University Of Petroleum & Energy Studies, Dehradun

Object Oriented Programming Lab

Assignment : Java Application in a Domain of Choice

Submitted By - Aarohi Mathur - 500123455, B5

Brijesh Agarwal - 500121826, B5

Rashi Raj - 500122395 , B5

Shubh Natani- 500120201, B5

Submitted To - Deepak Kumar Sharma Sir

Student Housing Management System (Phase 1)

https://github.com/Brijeshagar/Student-Housing-management-system-

---

Team Members

1. Name - Shubh Natani
   SAP ID - 500120201

2. Name - Brijesh Agarwal
   SAP ID - 500121826

3. Name - Rashi Raj
   SAP ID - 500122395

4. Name - Aarohi Mathur
   SAP ID - 500123455

Phase 1 — Command-Line Application

1. Title

Student Housing Management System

---

2. System Overview

The **Student Housing Management System** is a Java command-line based application to help students find and reserve appropriate accommodations—i.e., PGs, hostels, and flats—within their vicinity of study. The system provides an intuitive interface with easy housing search capabilities through filtering and reservation features.

We chose this project as, being students ourselves, we experienced how overwhelming it can be to find suitable accommodation amidst countless options and limited time—especially during year-end.

3. Key Features / Functionalities

1. Accommodation Management

- Load housing data from CSV file
- Differentiate properties by accommodation type (PG, Hostel, Flat)
- Detailed view of property information

2. User System

- Account registration and login
- User session handling

3. Search & Filtering

- Filter options by location, rent, facilities
- Sort on rating, policies by gender

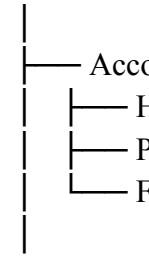
4. Booking System

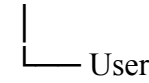
- Simplified booking system
- Store user contact information and retrieve them

---

4. Class Structure

Class Diagram Overview

HousingProperty (Abstract)

Accommodation (Base class)
Hostel (Specialized)
PG (Specialized)
Flat (Specialized)

Authentication (Interface)

UserAuthenticator (Implements Authentication)

