursBox = new JComboBox<Integer>(new Integer[]{1, 2, 3, 4,
5, 6}), new GridBagConstraints() {{
                    gridx = 1;
                }});
            }});

            centerPanel.add(new JPanel() {{
                setBackground(Color.white);
                setLayout(new GridBagLayout());
                add(new AppDefaultButton("USE") {{
                    addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            try {
                                MongoModifier mongo = new MongoModifier();
                                Guest guest =
mongo.getGuest(screenPanel.getAssociatedUsername());
                                amenity.utilize(guest,
(int)hoursBox.getSelectedItem());
                                screenPanel.setAssociatedGuest(guest);

                                screenPanel.remove(1);
                                screenPanel.add(new
UtilizeAmenitiesPagePanel(), BorderLayout.CENTER);
                                screenPanel.revalidate();
                                screenPanel.repaint();
                            }
                            catch (GuestNotFoundException
guestNotFoundException) {
```

```
                                guestNotFoundException.printStackTrace();
                            }
                        }
                    });
                }}, new GridBagConstraints());
            }});
            add(centerPanel, new GridBagConstraints());
        }
    }
}

// USERDETAILSPAGEPANEL CLASS:
package screencomponents.pages;

import accessors.User;
import database.MongoModifier;
import hotelexceptions.UserNotFoundException;
import screencomponents.ScreenPanel;

import javax.swing.*;
import java.awt.*;

public class UserDetailsPagePanel extends JPanel {

    public UserDetailsPagePanel(String username) {
        setBackground(Color.white);

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setLayout(new GridBagLayout());
    }

    @Override
    public void addNotify() {
        super.addNotify();
        JPanel userDetailsPanel = new JPanel();
        userDetailsPanel.setBackground(Color.white);
        userDetailsPanel.setLayout(new GridLayout(0, 1));
        userDetailsPanel.setBorder(BorderFactory.createLineBorder(Color.black,
7));
        User user = ((ScreenPanel)getParent()).getAssociatedUser();

        userDetailsPanel.add(new JLabel("Username: " + user.getUsername()) {{
```

```
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.PLAIN, 50));
            setHorizontalAlignment(JLabel.LEFT);
            setVerticalAlignment(JLabel.CENTER);
            setBorder(BorderFactory.createMatteBorder(0, 0, 7, 0,
Color.black));
        }});
        userDetailsPanel.add(new JLabel("Name: " + user.getName()) {{
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.PLAIN, 50));
            setHorizontalAlignment(JLabel.LEFT);
            setVerticalAlignment(JLabel.CENTER);
            setBorder(BorderFactory.createMatteBorder(0, 0, 7, 0,
Color.black));
        }});
        userDetailsPanel.add(new JLabel("Phone Number: " +
user.getPhoneNumber()) {{
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.PLAIN, 50));
            setHorizontalAlignment(JLabel.LEFT);
            setVerticalAlignment(JLabel.CENTER);
            setBorder(BorderFactory.createMatteBorder(0, 0, 7, 0,
Color.black));
        }});
        userDetailsPanel.add(new JLabel("Email Address: " + user.getEmailId())
{{
            setForeground(Color.black);
            setFont(new Font("Times New Roman", Font.PLAIN, 50));
            setHorizontalAlignment(JLabel.LEFT);
            setVerticalAlignment(JLabel.CENTER);
        }});

        add(userDetailsPanel, new GridBagConstraints());
    }
}

// UTILIZEAMENITIESPAGEPANEL CLASS:
package screencomponents.pages;

import hotelcomponents.GameRoom;
import hotelcomponents.Gym;
import hotelcomponents.SwimmingPool;
import layouts.WrapLayout;
import screencomponents.AppDefaultButton;
```

```
import screencomponents.ScreenPanel;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.time.LocalTime;

public class UtilizeAmenitiesPagePanel extends JPanel {

    public UtilizeAmenitiesPagePanel() {
        setBackground(Color.white);
        setLayout(new BorderLayout());

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));

        JPanel buttonsPanel = new JPanel();
        buttonsPanel.setBackground(Color.white);
        buttonsPanel.setLayout(new WrapLayout(WrapLayout.CENTER, 500, 80));

        buttonsPanel.add(new AppDefaultButton("GAME ROOM") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 60));

        }});
        buttonsPanel.add(new AppDefaultButton("SWIMMING POOL") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 60));
        }});
        buttonsPanel.add(new AppDefaultButton("GYM") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 60));
        }});
        buttonsPanel.add(new AppDefaultButton("RESTAURANT") {{
            setFont(new Font("Times New Roman", Font.PLAIN, 60));
        }});

        for (Component button : buttonsPanel.getComponents()) {
            ((JButton)button).addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    ScreenPanel screenPanel =
(ScreenPanel)UtilizeAmenitiesPagePanel.this.getParent();
                    switch(((AppDefaultButton)button).getOriginalText()) {
                        case "GYM" :
```

```
                            screenPanel.remove(1);
                            screenPanel.add(new UseAmenityPagePanel<Gym>(new
Gym(LocalTime.of(7, 0),
                                    LocalTime.of(20, 0), 30, 500)));
                            break;
                        case "GAME ROOM" :
                            screenPanel.remove(1);
                            screenPanel.add(new
UseAmenityPagePanel<GameRoom>(new GameRoom(LocalTime.of(11, 0),
                                    LocalTime.of(22, 0), 30, 600)));
                            break;
                        case "SWIMMING POOL" :
                            screenPanel.remove(1);
                            screenPanel.add(new
UseAmenityPagePanel<SwimmingPool>(new SwimmingPool(LocalTime.of(9, 0),
                                    LocalTime.of(21, 0), 30, 350)));
                            break;
                    }
                    screenPanel.revalidate();
                    screenPanel.repaint();
                }
            });
        }
        add(buttonsPanel, BorderLayout.CENTER);
    }
}

// PACKAGE SCREEENCOMPONENTS:

// APPDEFAULTBUTTON CLASS:
package screencomponents;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class AppDefaultButton extends JButton {
    private String originalText;
    public AppDefaultButton(String text) {
        super("\n " + text + " \n");
        originalText = text;
        setUI(new BasicButtonUI());
```

```
        setBackground(Color.white);
        setForeground(Color.black);
        setFont(new Font("Times New Roman", Font.PLAIN, 30));
        setFocusPainted(false);
        setBorder(BorderFactory.createLineBorder(Color.black, 4));
        addMouseListener(new MouseListener(this));
    }
    public AppDefaultButton() {
        setUI(new BasicButtonUI());
        setBackground(Color.white);
        setForeground(Color.black);
        setFont(new Font("Times New Roman", Font.PLAIN, 30));
        setFocusPainted(false);
        addMouseListener(new MouseListener(this));
    }

    public String getOriginalText() {
        return originalText;
    }

    private class MouseListener extends MouseAdapter {
        private JButton jb;
        MouseListener(JButton jb){
            this.jb = jb;
        }
        @Override
        public void mouseEntered(MouseEvent e) {
            jb.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        }
        @Override
        public void mouseExited(MouseEvent e) {
            jb.setCursor(Cursor.getDefaultCursor());
        }
    }

}

// LOGINPANEL CLASS:
package screencomponents;

import accessors.User;
import database.MongoModifier;
import hotelcomponents.rooms.Room;
import hotelexceptions.GuestNotFoundException;
```

```java
import hotelexceptions.IllegalPasswordAccessException;
import hotelexceptions.PreexisistingRoomException;
import hotelexceptions.UserNotFoundException;
import payments.PDFCreator;
import screencomponents.admincomponents.AddRoomPagePanel;
import screencomponents.admincomponents.CheckFeedbackPagePanel;
import screencomponents.admincomponents.RemoveRoomPagePanel;
import screencomponents.constraints.DropDownPanel;
import screencomponents.constraints.PriceBar;
import screencomponents.pages.*;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.*;
import java.io.FileNotFoundException;
import java.util.HashSet;

public class LoginPanel extends JPanel {
    protected JTextField usernameField;
    protected JPasswordField passwordField;
    protected JLabel loginTitle;
    protected JLabel errorLabel;
    protected JButton enterButton;

    protected GridBagConstraints c = new GridBagConstraints();

    private static final String ADMIN_USERNAME = "ADMIN";
    private static final String ADMIN_PASSWORD = "adpass1";

    public LoginPanel() {
        setLayout(new GridBagLayout());
        setBorder(BorderFactory.createLineBorder(Color.black, 5));
        setBackground(Color.white);

        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.insets = new Insets(20, 40, 0, 40);
        c.anchor = GridBagConstraints.CENTER;

        loginTitle = new JLabel("LOGIN");
```

```
loginTitle.setBackground(Color.white);
loginTitle.setForeground(Color.black);
loginTitle.setHorizontalAlignment(JLabel.CENTER);
loginTitle.setVerticalAlignment(JLabel.CENTER);
loginTitle.setFont(new Font("Times New Roman", Font.BOLD, 50));
add(loginTitle, c);

c.gridx = 0;
c.gridy = 1;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

JLabel usernameText = new JLabel("Username: ");
usernameText.setForeground(Color.black);
usernameText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(usernameText, c);

c.gridx = 1;
c.gridy = 1;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

usernameField = new JTextField(10);
usernameField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
usernameField.addActionListener(new CredentialsHandler());
add(usernameField, c);

c.gridx = 0;
c.gridy = 2;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

JLabel passwordText = new JLabel("Password: ");
passwordText.setForeground(Color.black);
passwordText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
```

```
add(passwordText, c);

c.gridx = 1;
c.gridy = 2;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

passwordField = new JPasswordField(10);
passwordField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
passwordField.addActionListener(new CredentialsHandler());
add(passwordField, c);

c.gridx = 0;
c.gridy = 7;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 2;
c.insets = new Insets(20, 20, 20, 20);
c.anchor = GridBagConstraints.CENTER;

enterButton = new JButton("ENTER");
enterButton.setUI(new BasicButtonUI());
enterButton.setForeground(Color.black);
enterButton.setBackground(Color.white);
enterButton.setFocusPainted(false);

enterButton.setFont((new Font("Times New Roman", Font.PLAIN, 20)));
enterButton.addActionListener(new CredentialsHandler());
enterButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseEntered(MouseEvent e) {
        setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    }

    @Override
    public void mouseExited(MouseEvent e) {
        setCursor(Cursor.getDefaultCursor());
    }
});
add(enterButton, c);
```

```
        c.gridx = 0;
        c.gridy = 8;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.insets = new Insets(0, 20, 5, 20);
        c.anchor = GridBagConstraints.CENTER;

        errorLabel = new JLabel("Username or Password is Incorrect");
        errorLabel.setForeground(Color.red);
        errorLabel.setFont(new Font("Times New Roman", Font.PLAIN, 15));
        errorLabel.setHorizontalAlignment(JLabel.CENTER);
        errorLabel.setVerticalAlignment(JLabel.CENTER);
        errorLabel.setVisible(false);
        add(errorLabel, c);

    }
    private class CredentialsHandler implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            usernameField.setText("");
            passwordField.setText("");

            MongoModifier mongo = new MongoModifier();
            ScreenPanel screenPanel = (ScreenPanel) getParent().getParent();

            if (username.equals(ADMIN_USERNAME) &&
password.equals(ADMIN_PASSWORD)) {
                screenPanel.remove(1);
                screenPanel.add(new AddRoomPagePanel());
                screenPanel.revalidate();
                screenPanel.repaint();

                ((NavPanel)screenPanel.getComponent(0)).removeAll();
                ((NavPanel)screenPanel.getComponent(0)).add(new NavButton("Add
Rooms") {{
                    setFont(new Font("Times New Roman", Font.PLAIN, 25));
                    addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            screenPanel.remove(1);
```

```
                        screenPanel.add(new AddRoomPagePanel());
                        screenPanel.revalidate();
                        screenPanel.repaint();
                    }
                });
            }});

            ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Remove Rooms") {{
                setFont(new Font("Times New Roman", Font.PLAIN, 25));
                addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        screenPanel.remove(1);
                        screenPanel.add(new JScrollPane(new
RemoveRoomPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{
                            getVerticalScrollBar().setUnitIncrement(10);
                        }}, BorderLayout.CENTER);
                        screenPanel.revalidate();
                        screenPanel.repaint();
                    }
                });
            }});

            ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Check Feedback") {{
                setFont(new Font("Times New Roman", Font.PLAIN, 25));
                addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        screenPanel.remove(1);
                        screenPanel.add(new JScrollPane(new
CheckFeedbackPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{
                            getVerticalScrollBar().setUnitIncrement(10);
                        }}, BorderLayout.CENTER);
                        screenPanel.revalidate();
                        screenPanel.repaint();
                    }
                });
            }});
```

```
                ((NavPanel)screenPanel.getComponent(0)).add(new NavButton("Sign
Out") {{
                    addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                            JFrame window =
(JFrame)SwingUtilities.getWindowAncestor(screenPanel);
                            window.getContentPane().removeAll();
                            window.add(new ScreenPanel(), BorderLayout.CENTER);
                            window.revalidate();
                            window.repaint();
                        }
                    });
                }});
                return;
            }
            for (User user : mongo.getAllUsers()) {
                if (user.getUsername().equals(username)) {
                    try {
                        if (mongo.getPassword(username,
"awZYbHxLl4WykOhSiujjJm7U9VuEYx2X").equals(password)) {
                            try {

screenPanel.setAssociatedGuest(mongo.getGuest(username));
                                screenPanel.setAssociatedUsername(username);

screenPanel.setAssociatedUser(screenPanel.getAssociatedGuest());
                                screenPanel.remove(1);
                                screenPanel.add(new
UtilizeAmenitiesPagePanel());
                                screenPanel.revalidate();
                                screenPanel.repaint();


((NavPanel)screenPanel.getComponent(0)).removeAll();
                                ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Utilize Amenities") {{
                                    setFont(new Font("Times New Roman",
Font.PLAIN, 25));
                                    addActionListener(new ActionListener() {
                                        @Override
                                        public void actionPerformed(ActionEvent
e) {
                                            screenPanel.remove(1);
```

```
                                        screenPanel.add(new
UtilizeAmenitiesPagePanel());
                                        screenPanel.revalidate();
                                        screenPanel.repaint();
                                    }
                                });
                            }});

                            ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Give Feedback") {{
                                setFont(new Font("Times New Roman",
Font.PLAIN, 25));
                                addActionListener(new ActionListener() {
                                    @Override
                                    public void actionPerformed(ActionEvent
e) {
                                        screenPanel.remove(1);
                                        screenPanel.add(new
FeedbackPagePanel());
                                        screenPanel.revalidate();
                                        screenPanel.repaint();
                                    }
                                });
                            }});

                            ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Check Out") {{
                                addActionListener(new ActionListener() {
                                    @Override
                                    public void actionPerformed(ActionEvent
e) {
                                        if
(((JButton)e.getSource()).getText().equals("Check Out")) {

((JButton)e.getSource()).setText("Confirm?");

((JButton)e.getSource()).setBackground(Color.red);
                                        }
                                        else {
                                            PDFCreator pdf = new
PDFCreator();
                                        try {

pdf.createPDF(screenPanel.getAssociatedGuest());
```

```
                                                }
                                                catch (FileNotFoundException
fileNotFoundException) {

fileNotFoundException.printStackTrace();
                                                }
                                                MongoModifier mongo = new
MongoModifier();

mongo.removeBill(screenPanel.getAssociatedGuest());

mongo.removeGuest(screenPanel.getAssociatedGuest());
                                                for (Room room :
screenPanel.getAssociatedGuest().getRooms()) {
                                                    mongo.removeRoom(room);
                                                    try {
                                                        mongo.addRoom(room);
                                                    }
                                                    catch
(PreexisistingRoomException preexisistingRoomException) {

preexisistingRoomException.printStackTrace();
                                                    }
                                                }
                                                System.exit(0);
                                            }
                                        }
                                    });
                                }});

                                ((NavPanel)screenPanel.getComponent(0)).add(new
NavButton("Sign Out") {{
                                    addActionListener(new ActionListener() {
                                        @Override
                                        public void actionPerformed(ActionEvent
e) {
                                            JFrame window =
(JFrame)SwingUtilities.getWindowAncestor(screenPanel);

window.getContentPane().removeAll();
                                            window.add(new ScreenPanel(),
BorderLayout.CENTER);
                                            window.revalidate();
                                            window.repaint();
```

```
                                    }
                                });
                            }});
                            return;
                        }
                        catch (GuestNotFoundException ex) {

                        }
                        screenPanel.remove(1);
                        ((JPanel)screenPanel.getComponent(0)).removeAll();
                        ((JPanel)screenPanel.getComponent(0)).add(new
NavButton("Search") {{
                            addActionListener(new ActionListener() {
                                @Override
                                public void actionPerformed(ActionEvent e)
{

//System.out.println(((JScrollPane)(((JComponent)(e.getSource())).getParent().g
etParent()).getComponent(1)).getViewport().getView());
                                    try {
                                        ((SearchPagePanel) ((JScrollPane)
(((JComponent)
(e.getSource()))).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).reDisplayRooms();
                                        ((SearchPagePanel) ((JScrollPane)
(((JComponent)
(e.getSource()))).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).revalidate();
                                        ((SearchPagePanel) ((JScrollPane)
(((JComponent)
(e.getSource()))).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).repaint();
                                    }
                                    catch (ClassCastException ex) {
                                        ex.printStackTrace();
                                        screenPanel.remove(1);
                                        screenPanel.add(new JScrollPane(new
SearchPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{

getVerticalScrollBar().setUnitIncrement(10);
                                        }}, BorderLayout.CENTER);
                                        ((SearchPagePanel) ((JScrollPane)
(((JComponent)
```

```
(e.getSource())).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).reDisplayRooms();
                                                ((SearchPagePanel) ((JScrollPane)
(((JComponent)
(e.getSource())).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).revalidate();
                                                ((SearchPagePanel) ((JScrollPane)
(((JComponent)
(e.getSource())).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).repaint();
                                            }
                                        }
                                    });
                                }});
                                PriceBar pbl = new PriceBar(PriceBar.LOWER);
                                PriceBar pbh = new PriceBar(PriceBar.UPPER);
                                pbl.setAssociatedSlider(pbh.getCostSlider());
                                pbh.setAssociatedSlider(pbl.getCostSlider());
                                ((JPanel)screenPanel.getComponent(0)).add(pbl);
                                ((JPanel)screenPanel.getComponent(0)).add(pbh);
                                ((JPanel)screenPanel.getComponent(0)).add(new
DropDownPanel<String>("Capacity", new String[]{"ALL", "1", "2", "3", "4", "5",
"6"}));
                                ((JPanel)screenPanel.getComponent(0)).add(new
DropDownPanel<String>("Room  ", new String[]{"ALL", "Single Room", "Deluxe
Room", "Double Room", "Executive Room", "Apartment Room", "Cabana Room", "King
Room", "Queen Room"}));
                                //System.out.println("HIIII");
                                screenPanel.add(new JScrollPane(new
SearchPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{
                                    getVerticalScrollBar().setUnitIncrement(10);
                                }}, BorderLayout.CENTER);

                                ((JPanel)screenPanel.getComponent(0)).add(new
NavButton("User Details") {{
                                    addActionListener(new ActionListener() {
                                        @Override
                                        public void actionPerformed(ActionEvent e)
{
                                            if
(((JButton)e.getSource()).getText().equals("User Details")) {
                                                screenPanel.remove(1);
```

```
                                        screenPanel.add(new
UserDetailsPagePanel(username));
                                        ((JButton)
e.getSource()).setText("Close Details");
                                    }
                                    else {
                                        screenPanel.remove(1);
                                        screenPanel.add(new JScrollPane(new
SearchPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{

getVerticalScrollBar().setUnitIncrement(10);
                                        }}, BorderLayout.CENTER);
                                        ((JButton)
e.getSource()).setText("User Details");
                                    }
                                }
                            });
                        }});

                        ((JPanel)screenPanel.getComponent(0)).add(new
NavButton("Confirm Booking") {{
                            setFont(new Font("Times New Roman", Font.PLAIN,
24));
                            addActionListener(new ActionListener() {
                                @Override
                                public void actionPerformed(ActionEvent e)
{
                                    if
(((JButton)e.getSource()).getText().equals("Confirm Booking")) {
                                        HashSet<Room> rooms =
((SearchPagePanel) ((JScrollPane) (((JComponent)
(e.getSource()))).getParent().getParent()).getComponent(1)).getViewport().getVie
w()).getBookedRooms();
                                        screenPanel.remove(1);
                                        screenPanel.add(new
ConfirmBookingPanel(rooms));
                                        ((JButton)
e.getSource()).setText("Close");
                                    }
                                    else {
                                        screenPanel.remove(1);
```

```
                                        screenPanel.add(new JScrollPane(new
SearchPagePanel(), JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER) {{

getVerticalScrollBar().setUnitIncrement(10);
                                        }}, BorderLayout.CENTER);
                                        ((JButton)
e.getSource()).setText("Confirm Booking");
                                    }
                                }
                            });
                        }});

                        ((JPanel)screenPanel.getComponent(0)).add(new
NavButton("Sign Out") {{
                            setBorder(null);
                            addActionListener(new ActionListener() {
                                @Override
                                public void actionPerformed(ActionEvent e)
{
                                    JFrame window =
(JFrame)SwingUtilities.getWindowAncestor(screenPanel);
                                    window.getContentPane().removeAll();
                                    window.add(new ScreenPanel(),
BorderLayout.CENTER);
                                    window.revalidate();
                                    window.repaint();
                                }
                            });
                        }});

                        screenPanel.revalidate();
                        screenPanel.repaint();

((ScreenPanel)screenPanel).setAssociatedUsername(username);

screenPanel.setAssociatedUser(mongo.getUser(username));
                    }
                }
                catch (IllegalPasswordAccessException |
UserNotFoundException ex) {
                    System.out.println(ex);

                }
```

```
                }
                else {
                    errorLabel.setVisible(true);
                }
            }

        }
    }
}

// MAIN CLASS:
package screencomponents;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main {

    private JTextPane pleaseLoginText;
    private JFrame window;
    private JPanel screenPanel;
    private JPanel navPanel;
    private JPanel searchPanel;
    private JPanel loginPagePanel;

    private SignUpPanel signUpPanel;


    private Font hugeFont = new Font("Times New Roman", Font.PLAIN, 90);
    private Font largeFont = new Font("Times New Roman", Font.PLAIN, 60);
    private Font mediumFont = new Font("Times New Roman", Font.PLAIN, 45);
    private Font smallFont = new Font("Times New Roman", Font.PLAIN, 30);
    private Font littleFont = new Font("Times New Roman", Font.PLAIN, 24);
    private Font vsmallFont = new Font("Times New Roman", Font.PLAIN, 20);
    private Font tinyFont = new Font("Times New Roman", Font.PLAIN, 15);

    public Main() {
        window = new JFrame();
        window.setSize(1200, 800);
        window.setTitle("CheckInnMate");
        //window.setResizable(false);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        window.setLocationRelativeTo(null);
        window.getContentPane().setBackground(Color.white);
        window.add(new ScreenPanel(), BorderLayout.CENTER);
        window.setVisible(true);
    }


    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Main();
            }
        });
    }
}

// NAVBUTTON CLASS:
package screencomponents;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class NavButton extends JButton {
    public NavButton (String content){
        setText(content);
        setUI(new BasicButtonUI());
        setBackground(Color.white);
        setForeground(Color.black);
        setFont(new Font("Times New Roman", Font.PLAIN, 30));
        setFocusPainted(false);
        setBorder(BorderFactory.createMatteBorder(0, 0, 4, 0, Color.black));
        addMouseListener(new MouseListener(this));
    }
    private class MouseListener extends MouseAdapter {
        private JButton jb;
        MouseListener(JButton jb){
            this.jb = jb;
        }
        @Override
        public void mouseEntered(MouseEvent e) {
```

```
            jb.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        }
        @Override
        public void mouseExited(MouseEvent e) {
            jb.setCursor(Cursor.getDefaultCursor());
        }
    }

}

// NAVPANEL CLASS:
package screencomponents;

import screencomponents.pages.LoginPagePanel;
import screencomponents.pages.SearchPagePanel;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class NavPanel extends JPanel {
    private NavButton searchPageButton;
    private NavButton userDetailsPageButton;
    private NavButton utilizeAmenitiesButton;
    private NavButton checkOutButton;
    private NavButton signOutButton;
    private NavButton signUpButton;
    private NavButton loginButton;
    private NavButton modifyRoomsPageButton;
    private JPanel screenPanel;

    private Font smallFont = new Font("Times New Roman", Font.PLAIN, 30);
    private Font littleFont = new Font("Times New Roman", Font.PLAIN, 24);

    public NavPanel(JPanel screenPanel) {

setBorder(BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(8,
8, 8, 8),
                BorderFactory.createLineBorder(Color.black, 4)));
        setBackground(Color.white);
        setPreferredSize(new Dimension(200, 133));
        setLayout(new GridLayout(8, 1));
        this.screenPanel = screenPanel;
```

```
        signUpButton = new NavButton("Sign Up");
        signUpButton.addActionListener(new NavButtonClicked());
        add(signUpButton);

        loginButton = new NavButton("Login");
        loginButton.addActionListener(new NavButtonClicked());
        //navPanel.add(loginButton);
    }

    private class NavButtonClicked implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            switch (((NavButton)e.getSource()).getText()) {
                case "Sign Up" :

((JComponent)screenPanel.getComponent(1)).getComponent(0).setVisible(false);

((JComponent)screenPanel.getComponent(1)).getComponent(1).setVisible(true);
                    remove(signUpButton);
                    add(loginButton);
                    revalidate();
                    repaint();
                    break;
                case "Login" :

((JComponent)screenPanel.getComponent(1)).getComponent(0).setVisible(true);

((JComponent)screenPanel.getComponent(1)).getComponent(1).setVisible(false);
                    remove(loginButton);
                    add(signUpButton);
                    revalidate();
                    repaint();
                    break;
                default:
                    System.out.println("ILLEGAL OPTION");
                    break;
            }
        }
    }
}