5. Complete Code :

```
import java.io.*;
import java.util.Scanner;

public class test{
   public static void main(String[] args) {
      HousingManager manager = new HousingManager();
```

```
        manager.loadAccommodations("oops_dataset2.csv");
        manager.startSystem();
    }
}

abstract class HousingProperty {
    private final String name;

    public HousingProperty(String name) {
        this.name = name;
    }

    public final String getName() {
        return name;
    }

    public abstract void displayBasicInformation();
}

class Accommodation extends HousingProperty {
    private final String type;
    private final String location;
    private final int rent;
    private final boolean wifi;
    private final boolean meals;
    private final boolean security;
    private final double rating;
    private final int roomSharing;
    private final boolean laundry;
    private final boolean mess;
    private final boolean ac;
    private final boolean geyser;
    private final boolean kitchen;
    private final String curfew;
    private final String gender;

    public Accommodation(String name, String type, String location, int rent, boolean wifi,
                         boolean meals, boolean security, double rating, int roomSharing,
                         boolean laundry, boolean mess, boolean ac, boolean geyser,
                         boolean kitchen, String curfew, String gender) {
```

```
    super(name);
    this.type = type;
    this.location = location;
    this.rent = rent;
    this.wifi = wifi;
    this.meals = meals;
    this.security = security;
    this.rating = rating;
    this.roomSharing = roomSharing;
    this.laundry = laundry;
    this.mess = mess;
    this.ac = ac;
    this.geyser = geyser;
    this.kitchen = kitchen;
    this.curfew = curfew;
    this.gender = gender;
}

public String getType() {
return type;
}
public String getLocation() {
return location;
}
public int getRent() {
return rent;
}
public boolean hasWifi() {
return wifi;
}
public boolean hasMeals() {
return meals;
}
public boolean hasSecurity() {
return security;
}
public double getRating() {
return rating;
}
public int getRoomSharing() {
```

```
return roomSharing;
}
public boolean hasLaundry() {
return laundry;
}
public boolean hasMess() {
return mess;
}
public boolean hasAC() {
return ac;
}
public boolean hasGeyser() {
return geyser;
}
public boolean hasKitchen() {
return kitchen;
}
public String getCurfew() {
return curfew;
}
public String getGender() {
return gender;
}

public void displayBasicInformation() {
    System.out.println("\n--- " + type + " Accommodation ---");
    System.out.println("Name: " + super.getName());
    System.out.println("Location: " + location);
    System.out.println("Rent: Rs" + rent);
    System.out.println("Rating: " + rating + "/5");
    System.out.println("Gender: " + gender);
    if (!type.equalsIgnoreCase("Flat")) {
        System.out.println("Room Sharing: " + roomSharing + " sharing");
        System.out.println("Curfew: " + (curfew.equals("None") ? "No curfew" : curfew));
    } else {
        System.out.println("Private apartment");
    }
}

public void displayDetails() {
```

```
        displayBasicInformation();
        System.out.println("\nFacilities:");
        System.out.println("- WiFi: " + (wifi ? "Yes" : "No"));
        System.out.println("- Meals: " + (meals ? "Yes" : "No"));
        System.out.println("- Security: " + (security ? "Yes" : "No"));
        System.out.println("- Laundry: " + (laundry ? "Yes" : "No"));
        System.out.println("- Mess: " + (mess ? "Yes" : "No"));
        System.out.println("- AC: " + (ac ? "Yes" : "No"));
        System.out.println("- Geyser: " + (geyser ? "Yes" : "No"));
        System.out.println("- Kitchen: " + (kitchen ? "Yes" : "No"));
    }
}

class Hostel extends Accommodation {
    private final boolean studyRoom;
    private final boolean sportsFacility;

    public Hostel(String name, String location, int rent, boolean wifi,
                  boolean meals, boolean security, double rating, int roomSharing,
                  boolean laundry, boolean mess, boolean ac, boolean geyser,
                  boolean kitchen, String curfew, String gender,
                  boolean studyRoom, boolean sportsFacility) {
        super(name, "Hostel", location, rent, wifi, meals, security, rating,
              roomSharing, laundry, mess, ac, geyser, kitchen, curfew, gender);
        this.studyRoom = studyRoom;
        this.sportsFacility = sportsFacility;
    }

    public void displayDetails() {
        super.displayDetails();
        System.out.println("Hostel-specific Facilities:");
        System.out.println("- Study Room: " + (studyRoom ? "Yes" : "No"));
        System.out.println("- Sports Facility: " + (sportsFacility ? "Yes" : "No"));
    }
}

class PG extends Accommodation {
    private final boolean parking;
    private final boolean powerBackup;
```

```
    public PG(String name, String location, int rent, boolean wifi,
            boolean meals, boolean security, double rating, int roomSharing,
            boolean laundry, boolean mess, boolean ac, boolean geyser,
            boolean kitchen, String curfew, String gender,
            boolean parking, boolean powerBackup) {
        super(name, "PG", location, rent, wifi, meals, security, rating,
             roomSharing, laundry, mess, ac, geyser, kitchen, curfew, gender);
        this.parking = parking;
        this.powerBackup = powerBackup;
    }

    public void displayDetails(){
        super.displayDetails();
        System.out.println("PG-specific Facilities:");
        System.out.println("- Parking: " + (parking ? "Yes" : "No"));
        System.out.println("- Power Backup: " + (powerBackup ? "Yes" : "No"));
    }
}

class Flat extends Accommodation{
    private final boolean furnished;
    private final boolean maintenanceIncluded;

    public Flat(String name, String location, int rent, boolean wifi,
             boolean meals, boolean security, double rating,
             boolean laundry, boolean mess, boolean ac, boolean geyser,
             boolean kitchen, String gender,
             boolean furnished, boolean maintenanceIncluded) {
        super(name, "Flat", location, rent, wifi, meals, security, rating,
            1, laundry, mess, ac, geyser, kitchen, "None", gender);
        this.furnished = furnished;
        this.maintenanceIncluded = maintenanceIncluded;
    }

    public void displayDetails(){
        super.displayDetails();
        System.out.println("Flat-specific Facilities:");
        System.out.println("- Furnished: " + (furnished ? "Yes" : "No"));
        System.out.println("- Maintenance Included: " + (maintenanceIncluded ? "Yes" : "No"));
    }
```

```
}

interface Authentication{
    boolean login(String username, String password);
    void register(String username, String password);
}

class UserAuthenticator implements Authentication{
    private String username;
    private String password;
    private static int registrationCount = 0;

    public boolean login(String username, String password){
        return this.username != null && this.username.equals(username) &&
                this.password != null && this.password.equals(password);
    }

    public void register(String username, String password){
        this.username = username;
        this.password = password;
        registrationCount++;
        System.out.println("Registration successful! You can now login with your credentials.");
    }

    public void register(String username, String password, boolean displayCount) {
        register(username, password);
    }
}

class HousingManager{
    private Accommodation[] accommodations;
    private int accommodationCount;
    private final Scanner scanner;
    private final UserAuthenticator authenticator;
    private String currentUser;

    public HousingManager() {
        accommodations = new Accommodation[100];
        accommodationCount = 0;
        scanner = new Scanner(System.in);
```

```
        authenticator = new UserAuthenticator();
        currentUser = null;
    }

    public void loadAccommodations(String filename) {
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;
            boolean firstLine = true;

            while ((line = br.readLine()) != null && accommodationCount <
accommodations.length) {
                if (firstLine) {
                    firstLine = false;
                    continue;
                }

                String[] values = line.split(",");
                if (values.length >= 22) {
                    String type = values[1].trim();
                    Accommodation acc;

                    if (type.equalsIgnoreCase("Hostel")) {
                        acc = new Hostel(
                            values[0].trim(), values[2].trim(),
                            Integer.parseInt(values[3].trim()),
                            values[4].trim().equals("Yes"),
                            values[5].trim().equals("Yes"),
                            values[6].trim().equals("Yes"),
                            Double.parseDouble(values[7].trim()),
                            Integer.parseInt(values[8].trim()),
                            values[9].trim().equals("Yes"),
                            values[10].trim().equals("Yes"),
                            values[11].trim().equals("Yes"),
                            values[12].trim().equals("Yes"),
                            values[13].trim().equals("Yes"),
                            values[14].trim(), values[15].trim(),
                            values[17].trim().equals("Yes"),
                            values[18].trim().equals("Yes")
                        );
                    } else if (type.equalsIgnoreCase("PG")){
```

```
            acc = new PG(
                values[0].trim(), values[2].trim(),
                Integer.parseInt(values[3].trim()),
                values[4].trim().equals("Yes"),
                values[5].trim().equals("Yes"),
                values[6].trim().equals("Yes"),
                Double.parseDouble(values[7].trim()),
                Integer.parseInt(values[8].trim()),
                values[9].trim().equals("Yes"),
                values[10].trim().equals("Yes"),
                values[11].trim().equals("Yes"),
                values[12].trim().equals("Yes"),
                values[13].trim().equals("Yes"),
                values[14].trim(), values[15].trim(),
                values[16].trim().equals("Yes"),
                values[17].trim().equals("Yes")
            );
        } else if (type.equalsIgnoreCase("Flat")) {
            acc = new Flat(
                values[0].trim(), values[2].trim(),
                Integer.parseInt(values[3].trim()),
                values[4].trim().equals("Yes"),
                values[5].trim().equals("Yes"),
                values[6].trim().equals("Yes"),
                Double.parseDouble(values[7].trim()),
                values[9].trim().equals("Yes"),
                values[10].trim().equals("Yes"),
                values[11].trim().equals("Yes"),
                values[12].trim().equals("Yes"),
                values[13].trim().equals("Yes"),
                values[15].trim(),
                values[20].trim().equals("Yes"),
                values[21].trim().equals("Yes")
            );
        } else {
            acc = new Accommodation(
                values[0].trim(), type, values[2].trim(),
                Integer.parseInt(values[3].trim()),
                values[4].trim().equals("Yes"),
                values[5].trim().equals("Yes"),
```

```
                    values[6].trim().equals("Yes"),
                    Double.parseDouble(values[7].trim()),
                    Integer.parseInt(values[8].trim()),
                    values[9].trim().equals("Yes"),
                    values[10].trim().equals("Yes"),
                    values[11].trim().equals("Yes"),
                    values[12].trim().equals("Yes"),
                    values[13].trim().equals("Yes"),
                    values[14].trim(), values[15].trim()
                );
            }

            accommodations[accommodationCount++] = acc;
        }
    }
    System.out.println("Loaded " + accommodationCount + " accommodations from file.");
  } catch (IOException e) {
    System.out.println("Error loading accommodations: " + e.getMessage());
  } catch (NumberFormatException e) {
    System.out.println("Error parsing data: " + e.getMessage());
  }
}

public void startSystem(){
  System.out.println("\n=== Welcome to Student Housing Management System ===");
  System.out.println("1. Login");
  System.out.println("2. Register");
  System.out.println("3. Exit");
  System.out.print("Enter your choice: ");