// ROOMPANEL CLASS:
package screencomponents;
```

```
import hotelcomponents.rooms.Room;
import hotelcomponents.rooms.SingleRoom;

import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class RoomPanel extends JPanel {

    private Font roomFont = new Font("Times New Roman", Font.PLAIN, 25);
    private JButton bookButton, detailsButton;
    private JLabel imageLabel, capacityLabel, areaLabel;
    private JPanel gapPanel1, gapPanel2, gapPanel3;
    private Room room;

    public RoomPanel(Room room, ActionListener action) {
        this.room = room;
        setBackground(Color.white);
        setLayout(new GridBagLayout());
        setBorder(BorderFactory.createLineBorder(Color.black, 5));
        //setPreferredSize(new Dimension(0, 400));
        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 2;
        c.gridheight = 1;
        c.insets = new Insets(-1,-1, 1, -1);
        c.anchor = GridBagConstraints.CENTER;

        System.out.println(room.getType());
        Image img = new ImageIcon(RoomPanel.class.getResource("/RoomImages/" +
room.getType() + ".jpg")).getImage();
        add(imageLabel = new JLabel(new ImageIcon(img.getScaledInstance(400,
200, Image.SCALE_SMOOTH))), c);

        c.gridy = 1;
        c.anchor = GridBagConstraints.WEST;
```

```
        c.insets = new Insets(-1,-1, 0, -1);

        add(gapPanel1 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 2;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel typeLabel = new JLabel("Room: " +
room.getClass().getSimpleName());
        typeLabel.setForeground(Color.black);
        typeLabel.setFont(roomFont);
        typeLabel.setHorizontalAlignment(JLabel.LEFT);
        typeLabel.setVerticalAlignment(JLabel.CENTER);

        add(typeLabel, c);

        c.gridy = 3;
        c.insets = new Insets(-1, -1, 0, -1);

        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 4;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel roomNoLabel = new JLabel("Room No: " + room.getRoomId());
        roomNoLabel.setForeground(Color.black);
        roomNoLabel.setFont(roomFont);
        roomNoLabel.setHorizontalAlignment(JLabel.LEFT);
        roomNoLabel.setVerticalAlignment(JLabel.CENTER);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(roomNoLabel, c);

        c.gridy = 5;
        c.insets = new Insets(-1, -1, 0, -1);
```

```
        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 6;
        c.insets = new Insets(-1, 5, 0, -1);

        JLabel roomCostLabel = new JLabel("Cost Per Night: \u20B9" +
(int)room.getCost());
        roomCostLabel.setForeground(Color.black);
        roomCostLabel.setFont(roomFont);
        roomCostLabel.setHorizontalAlignment(JLabel.LEFT);
        roomCostLabel.setVerticalAlignment(JLabel.CENTER);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(roomCostLabel, c);

        c.gridy = 7;
        c.insets = new Insets(-1, -1, 0, -1);

        add(new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
        }}, c);

        c.gridx = 0;
        c.gridy = 8;
        c.insets = new Insets(-1, 5, 0, -1);

        capacityLabel = new JLabel("Capacity: " + room.getCapacity());
        capacityLabel.setForeground(Color.black);
        capacityLabel.setFont(roomFont);
        capacityLabel.setHorizontalAlignment(JLabel.LEFT);
        capacityLabel.setVerticalAlignment(JLabel.CENTER);
        capacityLabel.setVisible(false);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(capacityLabel, c);

        c.gridy = 9;
        c.insets = new Insets(-1, -1, 0, -1);
```

```
        add(gapPanel2 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
            setVisible(false);
        }}, c);

        c.gridx = 0;
        c.gridy = 10;
        c.insets = new Insets(-1, 5, 0, -1);

        areaLabel = new JLabel("Area: " + room.getArea() + "m\u00B2");
        areaLabel.setForeground(Color.black);
        areaLabel.setFont(roomFont);
        areaLabel.setHorizontalAlignment(JLabel.LEFT);
        areaLabel.setVerticalAlignment(JLabel.CENTER);
        areaLabel.setVisible(false);
        //typeLabel.setBorder(BorderFactory.createMatteBorder(0, 0, 5, 0,
Color.black);
        add(areaLabel, c);

        c.gridy = 11;
        c.insets = new Insets(-1, -1, 0, -1);

        add(gapPanel3 = new JPanel(){{
            setPreferredSize(new Dimension(400, 4));
            setBackground(Color.black);
            setVisible(false);
        }}, c);


        c.gridx = 0;
        c.gridy = 12;
        c.gridwidth = 1;
        c.insets = new Insets(-1, -1, -1, -1);
        c.fill = GridBagConstraints.BOTH;

        bookButton = new JButton("BOOK ROOM");
        bookButton.setUI(new BasicButtonUI());
        bookButton.setBackground(Color.white);
        bookButton.setForeground(Color.black);
        //bookButton.setBorderPainted(false);
        bookButton.setFocusPainted(false);
        bookButton.setFont(roomFont);
```

```
        bookButton.setBorder(BorderFactory.createMatteBorder(0, 0, 0, 4,
Color.black));
        bookButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {

bookButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                bookButton.setCursor(Cursor.getDefaultCursor());
            }
        });
        bookButton.addActionListener(action);
        add(bookButton, c);

        c.gridx = 1;
        c.gridy = 12;
        c.gridwidth = 1;
        c.insets = new Insets(-1, -1, -1, -1);
        c.fill = GridBagConstraints.BOTH;

        detailsButton = new JButton("GET DETAILS");
        detailsButton.setUI(new BasicButtonUI());
        detailsButton.setBackground(Color.white);
        detailsButton.setForeground(Color.black);
        detailsButton.setBorderPainted(false);
        detailsButton.setFocusPainted(false);
        detailsButton.setFont(roomFont);
        detailsButton.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {

detailsButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                detailsButton.setCursor(Cursor.getDefaultCursor());
            }
        });
        detailsButton.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                if (((JButton)e.getSource()).getText().equals("GET DETAILS")) {
                    imageLabel.setVisible(false);
                    capacityLabel.setVisible(true);
                    areaLabel.setVisible(true);
                    gapPanel1.setVisible(false);
                    gapPanel2.setVisible(true);
                    gapPanel3.setVisible(true);
                    ((JButton)e.getSource()).setText("CLOSE");
                }
                else {
                    imageLabel.setVisible(true);
                    capacityLabel.setVisible(false);
                    areaLabel.setVisible(false);
                    gapPanel1.setVisible(true);
                    gapPanel2.setVisible(false);
                    gapPanel3.setVisible(false);
                    ((JButton)e.getSource()).setText("GET DETAILS");
                }
            }
        });

        add(detailsButton, c);
    }

    public Room getRoom() {
        return room;
    }

    public void disableButtons() {
        bookButton.setVisible(false);
    }
    public static void main(String[] args) {
        JFrame window = new JFrame();
        window.setSize(800, 800);
        window.setLocationRelativeTo(null);
        window.setLayout(new FlowLayout());
        window.add(new RoomPanel(new SingleRoom(100, 2, 2000, 50, null),
null));
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);
    }
}
```

```java
// SCREENPANEL CLASS:
package screencomponents;

import accessors.Guest;
import accessors.User;
import screencomponents.pages.LoginPagePanel;

import javax.swing.*;
import java.awt.*;

public class ScreenPanel extends JPanel {
    private String associatedUsername;
    private Guest associatedGuest;
    private User associatedUser;

    public ScreenPanel() {
        setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
        setBackground(Color.white);
        setLayout(new BorderLayout());

        add(new NavPanel(this), BorderLayout.WEST);
        add(new LoginPagePanel(), BorderLayout.CENTER);
    }

    public void setAssociatedUsername(String associatedUsername) {
        this.associatedUsername = associatedUsername;
    }

    public String getAssociatedUsername() {
        return associatedUsername;
    }

    public void setAssociatedGuest(Guest associatedGuest) {
        this.associatedGuest = associatedGuest;
    }

    public Guest getAssociatedGuest() {
        return associatedGuest;
    }

    public User getAssociatedUser() {
        return associatedUser;
    }
```

```
    public void setAssociatedUser(User associatedUser) {
        this.associatedUser = associatedUser;
    }
}

// SIGNUPPANEL CLASS:
package screencomponents;

import accessors.User;
import database.MongoModifier;
import hotelexceptions.IllegalPasswordAccessException;
import hotelexceptions.PreexisistingUserException;
import hotelexceptions.UserNotFoundException;
import screencomponents.pages.LoginPagePanel;
import screencomponents.pages.SearchPagePanel;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.regex.Pattern;

public class SignUpPanel extends LoginPanel {
    private JPasswordField retypePasswordField;
    private JTextField nameField;
    private JTextField phoneNumberField;
    private JTextField emailField;

    private Pattern usernamePattern = Pattern.compile("^[\\w\\s]+$");
    private Pattern namePattern = Pattern.compile("^[a-zA-Z\\s]+$");
    private Pattern emailPattern =
Pattern.compile("^[\\w.+\\-]+@[\\w.+\\-]+$");
    private Pattern phoneNumberPattern =
Pattern.compile("^\\+\\d{1,3}\\s\\d{7,15}$");
    public SignUpPanel() {
        loginTitle.setText("SIGN UP");

        c.gridx = 0;
        c.gridy = 3;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(20, 20, 0, 20);
        c.anchor = GridBagConstraints.CENTER;
```

```
        JLabel retypePasswordLabel = new JLabel("Retype Password: ");
        retypePasswordLabel.setBackground(Color.white);
        retypePasswordLabel.setForeground(Color.black);
        retypePasswordLabel.setHorizontalAlignment(JLabel.CENTER);
        retypePasswordLabel.setVerticalAlignment(JLabel.CENTER);
        retypePasswordLabel.setFont(new Font("Times New Roman", Font.PLAIN,
20));
        add(retypePasswordLabel, c);

        c.gridx = 1;
        c.gridy = 3;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(20, 20, 0, 20);
        c.anchor = GridBagConstraints.CENTER;

        retypePasswordField = new JPasswordField(10);
        retypePasswordField.setFont(new Font("Times New Roman", Font.PLAIN,
20));
        retypePasswordField.addActionListener(new CredentialsHandler());
        add(retypePasswordField, c);

        c.gridx = 0;
        c.gridy = 4;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(20, 20, 0, 20);
        c.anchor = GridBagConstraints.CENTER;

        JLabel nameText = new JLabel("Full Name: ");
        nameText.setForeground(Color.black);
        nameText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        add(nameText, c);

        c.gridx = 1;
        c.gridy = 4;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(20, 20, 0, 20);
        c.anchor = GridBagConstraints.CENTER;
```

```
nameField = new JTextField(10);
nameField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
nameField.addActionListener(new CredentialsHandler());
add(nameField, c);

c.gridx = 0;
c.gridy = 5;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

JLabel phoneNumberText = new JLabel("Phone Number: ");
phoneNumberText.setForeground(Color.black);
phoneNumberText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(phoneNumberText, c);

c.gridx = 1;
c.gridy = 5;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

phoneNumberField = new JTextField(10);
phoneNumberField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
phoneNumberField.addActionListener(new CredentialsHandler());
add(phoneNumberField, c);

c.gridx = 0;
c.gridy = 6;
c.weightx = 1;
c.weighty = 1;
c.gridwidth = 1;
c.insets = new Insets(20, 20, 0, 20);
c.anchor = GridBagConstraints.CENTER;

JLabel emailText = new JLabel("Email: ");
emailText.setForeground(Color.black);
emailText.setFont(new Font("Times New Roman", Font.PLAIN, 20));
add(emailText, c);
```

```
        c.gridx = 1;
        c.gridy = 6;
        c.weightx = 1;
        c.weighty = 1;
        c.gridwidth = 1;
        c.insets = new Insets(20, 20, 0, 20);
        c.anchor = GridBagConstraints.CENTER;

        emailField = new JTextField(10);
        emailField.setFont(new Font("Times New Roman", Font.PLAIN, 20));
        emailField.addActionListener(new CredentialsHandler());
        add(emailField, c);

        //errorLabel.setText();
        enterButton.removeActionListener(enterButton.getActionListeners()[0]);
        enterButton.addActionListener(new CredentialsHandler());
        usernameField.addActionListener(new CredentialsHandler());
        passwordField.addActionListener(new CredentialsHandler());
        retypePasswordField.addActionListener(new CredentialsHandler());
    }
    private class CredentialsHandler implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            String retypePassword = new
String(retypePasswordField.getPassword());
            String name = nameField.getText();
            String phoneNumber = phoneNumberField.getText();
            String email = emailField.getText();

            MongoModifier mongo = new MongoModifier();

            if (!usernamePattern.matcher(username).find()) {
                errorLabel.setText("Invalid username : Only use letters,
digits, _ and spaces");
                errorLabel.setVisible(true);
            }
            else if (!password.equals(retypePassword)) {
                errorLabel.setText("Passwords do not match");
                errorLabel.setVisible(true);
            }
```

```
            else if (!namePattern.matcher(name).find()) {
                errorLabel.setText("Invalid Name : Only use letters and
spaces");
                errorLabel.setVisible(true);
            }
            else if (!phoneNumberPattern.matcher(phoneNumber).find()) {
                errorLabel.setText("Invalid Phone Number : Please also add
country code");
                errorLabel.setVisible(true);
            }
            else if (!emailPattern.matcher(email).find()) {
                errorLabel.setText("Invalid Email Format");
                errorLabel.setVisible(true);
            }
            else {
                try {
                    mongo.addUser(new User(username, name, phoneNumber, email),
password);
                }
                catch (PreexisistingUserException ex) {
                    errorLabel.setText("Username already exists");
                    errorLabel.setVisible(true);
                    return;
                }
                JPanel screenPanel = (JPanel)getParent().getParent();

((JComponent)screenPanel.getComponent(1)).getComponent(0).setVisible(true);

((JComponent)screenPanel.getComponent(1)).getComponent(1).setVisible(false);
                screenPanel.removeAll();
                screenPanel.add(new NavPanel(screenPanel), BorderLayout.WEST);
                screenPanel.add(new LoginPagePanel(), BorderLayout.CENTER);
                screenPanel.revalidate();
                screenPanel.repaint();
            }
        }
    }
}
```

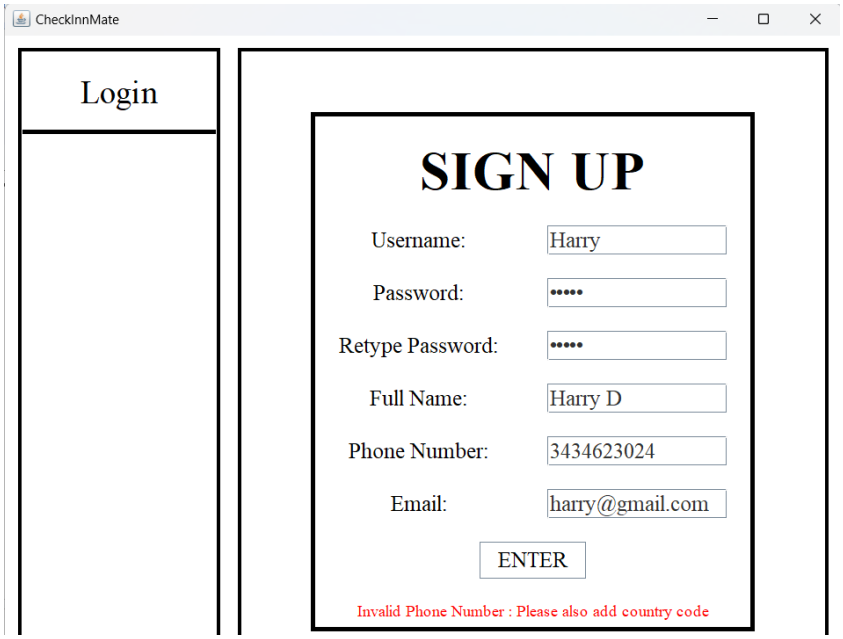
## PACKAGE STRING MODIFIER :

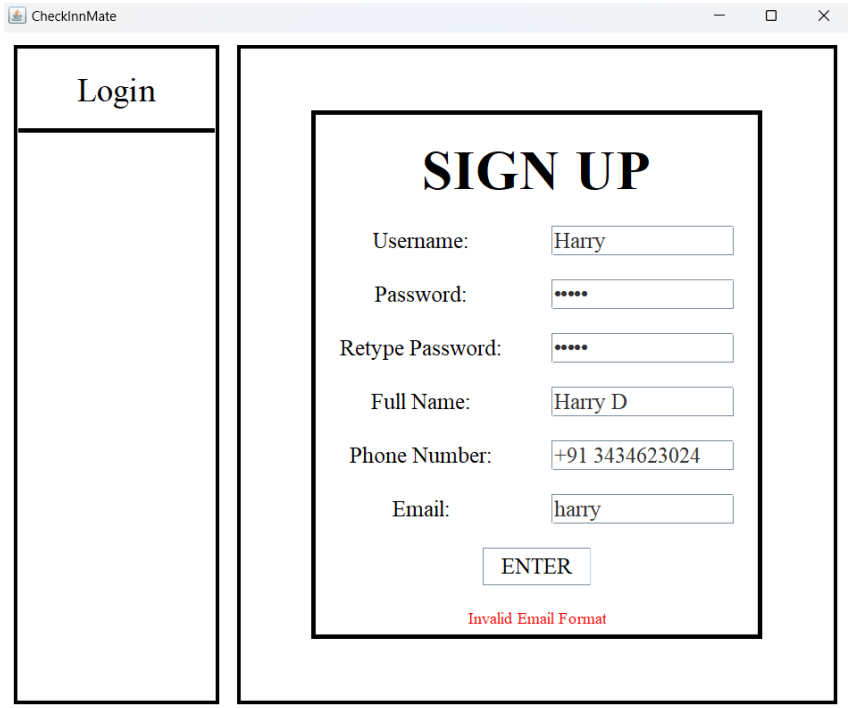
Java

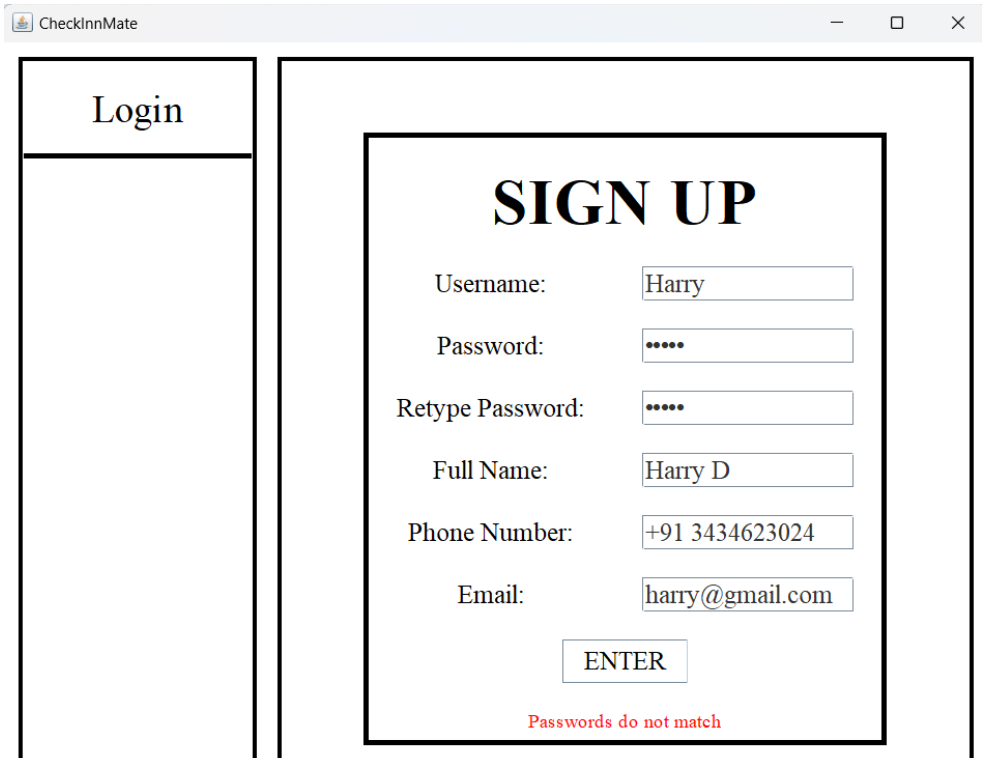
```java
// PACKAGE STRING MODIFIER


// STRINGMODIFIER CLASS:
package stringoperations;

public class StringModifier {
    public static String addSpace(String str) {
        for (int i = 1; i < str.length(); i++) {
            if (Character.isUpperCase(str.charAt(i))) {
                str = str.substring(0, i) + " " + str.substring(i);
                i++;
            }
        }
        return str;
    }