  int choice = scanner.nextInt();
  scanner.nextLine();

  switch (choice) {
    case 1:
      handleLogin();
      break;
    case 2:
      handleRegistration();
      break;
```

```
        case 3:
            System.out.println("Thank you for using the system. Goodbye!");
            System.exit(0);
            break;
        default:
            System.out.println("Invalid choice. Please try again.");
            startSystem();
    }
}

private void handleLogin() {
    System.out.print("\nEnter username: ");
    String username = scanner.nextLine();
    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    if (authenticator.login(username, password)) {
        currentUser = username;
        System.out.println("\nLogin successful! Welcome, " + username + "!");
        showMainMenu();
    } else {
        System.out.println("Invalid username or password. Please try again.");
        startSystem();
    }
}

private void handleRegistration() {
    System.out.print("\nEnter a username: ");
    String username = scanner.nextLine();
    System.out.print("Enter a password: ");
    String password = scanner.nextLine();

    authenticator.register(username, password, true);
    startSystem();
}

private void showMainMenu() {
    System.out.println("\n=== Main Menu ===");
    System.out.println("1. Search Accommodations");
    System.out.println("2. View All Accommodations");
```

```
        System.out.println("3. Filter Accommodations");
        System.out.println("4. Logout");
        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                searchAccommodations();
                break;
            case 2:
                viewAllAccommodations();
                break;
            case 3:
                filterAccommodations();
                break;
            case 4:
                currentUser = null;
                System.out.println("Logged out successfully.");
                startSystem();
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
                showMainMenu();
        }
    }

    private void searchAccommodations() {
        System.out.println("\n=== Search Accommodations ===");
        System.out.println("1. PG (Paying Guest)");
        System.out.println("2. Hostel");
        System.out.println("3. Flat");
        System.out.println("4. View All Types");
        System.out.println("5. Back to Main Menu");
        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();
        scanner.nextLine();
```

```
String type = "";
switch (choice) {
    case 1:
        type = "PG";
        break;
    case 2:
        type = "Hostel";
        break;
    case 3:
        type = "Flat";
        break;
    case 4:
        viewAllAccommodations();
        return;
    case 5:
        showMainMenu();
        return;
    default:
        System.out.println("Invalid choice. Please try again.");
        searchAccommodations();
        return;
}

System.out.println("\n=== Available " + type + " Accommodations ===");
boolean found = false;
for (int i = 0; i < accommodationCount; i++) {
    if (accommodations[i].getType().equalsIgnoreCase(type)) {
        System.out.println((i+1) + ". " + accommodations[i].getName() +
                           " - " + accommodations[i].getLocation() +
                           " (Rs" + accommodations[i].getRent() + ")");
        found = true;
    }
}

if (!found) {
    System.out.println("No " + type + " accommodations found.");
} else {
    viewAccommodationDetails();
}
```

```
    showMainMenu();
}

private void viewAllAccommodations() {
    System.out.println("\n=== All Available Accommodations ===");
    for (int i = 0; i < accommodationCount; i++) {
        System.out.println((i+1) + ". " + accommodations[i].getName() +
                           " (" + accommodations[i].getType() + ") - " +
                           accommodations[i].getLocation() +
                           " (Rs" + accommodations[i].getRent() + ")");
    }

    viewAccommodationDetails();
    showMainMenu();
}

private void viewAccommodationDetails(){
    System.out.print("\nEnter the number to view details (0 to go back): ");
    int choice = scanner.nextInt();
    scanner.nextLine();

    if (choice == 0) {
        return;
    } else if (choice > 0 && choice <= accommodationCount) {
        accommodations[choice-1].displayDetails();
        System.out.print("\nWould you like to book this accommodation? (yes/no): ");
        String bookChoice = scanner.nextLine();

        if (bookChoice.equalsIgnoreCase("yes")) {
            bookAccommodation(choice-1);
        }
    } else {
        System.out.println("Invalid choice. Please try again.");
        viewAccommodationDetails();
    }
}

private void bookAccommodation(int index){
    System.out.println("\n=== Booking Process ===");
    System.out.println("You are booking: " + accommodations[index].getName());
```

```
        System.out.println("Rent: Rs" + accommodations[index].getRent());

        System.out.print("Enter your full name: ");
        String name = scanner.nextLine();
        System.out.print("Enter your contact number: ");
        String contact = scanner.nextLine();
        System.out.print("Enter your university: ");
        String university = scanner.nextLine();

        System.out.println("\nBooking confirmed for " + name + "!");
        System.out.println("Contact: " + contact);
        System.out.println("Accommodation: " + accommodations[index].getName());
        System.out.println("Please contact the owner to complete the process.");

        System.out.println("\nThank you for using our service!");
    }

    private void filterAccommodations() {
        System.out.println("\n=== Filter Accommodations ===");
        System.out.println("Filter by:");
        System.out.println("1. Location");
        System.out.println("2. Rent Range");
        System.out.println("3. Facilities");
        System.out.println("4. Rating");
        System.out.println("5. Gender Preference");
        System.out.println("6. Back to Main Menu");
        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                filterByLocation();
                break;
            case 2:
                filterByRent();
                break;
            case 3:
                filterByFacilities();
```

```
            break;
        case 4:
            filterByRating();
            break;
        case 5:
            filterByGender();
            break;
        case 6:
            showMainMenu();
            break;
        default:
            System.out.println("Invalid choice. Please try again.");
            filterAccommodations();
    }
}

private void filterByLocation(){
    System.out.println("\nAvailable Locations:");
    System.out.println("- Prem Nagar");
    System.out.println("- Bhidoli");
    System.out.println("- Kandoli");
    System.out.print("Enter location to filter: ");
    String location = scanner.nextLine();

    System.out.println("\n=== Accommodations in " + location + " ===");
    boolean found = false;
    for (int i = 0; i < accommodationCount; i++) {
        if (accommodations[i].getLocation().equalsIgnoreCase(location)) {
            System.out.println((i+1) + ". " + accommodations[i].getName() +
                               " (" + accommodations[i].getType() + ") - " +
                               "Rs" + accommodations[i].getRent());
            found = true;
        }
    }

    if (!found) {
        System.out.println("No accommodations found in " + location);
    } else {
        viewAccommodationDetails();
    }
```

```
        showMainMenu();
    }

    private void filterByRent() {
        System.out.print("\nEnter minimum rent: ");
        int min = scanner.nextInt();
        System.out.print("Enter maximum rent: ");
        int max = scanner.nextInt();
        scanner.nextLine();

        System.out.println("\n=== Accommodations between Rs" + min + " and Rs" + max + "
===");
        boolean found = false;
        for (int i = 0; i < accommodationCount; i++) {
            if (accommodations[i].getRent() >= min && accommodations[i].getRent() <= max) {
                System.out.println((i+1) + ". " + accommodations[i].getName() +
                                   " (" + accommodations[i].getType() + ") - " +
                                   accommodations[i].getLocation() +
                                   " (Rs" + accommodations[i].getRent() + ")");
                found = true;
            }
        }

        if (!found) {
            System.out.println("No accommodations found in this rent range.");
        } else {
            viewAccommodationDetails();
        }

        showMainMenu();
    }

    private void filterByFacilities(){
        System.out.println("\nSelect facilities to filter by:");
        System.out.println("1. WiFi");
        System.out.println("2. Meals");
        System.out.println("3. Security");
        System.out.println("4. Laundry");
        System.out.println("5. AC");
```

```
System.out.println("6. Geyser");
System.out.println("7. Kitchen");
System.out.print("Enter facility numbers (comma separated, e.g., 1,3,5): ");

String[] choices = scanner.nextLine().split(",");
boolean[] selected = new boolean[7];

for (String choice : choices) {
   try {
      int num = Integer.parseInt(choice.trim());
      if (num >= 1 && num <= 7) {
         selected[num-1] = true;
      }
   } catch (NumberFormatException e){
      // Ignore invalid entries
   }
}

System.out.println("\n=== Accommodations with selected facilities ===");
boolean found = false;

for (int i = 0; i < accommodationCount; i++) {
   boolean matches = true;

   if (selected[0] && !accommodations[i].hasWifi()) matches = false;
   if (selected[1] && !accommodations[i].hasMeals()) matches = false;
   if (selected[2] && !accommodations[i].hasSecurity()) matches = false;
   if (selected[3] && !accommodations[i].hasLaundry()) matches = false;
   if (selected[4] && !accommodations[i].hasAC()) matches = false;
   if (selected[5] && !accommodations[i].hasGeyser()) matches = false;
   if (selected[6] && !accommodations[i].hasKitchen()) matches = false;

   if (matches) {
      System.out.println((i+1) + ". " + accommodations[i].getName() +
                  " (" + accommodations[i].getType() + ") - " +
                  accommodations[i].getLocation() +
                  " (Rs" + accommodations[i].getRent() + ")");
      found = true;
   }
}
```

```
    if (!found) {
        System.out.println("No accommodations found with all selected facilities.");
    } else {
        viewAccommodationDetails();
    }

    showMainMenu();
}

private void filterByRating(){
    System.out.print("\nEnter minimum rating (1-5): ");
    double minRating = scanner.nextDouble();
    scanner.nextLine();

    System.out.println("\n=== Accommodations with rating " + minRating + "+ ===");
    boolean found = false;
    for (int i = 0; i < accommodationCount; i++) {
        if (accommodations[i].getRating() >= minRating) {
            System.out.println((i+1) + ". " + accommodations[i].getName() +
                    " (" + accommodations[i].getType() + ") - " +
                    accommodations[i].getLocation() +
                    " (Rating: " + accommodations[i].getRating() + ")");
            found = true;
        }
    }

    if (!found){
        System.out.println("No accommodations found with this minimum rating.");
    } else {
        viewAccommodationDetails();
    }

    showMainMenu();
}

private void filterByGender() {
    System.out.println("\nSelect gender preference:");
    System.out.println("1. Male");
    System.out.println("2. Female");
```

```
System.out.println("3. Unisex");
System.out.print("Enter your choice: ");

int choice = scanner.nextInt();
scanner.nextLine();

String gender = "";
switch (choice) {
    case 1:
        gender = "Male";
        break;
    case 2:
        gender = "Female";
        break;
    case 3:
        gender = "Unisex";
        break;
    default:
        System.out.println("Invalid choice. Please try again.");
        filterByGender();
        return;
}

System.out.println("\n=== " + gender + " Accommodations ===");
boolean found = false;
for (int i = 0; i < accommodationCount; i++) {
    if (accommodations[i].getGender().equalsIgnoreCase(gender)) {
        System.out.println((i+1) + ". " + accommodations[i].getName() +
                           " (" + accommodations[i].getType() + ") - " +
                           accommodations[i].getLocation() +
                           " (Rs" + accommodations[i].getRent() + ")");
        found = true;
    }
}

if (!found) {
    System.out.println("No " + gender + " accommodations found.");
} else {
    viewAccommodationDetails();
}
```

```
        showMainMenu();
    }
}
```