    public static String removeSpace(String str) {
        for (int i = 1; i < str.length(); i++) {
            if (str.charAt(i) == ' ') {
                str = str.substring(0, i) + str.substring(i + 1);
                i--;
            }
        }
        return str;
    }
}
```

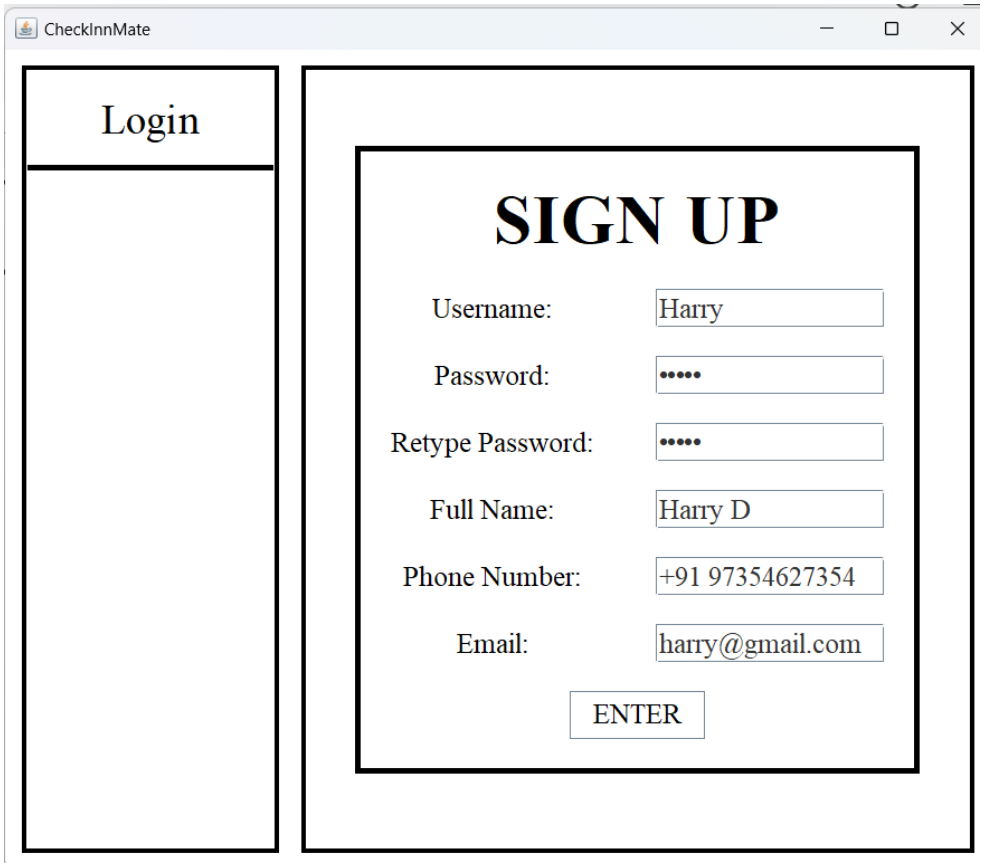
# VALIDATION

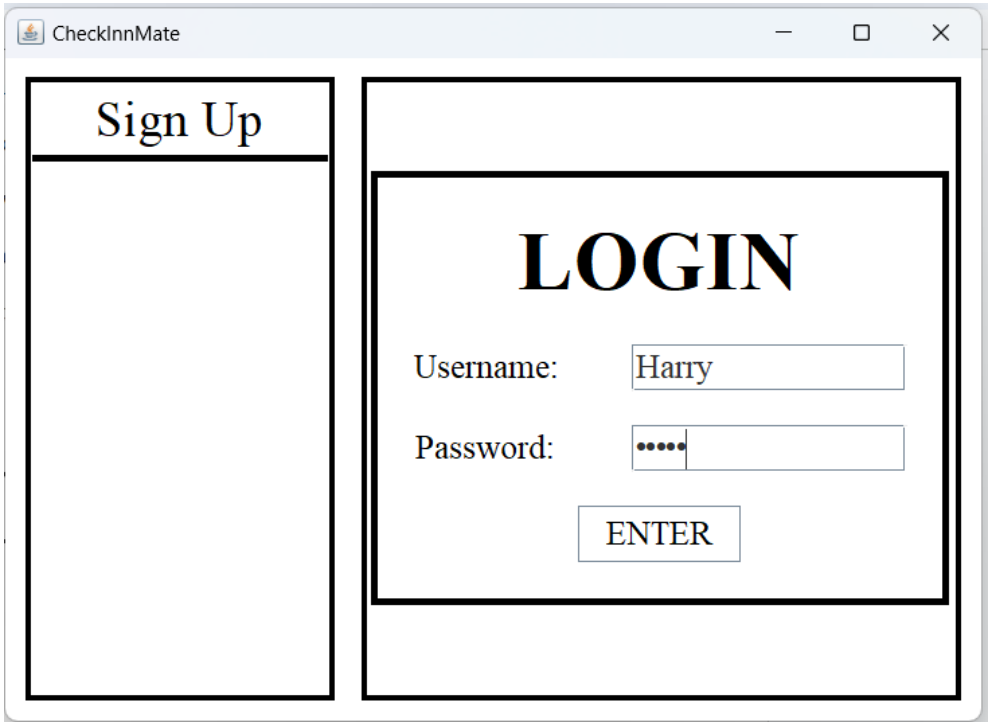
USER:

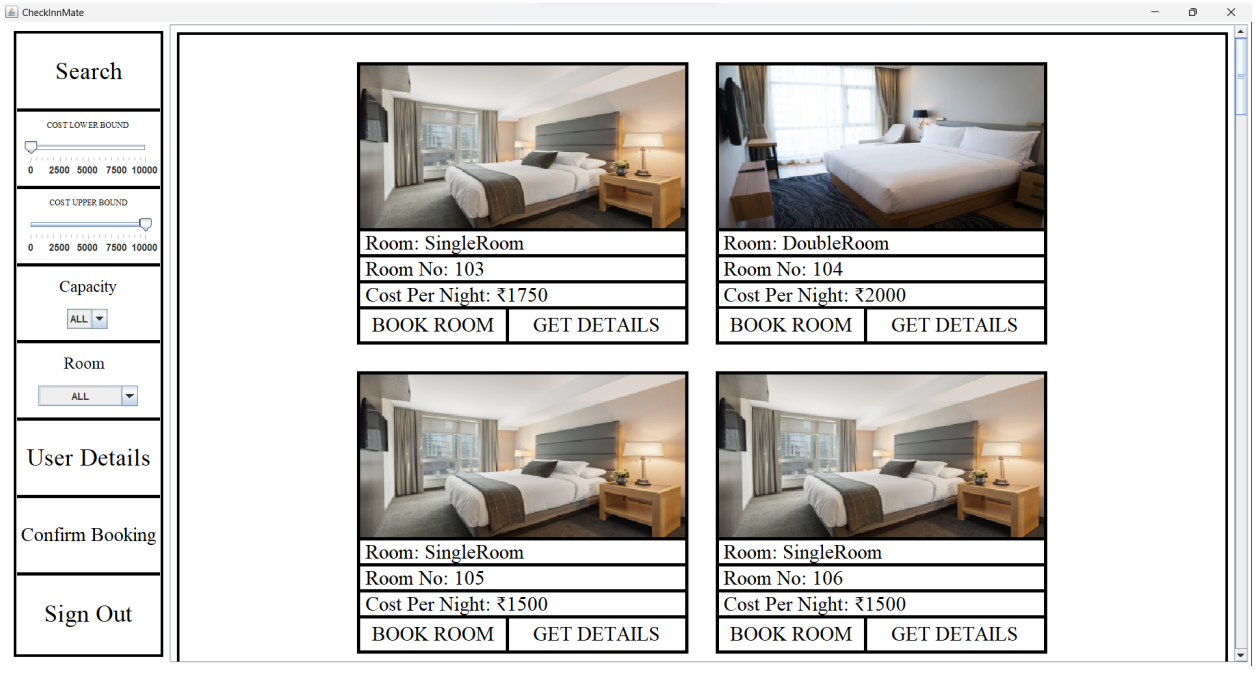


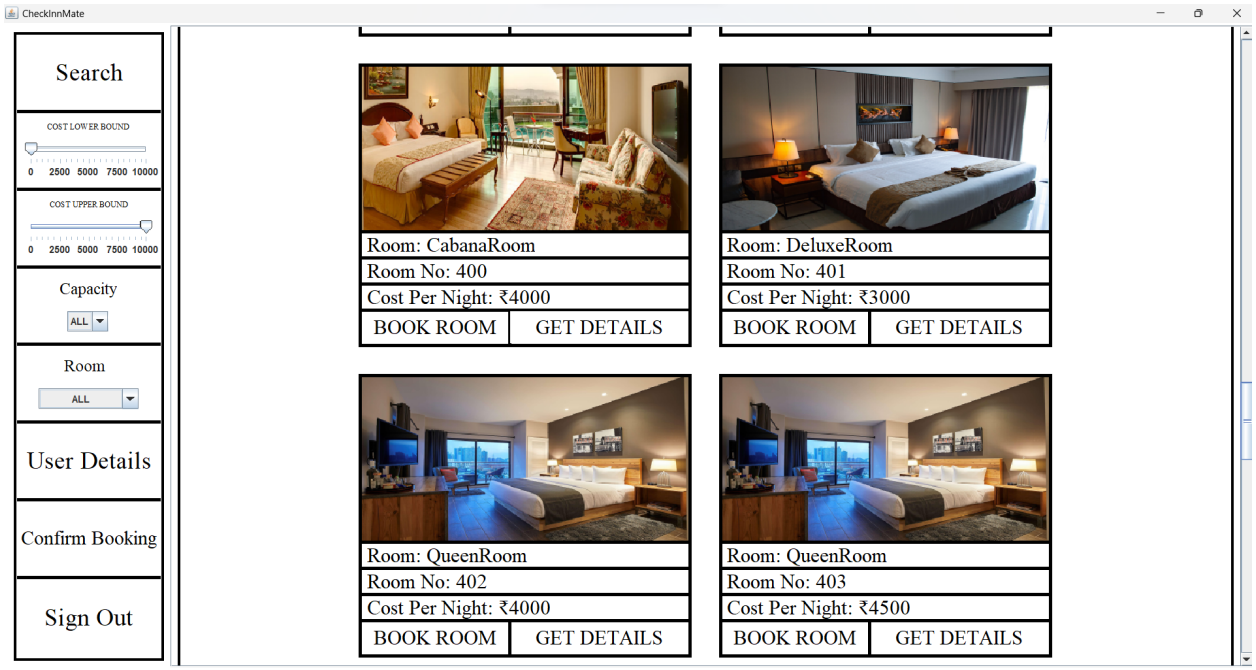



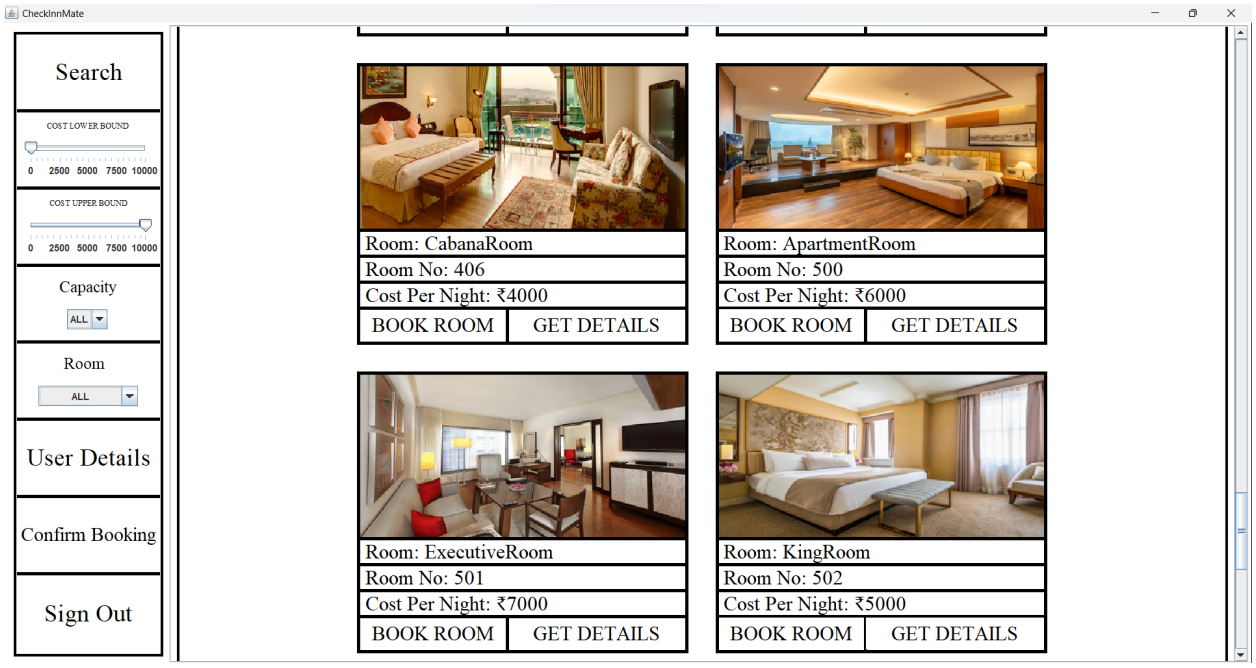

CheckInnMate
Login
SIGN UP
Username:
Harry
Password:
Retype Password:
Full Name:
Harry D
Phone Number:
+91 3434623024
Email:
harry@gmail.com
ENTER
Passwords do not match


CheckInnMate
Login
SIGN UP
Username:
Harry
Password:
Retype Password:
Full Name:
Harry D
Phone Number:
+91 97354627354
Email:
harry@gmail.com
ENTER

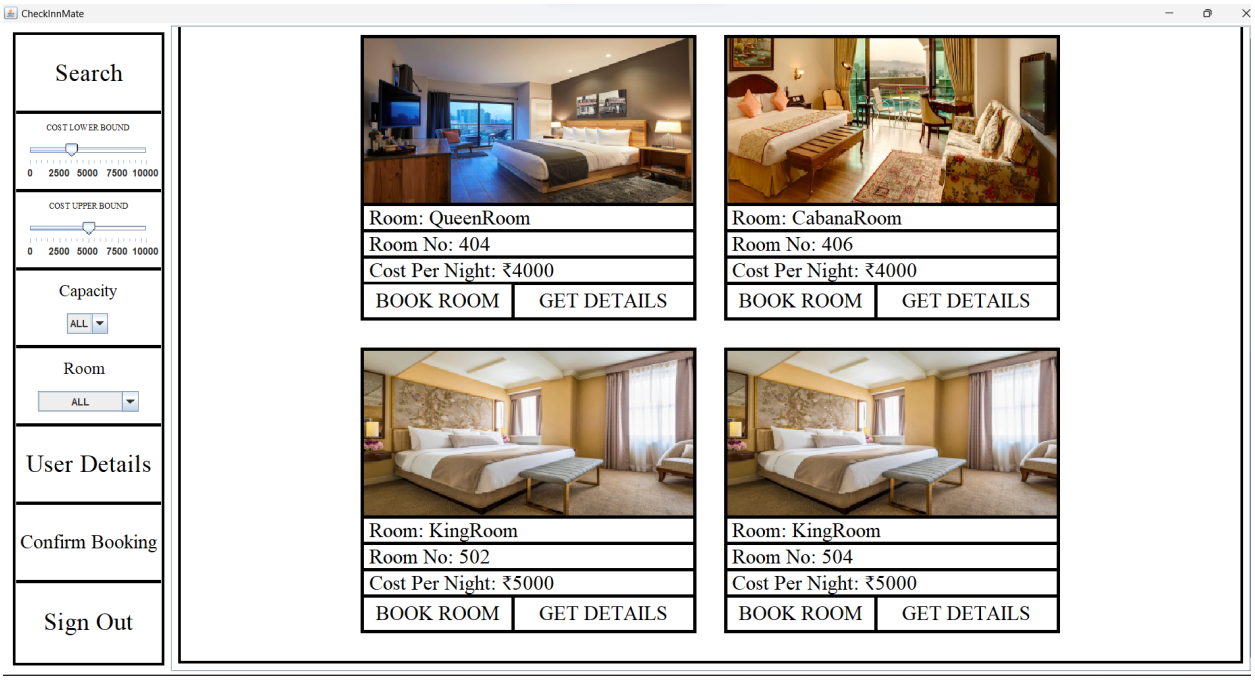

CheckInnMate
Sign Up
LOGIN
Username:
Harry
Password:
ENTER


CheckInnMate
Search
COST LOWER BOUND
0 2500 5000 7500 10000
COST UPPER BOUND
0 2500 5000 7500 10000
Capacity
ALL
Room
ALL
User Details
Confirm Booking
Sign Out
Room: SingleRoom
Room No: 103
Cost Per Night: ₹1750
BOOK ROOM
GET DETAILS
Room: DoubleRoom
Room No: 104
Cost Per Night: ₹2000
BOOK ROOM
GET DETAILS
Room: SingleRoom
Room No: 105
Cost Per Night: ₹1500
BOOK ROOM
GET DETAILS
Room: SingleRoom
Room No: 106
Cost Per Night: ₹1500
BOOK ROOM
GET DETAILS

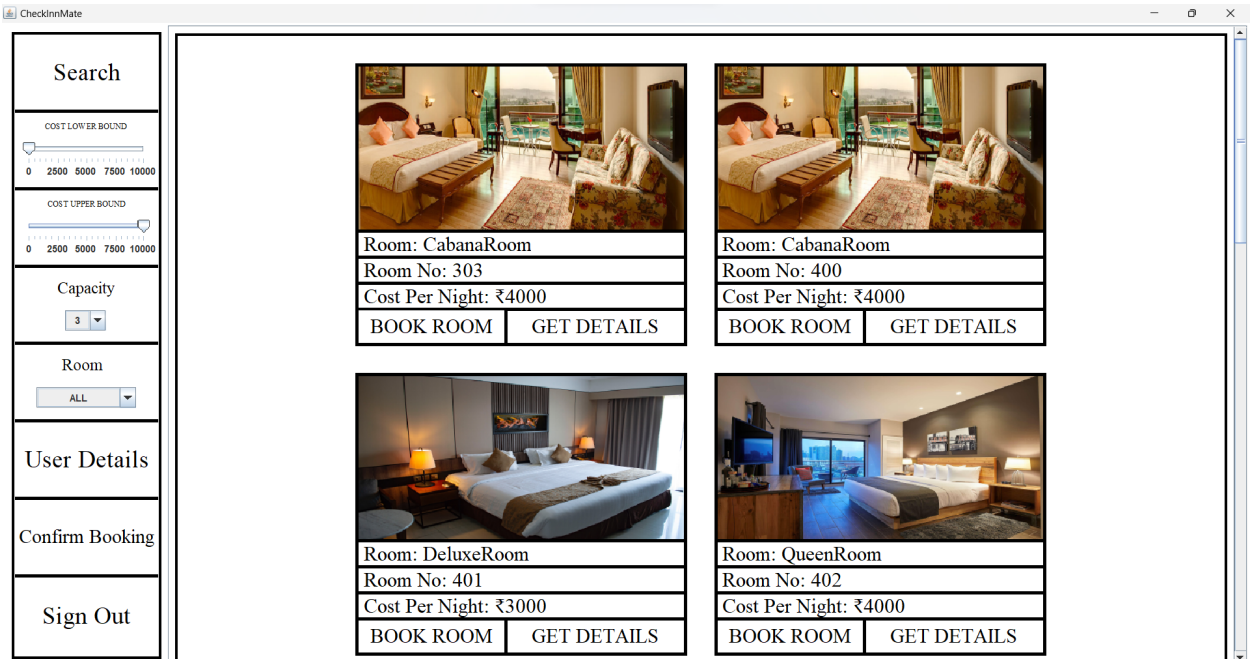



Searching on basis of price

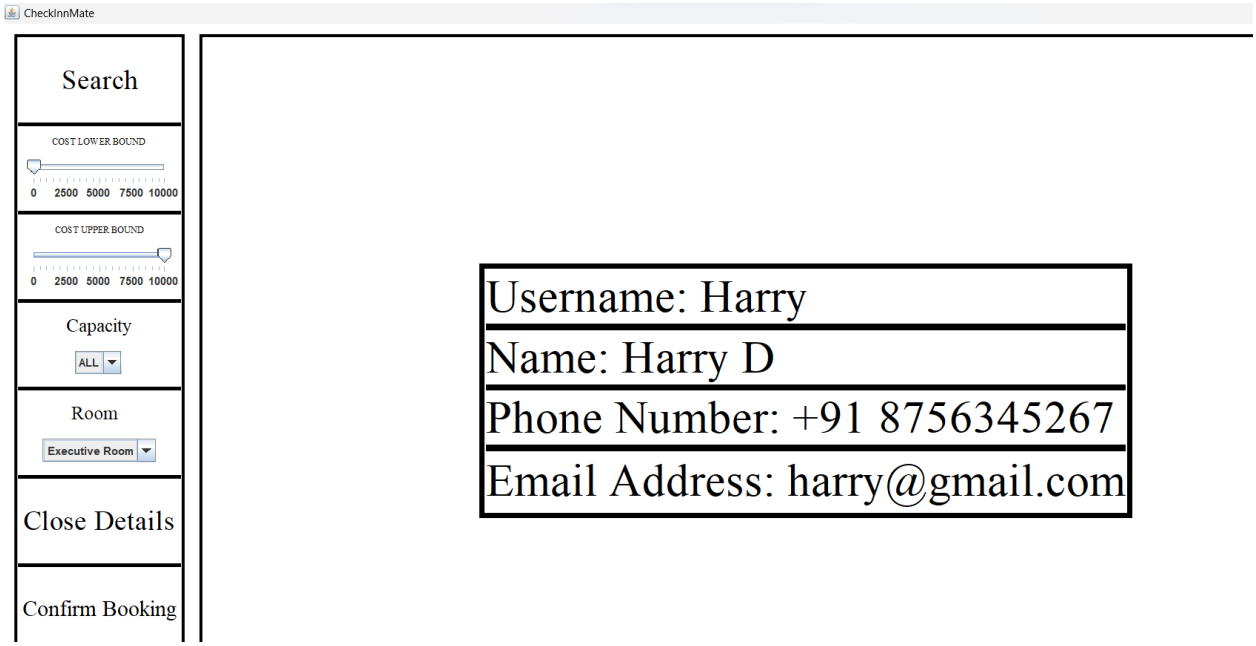



Searching on basis of capacity

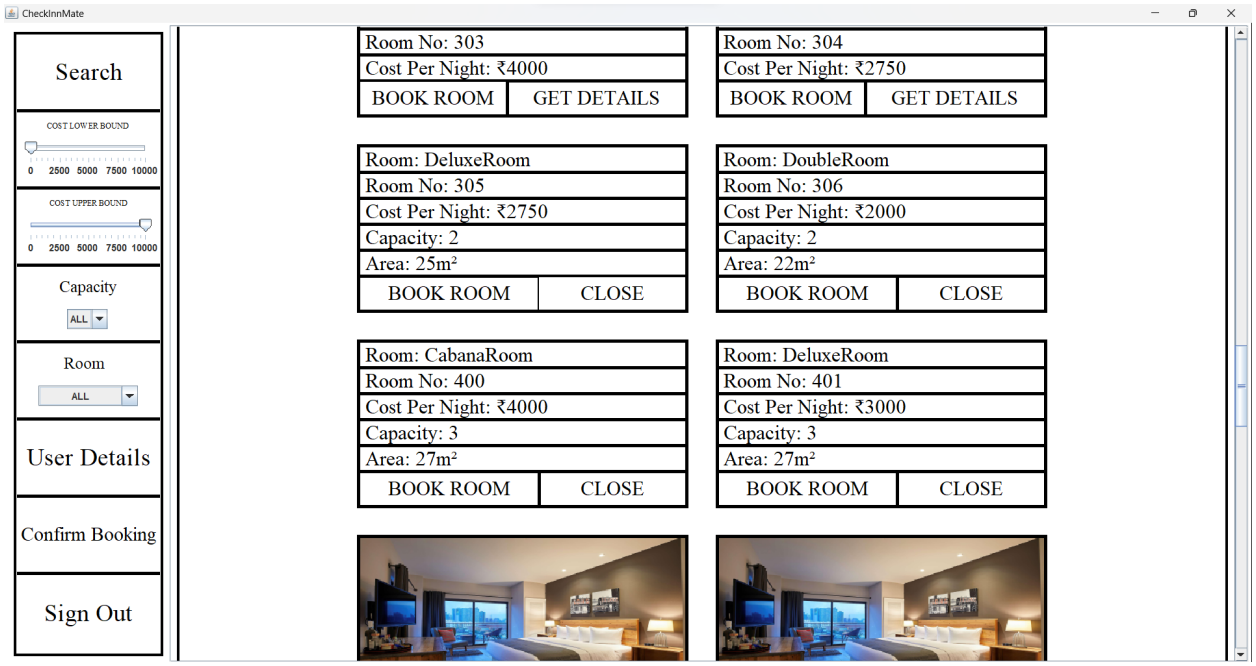



Searching on basis of type of room

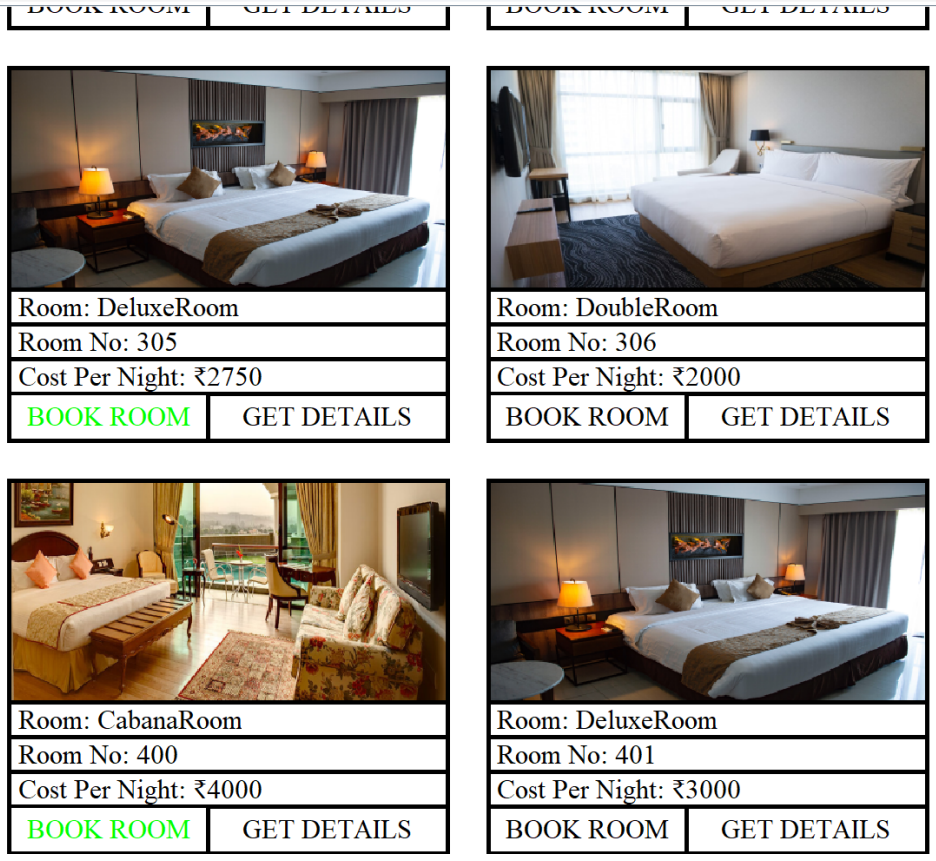

Checking user details



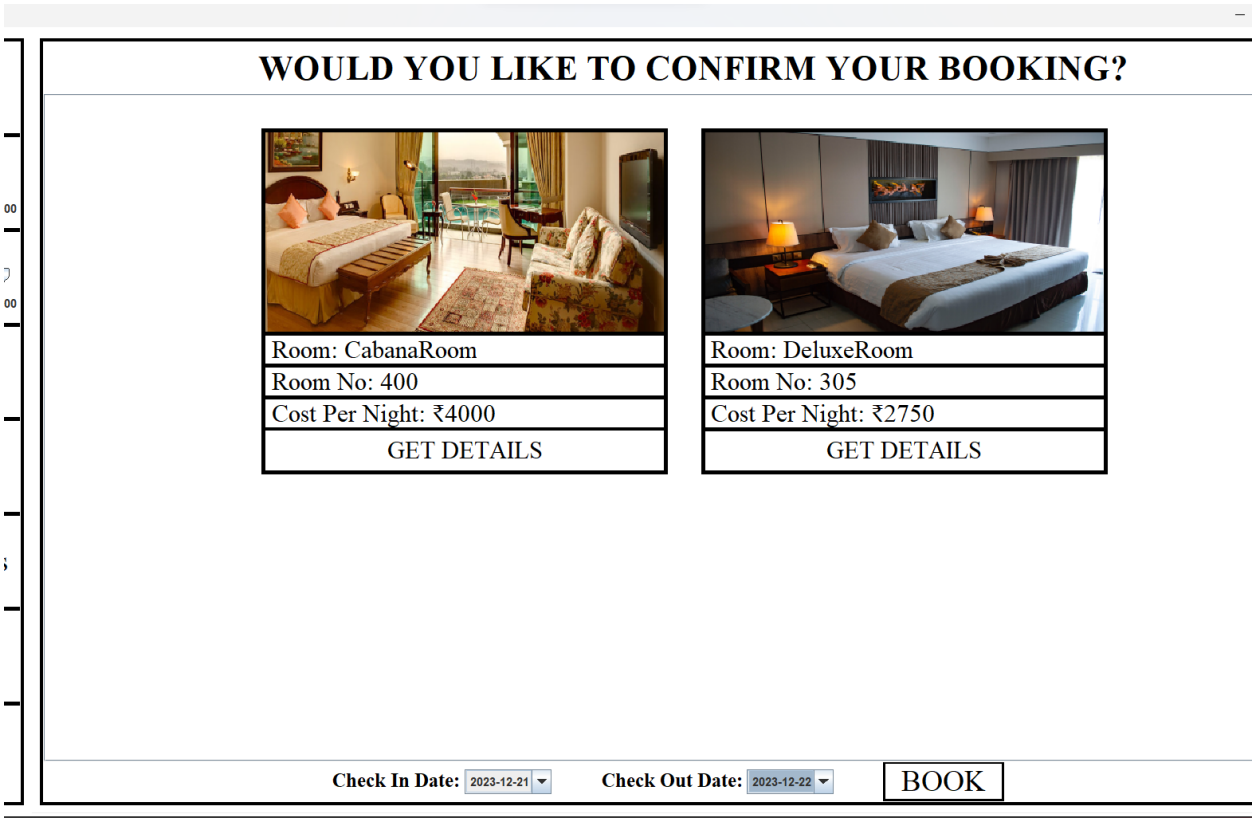

Check details of the room




Booking more than one room

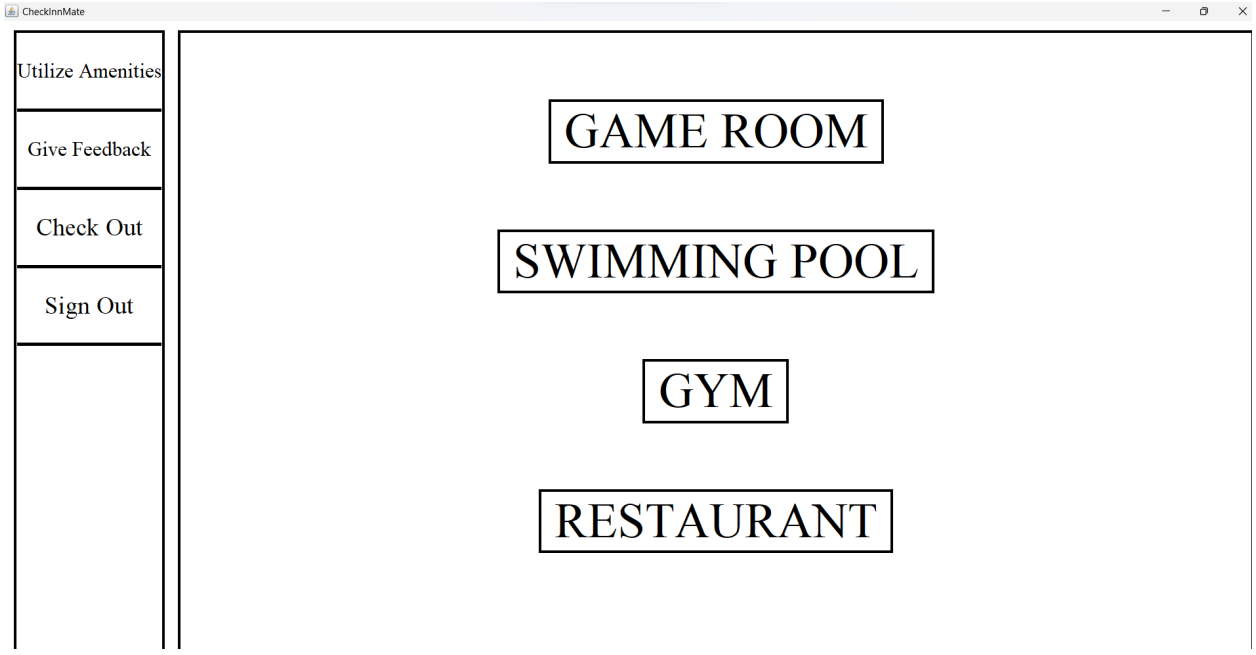



Confirming the booking of room

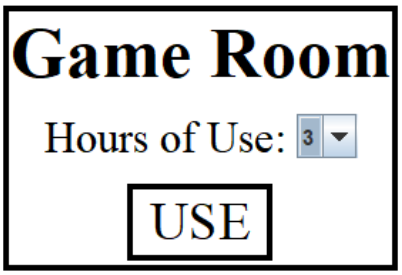



Once confirmed, the user is now a guest

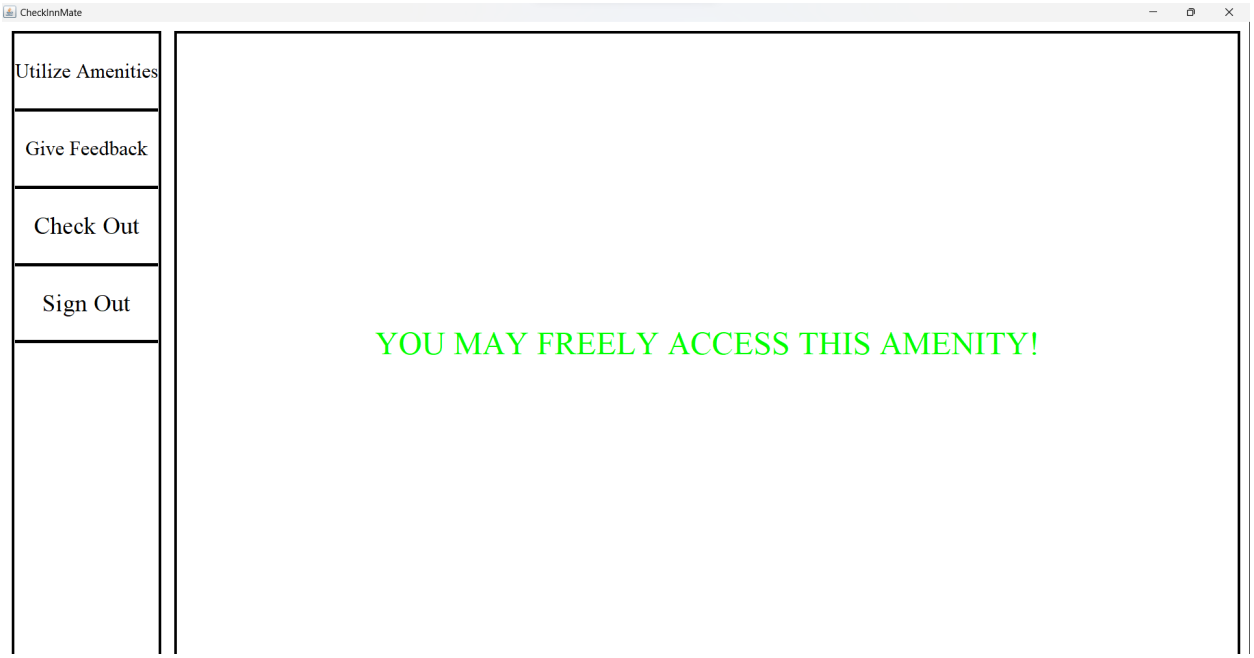
Amenity page:




Selecting hours of use ( if no free access)

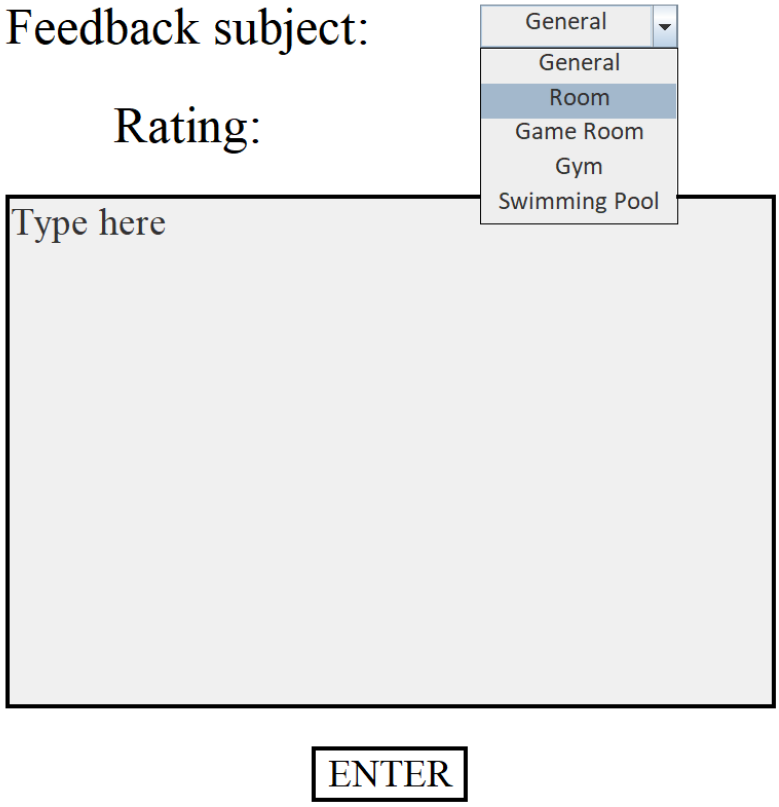



If you have free access to the amenity

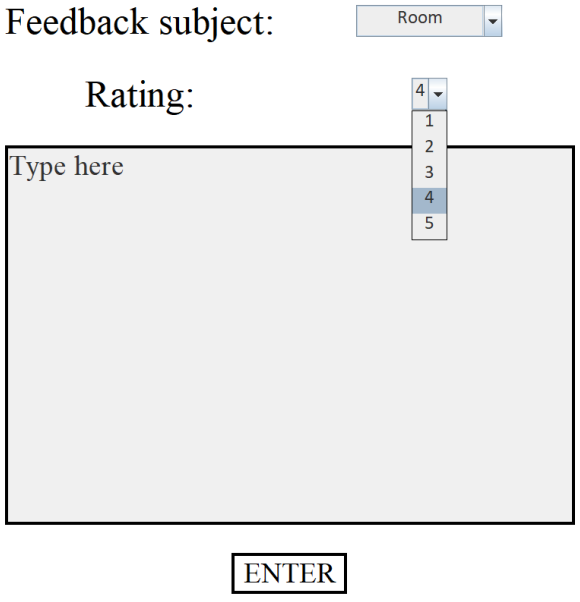



Give feedback:

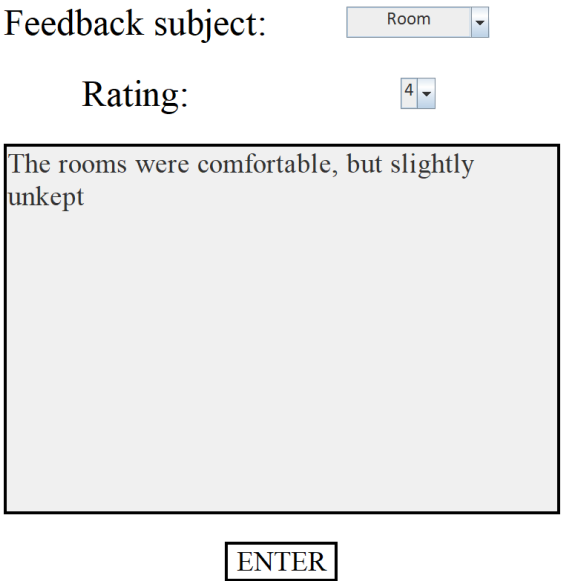
Select the feedback subject



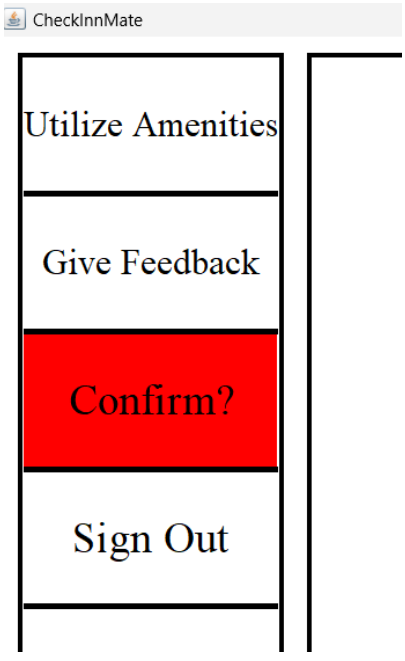

Give ratings




Give the reason for the feedback




Trying to checkout?




confirming checking out

The invoice generated

# CHECKINNMATE

Date: 2023-12-21

## INVOICE

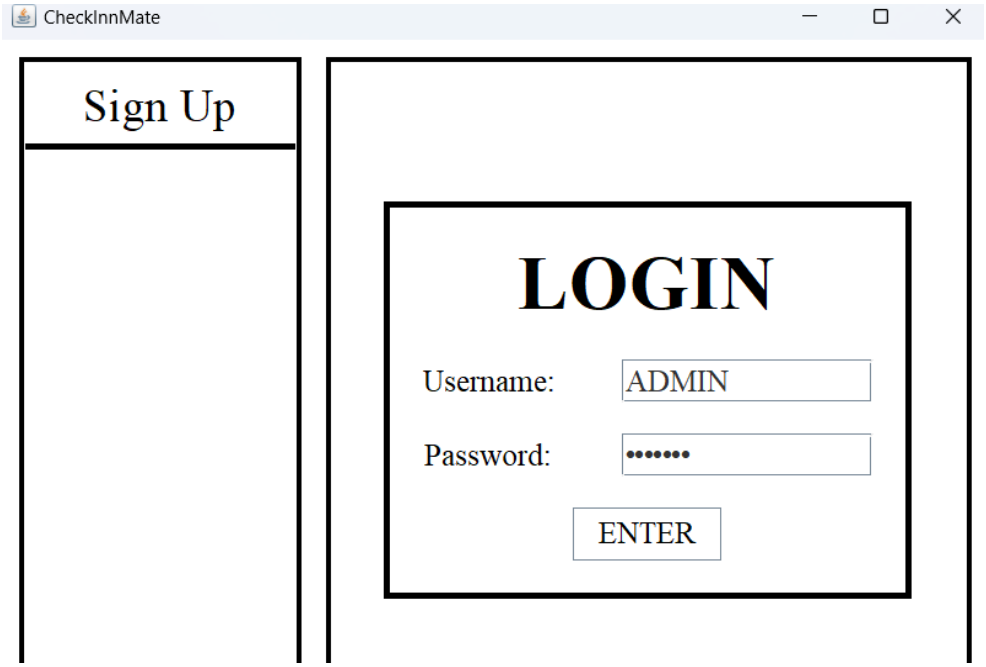
| Description | Quantity | Price |
|---|---|---|
| Cabana Room | 1 | 4000.0 |
| Deluxe Room | 1 | 2750.0 |
| Game Room x3 hours | 1 | 1800.0 |

**Total: 10350.0**

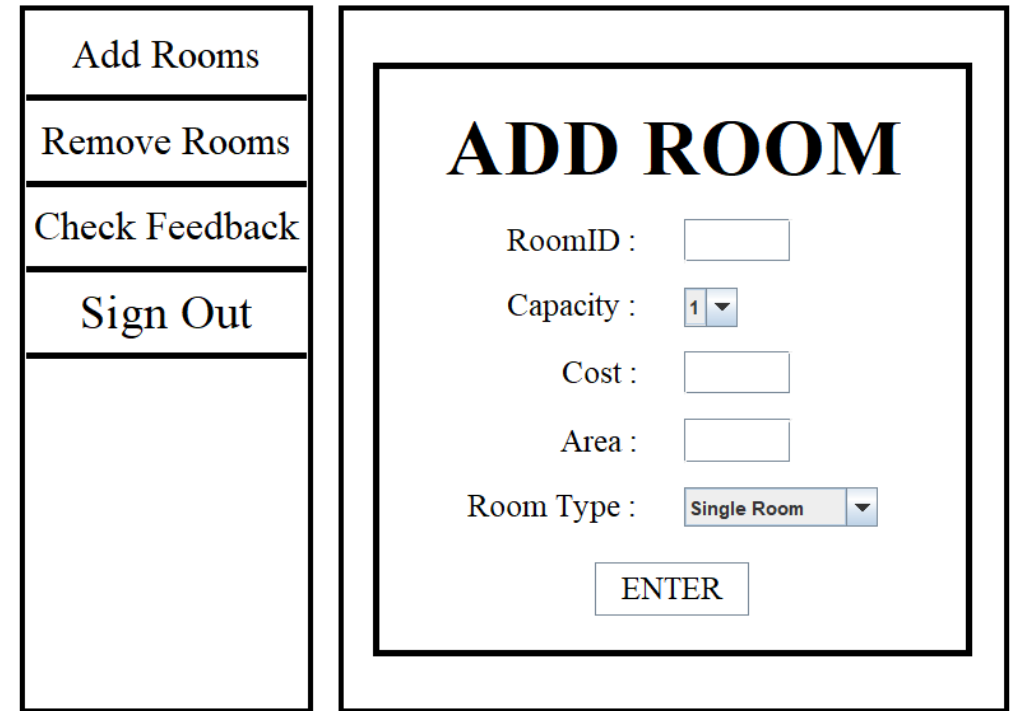
GST : 18%

**Final Bill 12213.0**

ADMIN:




Admin page




Adding room

# ADD ROOM

RoomID : 410

Capacity : 4

Cost : 4500

Area : 45

Room Type : Cabana Room

ENTER

Room Added Successfully!!

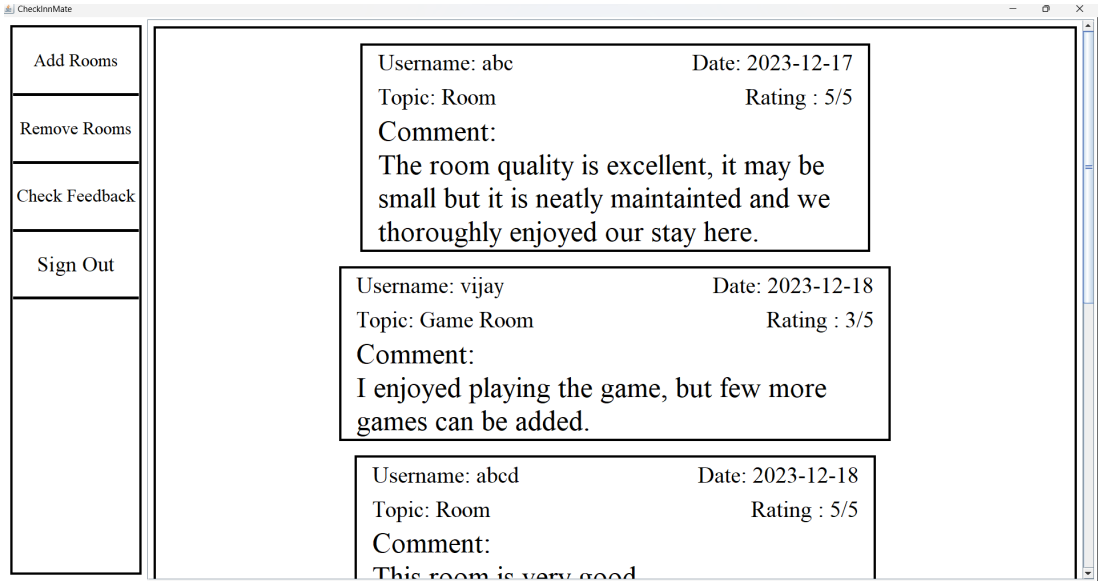
## Remove room

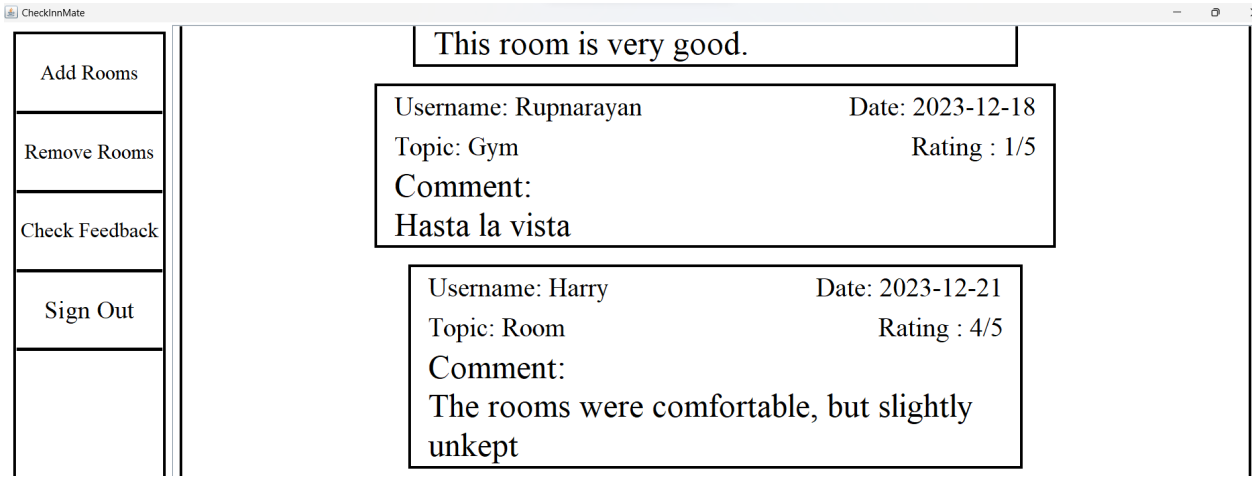
| | |
|---|---|
| Room No: 404 | Room No: 405 |
| Cost Per Night: ₹4000 | Cost Per Night: ₹3000 |
| REMOVE \| GET DETAILS | REMOVE \| GET DETAILS |

| | |
|---|---|
| Room: CabanaRoom | Room: CabanaRoom |
| Room No: 406 | Room No: 410 |
| Cost Per Night: ₹4000 | Cost Per Night: ₹4500 |
| REMOVE \| GET DETAILS | CONFIRM \| GET DETAILS |

| | |
|---|---|
| Room: ApartmentRoom | Room: ExecutiveRoom |
| Room No: 500 | Room No: 501 |

# Checking user feedbacks



# OBJECT ORIENTED FEATURES

**1. Abstraction and Encapsulation:**

Abstraction: Abstracting common features from various classes into a base class, such as the Room class, allows for a high-level representation of rooms with shared attributes and methods. It has also been used in Amenity class.

Encapsulation: Encapsulating data within classes like User, Admin, and Room ensures that the internal and personal details are hidden. For example, the specific details of a user's reservation history or an admin's privileges are encapsulated within their respective classes.

**2. Inheritance:**

The Room class serves as an abstract base class, and specific room types (StandardRoom, DeluxeRoom, Suite) inherits from it. This allows the derived classes to inherit common properties and behaviors while enabling customization for specific room types.
The Amenity class serves as abstract base class from which GameRoom, Gym and SwimmingPool inherit the utilize method. The front end of the application also involves inheritance, where various User Defined Components extends pre-existing components such as JPanel and JButton.

**3. Polymorphism:**

Polymorphism generally includes method overloading and method overriding. Method overloading has been made use of in various locations, such as in the MongoModifier class where the same getRooms method takes various different parameters, so as to retrieve a room based on different conditions. Similar methods have been implemented for users and guests as well. Constructor overloading is also used throughout the project. When it comes to overriding, methods from abstract class Room and abstract class Amenity are overridden in their respective subclasses. Various interfaces, both user defined (Ex. FeedbackGiver) as well as predefined (Ex. ActionListener) have been used throughout the program, with the overriding of their methods as well.