---

6. Functionalities Offered by the Code

1. User Registration and Login

User enters:
Name and password
Data is stored in a list of `User` objects.

Figure 1: User Registration

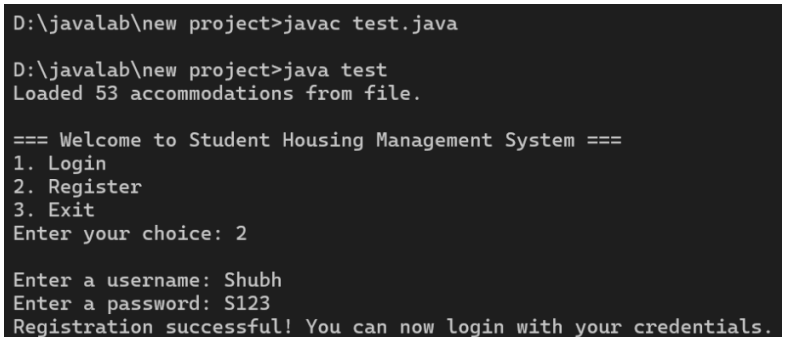
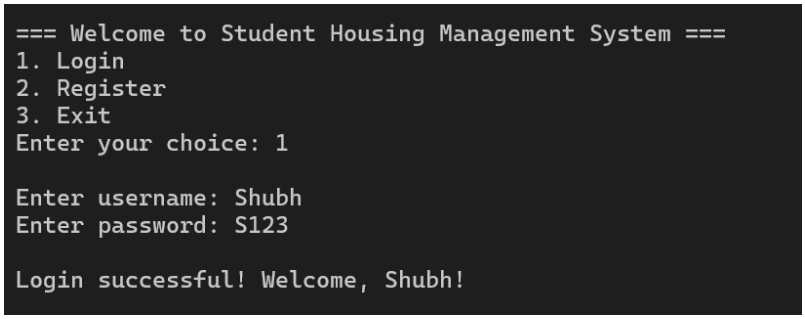


Figure 2: User Login



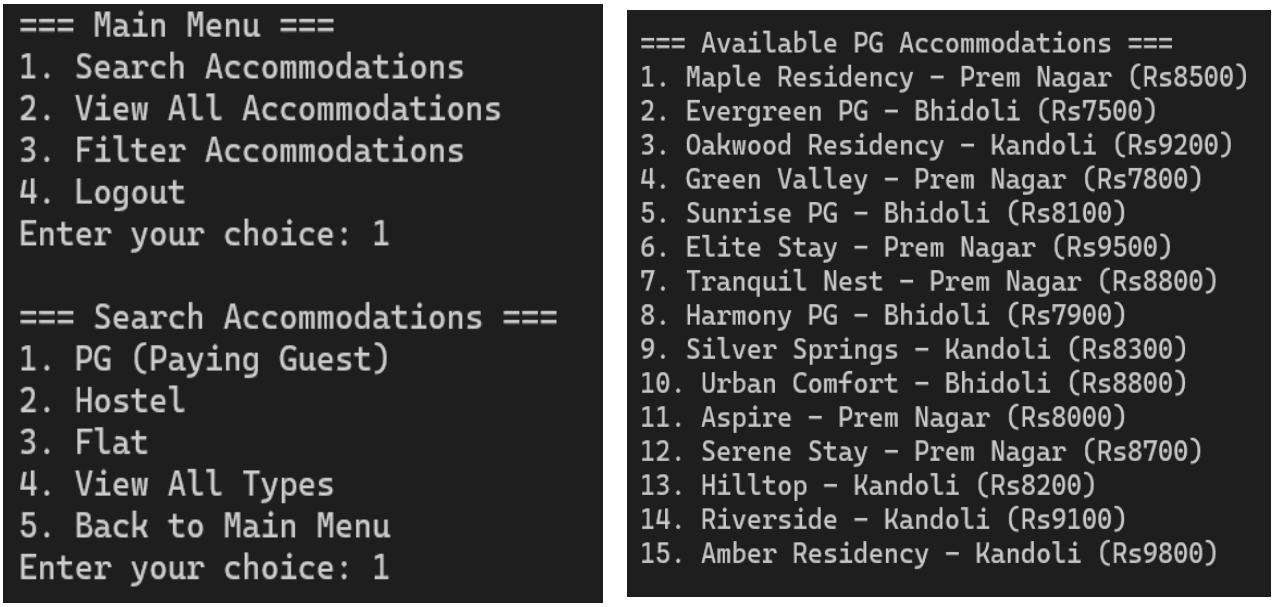
2. View All Available Accommodations- Displays a list of all accommodations with:

- Serial Number
- Name
- Type (e.g., PG, Flat, Hostel)
- Location
- Rent

Figure 3: View all Accommodations

3. View Specific Type Accommodations- Displays a list of Specific type of accommodations

Figure 4: View a specific type of accommodation eg 'PG'



4. Filter Accommodations- You can apply filters based on:

A. Location - **Figure 5** Filter Accommodation By Location

```
=== Main Menu ===
1. Search Accommodations
2. View All Accommodations
3. Filter Accommodations
4. Logout
Enter your choice: 3

=== Filter Accommodations ===
Filter by:
1. Location
2. Rent Range
3. Facilities
4. Rating
5. Gender Preference
6. Back to Main Menu
Enter your choice: 1

Available Locations:
- Prem Nagar
- Bhidoli
- Kandoli
Enter location to filter: bhidoli
```

```
=== Accommodations in bhidoli ===
2. Evergreen PG (PG) - Rs7500
5. Sunrise PG (PG) - Rs8100
8. Harmony PG (PG) - Rs7900
10. Urban Comfort (PG) - Rs8800
19. Riverside (Hostel) - Rs7200
21. Maple Residency (Hostel) - Rs7100
23. Mountain Breeze (Hostel) - Rs6900
24. Elite Boys (Hostel) - Rs7300
26. Silver Crest (Hostel) - Rs7700
27. Urban Haven (Hostel) - Rs6700
30. Hilltop Boys (Hostel) - Rs7200
32. Amber (Hostel) - Rs7100
35. Riverside (Flat) - Rs11500
36. Green Valley (Flat) - Rs12500
41. Maple Heights (Flat) - Rs14500
43. Urban Nest (Flat) - Rs10500
48. Aspire (Flat) - Rs11500
53. Royal Nest (Flat) - Rs11000
```

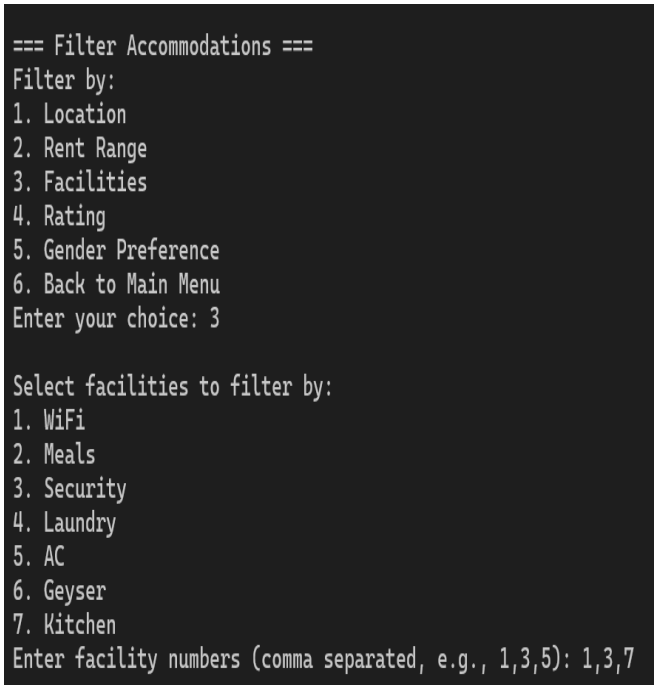
B. Rent Range - **Figure 6** Filter Accommodation by Rent Range

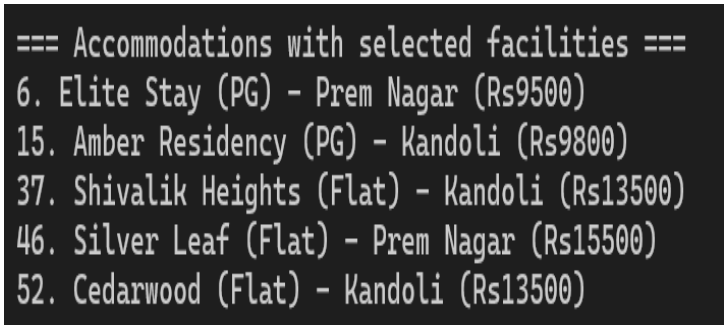
```
=== Filter Accommodations ===
Filter by:
1. Location
2. Rent Range
3. Facilities
4. Rating
5. Gender Preference
6. Back to Main Menu
Enter your choice: 2

Enter minimum rent: 4000
Enter maximum rent: 8000
```

```
=== Accommodations between Rs4000 and Rs8000 ===
2. Evergreen PG (PG) - Bhidoli (Rs7500)
4. Green Valley (PG) - Prem Nagar (Rs7800)
8. Harmony PG (PG) - Bhidoli (Rs7900)
11. Aspire (PG) - Prem Nagar (Rs8000)
16. Himalayan View (Hostel) - Kandoli (Rs6000)
17. Green Valley (Hostel) - Prem Nagar (Rs7000)
18. Shivalik Boys (Hostel) - Kandoli (Rs6500)
19. Riverside (Hostel) - Bhidoli (Rs7200)
20. Golden Nest (Hostel) - Kandoli (Rs6800)
21. Maple Residency (Hostel) - Bhidoli (Rs7100)
22. Oakwood Boys (Hostel) - Prem Nagar (Rs7400)
23. Mountain Breeze (Hostel) - Bhidoli (Rs6900)
24. Elite Boys (Hostel) - Bhidoli (Rs7300)
25. Tranquil Stay (Hostel) - Prem Nagar (Rs7600)
26. Silver Crest (Hostel) - Bhidoli (Rs7700)
27. Urban Haven (Hostel) - Bhidoli (Rs6700)
28. Aspire (Hostel) - Kandoli (Rs7500)
29. Serene Girls (Hostel) - Kandoli (Rs7900)
30. Hilltop Boys (Hostel) - Bhidoli (Rs7200)
31. Cedarwood Girls (Hostel) - Prem Nagar (Rs6800)
32. Amber (Hostel) - Bhidoli (Rs7100)
33. Blue Haven (Hostel) - Kandoli (Rs7400)
```