**4. Interface :**

Interfaces provide a way to achieve multiple inheritance, enforce a contract for implementing classes, promote code reusability, and facilitate design flexibility. FeedbackGiver serves as the interface guest class implements the Feedback system. RateBased serves as interface for methods like get and set rate, implemented by the amenity classes (gameroom, gym and

swimming pool). Other functional interfaces such as ActionListener and MouseListener are used all throughout the Front End of the program.

**5. Generic Types**:

Generic Types in DropDownMenu: The DropDownMenu class is made for the implementation selecting choices from a list of choices in the frontend part of the application. It actually uses the JComboBox generic class present in swing and tweeks it according to the display requirements. Essentially, the same DropDownMenu can be used for Strings, Integers, Doubles, etc. Beyond this, we have also made use of the Comparable<T> interface in our Room class. We have used this generic interface to define the compareTo function in such a way that Rooms are sorted in ascending order of their roomId. Collections such as ArrayList<T> and HashSet<T> are also used throughout the program to store Rooms, Feedback, etc.

Incorporating these concepts enhances the flexibility, maintainability, and scalability of the hotel management system by promoting modular design, code reuse, and the ability to extend and modify the system with minimal impact on existing functionality.

# INFERENCE

**1. Responsive User Interface with Swing:**
- Utilizes Swing for a responsive and user-friendly interface, ensuring a seamless experience for end-users.

**2. Ease of Usage for End Users:**
- Prioritizes end-user convenience by providing a straightforward booking process and direct access to amenities, enhancing overall user experience.

**3. MongoDB Database for Scalability:**
- Implements MongoDB database for easy scaling in the future, providing a robust foundation to accommodate the system's growth and increased data volume.

**4. Admin-Friendly Interface:**
- Develops a well-crafted administrative interface, enabling administrators to manage the system efficiently with intuitive controls and features.

**5. Portability and Scalability:**
- Designed to be easily portable and scalable, ensuring adaptability for future developments and implementations to meet evolving requirements.

**6. Automatic Bill Generation on Check-Out:**
- Enhances customer comfort with immediate automatic bill generation during check-out, streamlining the payment process and improving overall customer satisfaction.

**7. Technology Integration for Optimization:**
- Integrates technology to optimize user interactions, leveraging Swing for the frontend and MongoDB for the backend, ensuring a harmonious and efficient system.

**8. Future-Ready Development:**
- Anticipates future needs and developments, aligning with a scalable architecture to accommodate growth and technological advancements.

**9. Streamlined Administrative Tasks:**
- Streamlines administrative tasks by providing a well-developed interface, contributing to the overall efficiency and effectiveness of the system.

**10. Enhanced User and Administrator Experience:**
- Focuses on providing an enhanced experience for both users and administrators, creating a system that is intuitive, adaptable, and technologically robust.

# FUTURE EXPANSION

**Machine Learning for Demand Forecasting:**

Utilize Java libraries for machine learning, such as Apache Mahout or Deeplearning4j, to implement algorithms for demand forecasting.

**Chatbot for Customer Support:**

Implement a chatbot using Java-based natural language processing libraries like Apache OpenNLP or Stanford NLP.

**Automated Email Marketing:**

Use JavaMail API to integrate automated email marketing functionalities, sending personalized emails to past guests.

**Dynamic Pricing Algorithm:**

Implement dynamic pricing algorithms using Java to adjust room rates based on occupancy levels and other factors.

**Enhanced Analytics Dashboard:**

Develop a comprehensive analytics dashboard using Java-based frameworks like JavaFX or web frameworks like Spring Boot for web applications.

**Mobile Room Service Ordering:**

Create a mobile app feature using Java for Android development (using Android Studio) or Java-based frameworks like Codename One for cross-platform development.

**Biometric Security for Room Access:**

Integrate biometric authentication using Java libraries compatible with biometric devices, ensuring secure room access.

**Automated Billing and Invoicing:**

Implement automated billing and invoicing using Java for backend logic, with JavaFX or web frameworks for a user-friendly interface.

**Blockchain for Security and Transparency:**

Explore Java-based blockchain frameworks like Hyperledger Fabric for enhancing security and transparency in transactions.

**Augmented Reality (AR) for Virtual Tours:**

Develop an AR feature using Java and frameworks like ARCore for Android or ARToolkit for cross-platform AR development.