C. Facilities - User selects required facilities like (WiFi, Meals, AC etc) using numbers

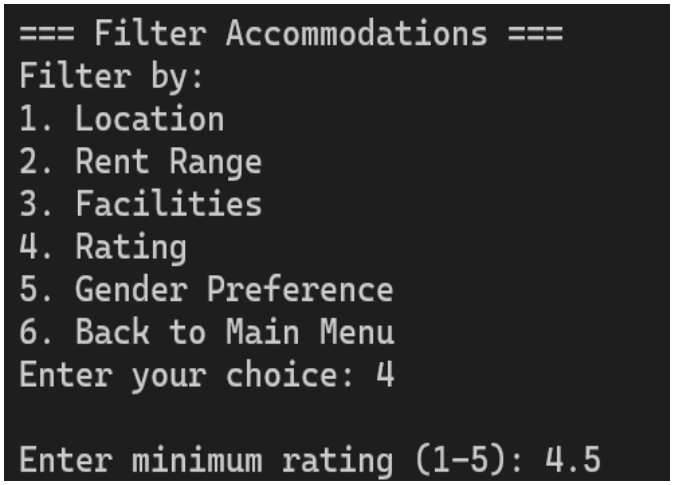
Figure 7 Filter Accommodation based on facilities

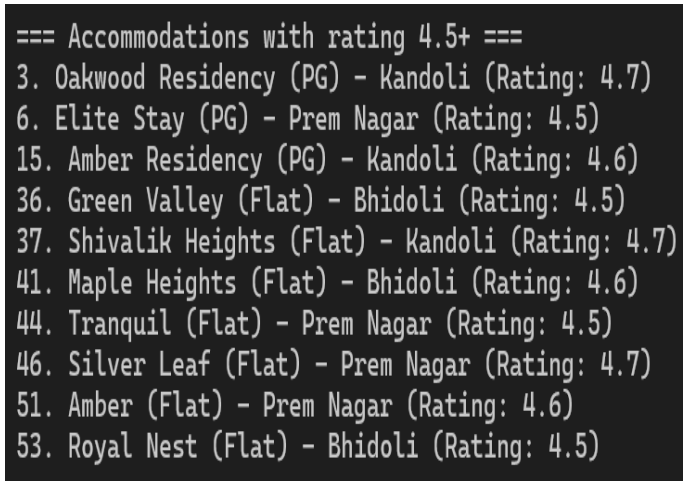




D. Rating -

Figure 8 Filter Accommodation based on rating





E. Gender Preference - Filter based on gender preference

-male
-female
-unisex

```
=== Filter Accommodations ===
Filter by:
1. Location
2. Rent Range
3. Facilities
4. Rating
5. Gender Preference
6. Back to Main Menu
Enter your choice: 5

Select gender preference:
1. Male
2. Female
3. Unisex
Enter your choice: 3
```

```
=== Unisex Accommodations ===
1. Maple Residency (PG) - Prem Nagar (Rs8500)
4. Green Valley (PG) - Prem Nagar (Rs7800)
7. Tranquil Nest (PG) - Prem Nagar (Rs8800)
9. Silver Springs (PG) - Kandoli (Rs8300)
11. Aspire (PG) - Prem Nagar (Rs8000)
14. Riverside (PG) - Kandoli (Rs9100)
19. Riverside (Hostel) - Bhidoli (Rs7200)
26. Silver Crest (Hostel) - Bhidoli (Rs7700)
32. Amber (Hostel) - Bhidoli (Rs7100)
34. Himalayan View (Flat) - Kandoli (Rs12000)
37. Shivalik Heights (Flat) - Kandoli (Rs13500)
40. Evergreen (Flat) - Prem Nagar (Rs11000)
43. Urban Nest (Flat) - Bhidoli (Rs10500)
46. Silver Leaf (Flat) - Prem Nagar (Rs15500)
50. Hilltop (Flat) - Prem Nagar (Rs13000)
53. Royal Nest (Flat) - Bhidoli (Rs11000)
```

Figure 9: Viewing Accommodation Details based on gender

5. Viewing Details of Accomodation

Figure 10: Viewing Accommodation Details

```
Enter the number to view details (0 to go back): 45

--- Flat Accommodation ---
Name: Golden Crest
Location: Prem Nagar
Rent: Rs12500
Rating: 3.8/5
Gender: Male
Private apartment
```

```
Facilities:
- WiFi: No
- Meals: No
- Security: Yes
- Laundry: No
- Mess: No
- AC: No
- Geyser: Yes
- Kitchen: Yes
Flat-specific Facilities:
- Furnished: No
- Maintenance Included: Yes
```

6. Booking an Accommodation

- Takes:
    - Full Name
    - Contact Number
    - University Name

Figure 11: Booking Confirmation

```
Would you like to book this accommodation? (yes/no): yes

=== Booking Process ===
You are booking: Golden Crest
Rent: Rs12500
Enter your full name: Shubh Natani
Enter your contact number: 8302386406
Enter your university: UPES

Booking confirmed for Shubh Natani!
Contact: 8302386406
Accommodation: Golden Crest
Please contact the owner to complete the process.

Thank you for using our service!
```

---

6. OOP Concepts Applied

- **Inheritance**: Hostel, PG, and Flat inherit from Accommodation
- **Abstraction**: HousingProperty serves as an abstract parent
- **Polymorphism**: Method overriding for displayDetails()
- **Encapsulation**: Use of private fields and accessors

---

7. Future Enhancements

- **Dynamic Data Storage**: Replacing fixed-size arrays with HashMap and ArrayList for flexible and efficient data handling.
- **SQL Integration**: Use MySQL/SQLite for storage, avoiding duplicate entries and improving performance, scalability, and reliability.

Phase 2 Documentation

1. Title

Student Housing Management System with GUI and Database Integration

---

2. Overview
In Phase 2, we changed our earlier command-line system into a full GUI application using Java Swing, and integrated it with an SQL database. This makes the system much easier to use and manage, with smoother booking, better data handling, and a separate interface for students and admins.

---

3. New Improvements

-> Graphical Interface

- Designed using Java Swing
- Separate views for students and admins
- Easy-to-use forms and menus

-> Admin Features

- Add, update, or delete accommodations
- View and confirm student bookings
- Secure admin login

-> Booking System

- Students can request bookings
- Admins can approve or reject requests
- Bookings shown with status updates

-> Smarter Data Handling

- HashMap for fast location searches
- ArrayList to store properties
- Grouping by type (hostel, PG, flat)

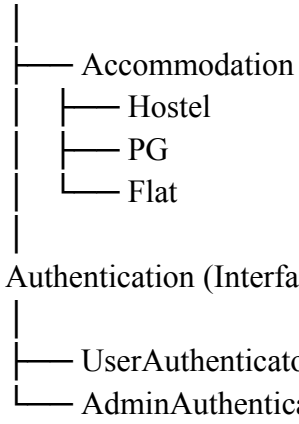
-> Better User Experience

- Form validation to prevent errors

- Pop-up messages for user actions
- Clean and responsive design

---

4. Class Structure

HousingProperty (Abstract)


Authentication (Interface)

Booking (Handles booking data)
HousingManager (Now supports GUI)

---

5. Main Features

-> Admin Panel

- Add/Edit/Delete accommodation details
- View all booking requests
- Confirm or reject bookings

-> Search & Filter

- Search by location
- Filter by rent (slider)
- Choose facilities and rating

-> Booking System

- Students submit booking forms
- Bookings shown as pending or confirmed
- Admin handles requests

7. Code Structure :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.List;
import java.util.Map;
import java.util.ArrayList;
import java.util.HashMap;
import java.sql.*;

public class StudentHousingSystemSwing1 {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            HousingManager manager = new HousingManager();
            manager.loadAccommodationsFromDatabase();
            manager.showLoginFrame();
        });
    }
}

abstract class HousingProperty {
    private final String name;
    public HousingProperty(String name) {
        this.name = name;
    }
    public final String getName() {
        return name;
    }
    public abstract void displayBasicInformation();
}

class Accommodation extends HousingProperty {
    private final String type;
    private final String location;
    private final Integer rent;
    private final Boolean wifi;
    private final Boolean meals;
    private final Boolean security;
     private final Double rating;
    private final Integer roomSharing;
    private final Boolean laundry;
    private final Boolean mess;
    private final Boolean ac;
    private final Boolean geyser;
    private final Boolean kitchen;
    private final String curfew;
    private final String gender;


     public Accommodation(String name, String type, String location, Integer rent, Boolean wifi,
                          Boolean meals, Boolean security, Double rating, Integer roomSharing,
                          Boolean laundry, Boolean mess, Boolean ac, Boolean geyser,
                          Boolean kitchen, String curfew, String gender) {
        super(name);
```

```
    this.type = type;
    this.location = location;
    this.rent = rent;
    this.wifi = wifi;
    this.meals = meals;
    this.security = security;
    this.rating = rating;
    this.roomSharing = roomSharing;
    this.laundry = laundry;
    this.mess = mess;
    this.ac = ac;
    this.geyser = geyser;
    this.kitchen = kitchen;
    this.curfew = curfew;
    this.gender = gender;
}

public String getType() {
return type;
}
public String getLocation() {
return location;
}
public Integer getRent() {
return rent;
}
public Boolean hasWifi() {
return wifi;
}
public Boolean hasMeals() {
return meals;
}
public Boolean hasSecurity() { return security; }
public Double getRating() { return rating; }
public Integer getRoomSharing() { return roomSharing; }
public Boolean hasLaundry() { return laundry; }
public Boolean hasMess() { return mess; }
public Boolean hasAC() { return ac; }
public Boolean hasGeyser() { return geyser; }
public Boolean hasKitchen() { return kitchen; }
public String getCurfew() { return curfew; }
public String getGender() { return gender; }

public void displayBasicInformation() {
    System.out.println("\n--- " + type + " Accommodation ---");
    System.out.println("Name: " + super.getName());
    System.out.println("Location: " + location);
    System.out.println("Rent: Rs" + rent);
    System.out.println("Rating: " + rating + "/5");
    System.out.println("Gender: " + gender);
    if (!type.equalsIgnoreCase("Flat")) {
        System.out.println("Room Sharing: " + roomSharing + " sharing");
        System.out.println("Curfew: " + (curfew.equals("None") ? "No curfew" : curfew));
    } else {
```

```
            System.out.println("Private apartment");
        }
    }

    public void displayDetails() {
        displayBasicInformation();
        System.out.println("\nFacilities:");
        System.out.println("- WiFi: " + (wifi ? "Yes" : "No"));
        System.out.println("- Meals: " + (meals ? "Yes" : "No"));
        System.out.println("- Security: " + (security ? "Yes" : "No"));
        System.out.println("- Laundry: " + (laundry ? "Yes" : "No"));
        System.out.println("- Mess: " + (mess ? "Yes" : "No"));
        System.out.println("- AC: " + (ac ? "Yes" : "No"));
        System.out.println("- Geyser: " + (geyser ? "Yes" : "No"));
        System.out.println("- Kitchen: " + (kitchen ? "Yes" : "No"));
    }
}

class Hostel extends Accommodation {
    private final Boolean studyRoom;
    private final Boolean sportsFacility;

    public Hostel(String name, String location, Integer rent, Boolean wifi,
                  Boolean meals, Boolean security, Double rating, Integer roomSharing,
                  Boolean laundry, Boolean mess, Boolean ac, Boolean geyser,
                  Boolean kitchen, String curfew, String gender,
                  Boolean studyRoom, Boolean sportsFacility) {
        super(name, "Hostel", location, rent, wifi, meals, security, rating,
              roomSharing, laundry, mess, ac, geyser, kitchen, curfew, gender);
        this.studyRoom = studyRoom;
        this.sportsFacility = sportsFacility;
    }

    public void displayDetails() {
        super.displayDetails();
        System.out.println("Hostel-specific Facilities:");
        System.out.println("- Study Room: " + (studyRoom ? "Yes" : "No"));
        System.out.println("- Sports Facility: " + (sportsFacility ? "Yes" : "No"));
    }

    public Boolean hasStudyRoom() { return studyRoom; }
    public Boolean hasSportsFacility() { return sportsFacility; }
}

class PG extends Accommodation {
    private final Boolean parking;
    private final Boolean powerBackup;

    public PG(String name, String location, Integer rent, Boolean wifi,
              Boolean meals, Boolean security, Double rating, Integer roomSharing,
              Boolean laundry, Boolean mess, Boolean ac, Boolean geyser,
              Boolean kitchen, String curfew, String gender,
              Boolean parking, Boolean powerBackup) {
        super(name, "PG", location, rent, wifi, meals, security, rating,
```

```
            roomSharing, laundry, mess, ac, geyser, kitchen, curfew, gender);
        this.parking = parking;
        this.powerBackup = powerBackup;
    }

    public void displayDetails() {
        super.displayDetails();
        System.out.println("PG-specific Facilities:");
        System.out.println("- Parking: " + (parking ? "Yes" : "No"));
        System.out.println("- Power Backup: " + (powerBackup ? "Yes" : "No"));
    }

    public Boolean hasParking() { return parking; }
    public Boolean hasPowerBackup() { return powerBackup; }
}

class Flat extends Accommodation {
    private final Boolean furnished;
    private final Boolean maintenanceIncluded;

    public Flat(String name, String location, Integer rent, Boolean wifi,
                Boolean meals, Boolean security, Double rating,
                Boolean laundry, Boolean mess, Boolean ac, Boolean geyser,
                Boolean kitchen, String gender,
                Boolean furnished, Boolean maintenanceIncluded) {
        super(name, "Flat", location, rent, wifi, meals, security, rating,
              1, laundry, mess, ac, geyser, kitchen, "None", gender);
        this.furnished = furnished;
        this.maintenanceIncluded = maintenanceIncluded;
    }

    public void displayDetails() {
        super.displayDetails();
        System.out.println("Flat-specific Facilities:");
        System.out.println("- Furnished: " + (furnished ? "Yes" : "No"));
        System.out.println("- Maintenance Included: " + (maintenanceIncluded ? "Yes" : "No"));
    }

    public Boolean isFurnished() { return furnished; }
    public Boolean isMaintenanceIncluded() { return maintenanceIncluded; }
}

interface Authentication {
    Boolean login(String username, String password);
    void register(String username, String password);
    Boolean isAdmin();
}

class DatabaseManager {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/student_housing";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "Bun02309";

    static {
```

```
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "MySQL JDBC Driver not found!", "Database Error",
JOptionPane.ERROR_MESSAGE);
            System.exit(1);
        }
    }

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
    }

    public static void initializeDatabase() {
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement()) {
                stmt.executeUpdate("CREATE TABLE IF NOT EXISTS users (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY, " +
                    "username VARCHAR(50) UNIQUE NOT NULL, " +
                    "password VARCHAR(50) NOT NULL, " +
                    "is_admin BOOLEAN DEFAULT FALSE)");
                stmt.executeUpdate("INSERT IGNORE INTO users (username, password, is_admin) VALUES " +
                    "('admin', 'admin123', TRUE)");
                stmt.executeUpdate("CREATE TABLE IF NOT EXISTS accommodations (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY, " +
                    "name VARCHAR(100) NOT NULL, " +
                    "type VARCHAR(20) NOT NULL, " +
                    "location VARCHAR(100) NOT NULL, " +
                    "rent INT NOT NULL, " +
                    "wifi BOOLEAN, " +
                    "meals BOOLEAN, " +
                    "security BOOLEAN, " +
                    "rating DOUBLE, " +
                    "room_sharing INT, " +
                    "laundry BOOLEAN, " +
                    "mess BOOLEAN, " +
                    "ac BOOLEAN, " +
                    "geyser BOOLEAN, " +
                    "kitchen BOOLEAN, " +
                    "curfew VARCHAR(50), " +
                    "gender VARCHAR(20), " +
                    "study_room BOOLEAN, " +
                    "sports_facility BOOLEAN, " +
                    "parking BOOLEAN, " +
                    "power_backup BOOLEAN, " +
                    "furnished BOOLEAN, " +
                    "maintenance_included BOOLEAN)");

stmt.executeUpdate("CREATE TABLE IF NOT EXISTS bookings (" +
                    "id INT AUTO_INCREMENT PRIMARY KEY, " +
                    "customer_name VARCHAR(100) NOT NULL, " +
                    "contact VARCHAR(20) NOT NULL, " +
                    "university VARCHAR(100) NOT NULL, " +
```

```
                "accommodation_id INT NOT NULL, " +
                "confirmed BOOLEAN DEFAULT FALSE, " +
                "FOREIGN KEY (accommodation_id) REFERENCES accommodations(id))");

        stmt.executeUpdate("CREATE TABLE IF NOT EXISTS password_reset_requests (" +
                "id INT AUTO_INCREMENT PRIMARY KEY, " +
                "username VARCHAR(50) NOT NULL, " +
                "token VARCHAR(100) NOT NULL, " +
                "expiry DATETIME NOT NULL, " +
                "FOREIGN KEY (username) REFERENCES users(username))");

        stmt.executeUpdate("CREATE TABLE IF NOT EXISTS user_activity_log (" +
                "id INT AUTO_INCREMENT PRIMARY KEY, " +
                "username VARCHAR(50) NOT NULL, " +
                "activity_type VARCHAR(50) NOT NULL, " +
                "activity_details TEXT, " +
                "activity_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP, " +
                "FOREIGN KEY (username) REFERENCES users(username))");

    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Database initialization failed: " + e.getMessage(),
                "Database Error", JOptionPane.ERROR_MESSAGE);
        System.exit(1);
    }
}

public static void addPasswordResetRequest(String username, String token) throws SQLException {
    try (Connection conn = getConnection();
         PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO password_reset_requests (username, token, expiry) VALUES (?, ?, DATE_ADD(NOW(), INTERVAL
1 HOUR))")) {

        stmt.setString(1, username);
        stmt.setString(2, token);
        stmt.executeUpdate();
    }
}

public static boolean validatePasswordResetToken(String username, String token) throws SQLException {
    try (Connection conn = getConnection();
         PreparedStatement stmt = conn.prepareStatement(
            "SELECT * FROM password_reset_requests WHERE username = ? AND token = ? AND expiry > NOW()")) {

        stmt.setString(1, username);
        stmt.setString(2, token);
        ResultSet rs = stmt.executeQuery();
        return rs.next();
    }
}

public static void deletePasswordResetToken(String username) throws SQLException {
    try (Connection conn = getConnection();
         PreparedStatement stmt = conn.prepareStatement(
```

```
            "DELETE FROM password_reset_requests WHERE username = ?")) {

            stmt.setString(1, username);
            stmt.executeUpdate();
        }
    }

    public static void logActivity(String username, String activityType, String details) throws SQLException {
        try (Connection conn = getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                "INSERT INTO user_activity_log (username, activity_type, activity_details) VALUES (?, ?, ?)")) {

            stmt.setString(1, username);
            stmt.setString(2, activityType);
            stmt.setString(3, details);
            stmt.executeUpdate();
        }
    }
}

class UserAuthenticator implements Authentication {
    private String username;
    private Boolean isAdmin = false;

    public Boolean login(String username, String password) {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                "SELECT * FROM users WHERE username = ? AND password = ?")) {

            stmt.setString(1, username);
            stmt.setString(2, password);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                this.username = username;
                this.isAdmin = rs.getBoolean("is_admin");
                DatabaseManager.logActivity(username, "LOGIN", "User logged in");
                return true;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    public void register(String username, String password) {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                "INSERT INTO users (username, password) VALUES (?, ?)")) {

            stmt.setString(1, username);
            stmt.setString(2, password);
            stmt.executeUpdate();
```

```
            this.username = username;
            DatabaseManager.logActivity(username, "REGISTER", "New user registered");
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException("Registration failed: " + e.getMessage());
        }
    }

    public Boolean isAdmin() {
        return isAdmin;
    }
}

class AdminAuthenticator extends UserAuthenticator {
    @Override
    public Boolean login(String username, String password) {
        Boolean result = super.login(username, password);
        if (result && isAdmin()) {
            DatabaseManager.logActivity(username, "ADMIN_LOGIN", "Admin logged in");
            return true;
        }
        return false;
    }
}

class Booking {
    public Integer id;
    private String customerName;
    private String contact;
    private String university;
    private Accommodation accommodation;
    private Boolean confirmed;

    public Booking(Integer id, String customerName, String contact, String university,
                   Accommodation accommodation) {
        this.id = id;
        this.customerName = customerName;
        this.contact = contact;
        this.university = university;
        this.accommodation = accommodation;
        this.confirmed = false;
    }

    public Booking(Integer id, String customerName, String contact, String university,
                   Accommodation accommodation, Boolean confirmed) {
        this(id, customerName, contact, university, accommodation);
        this.confirmed = confirmed;
    }

    public Integer getId() { return id; }
    public String getCustomerName() { return customerName; }
    public String getContact() { return contact; }
    public String getUniversity() { return university; }
    public Accommodation getAccommodation() { return accommodation; }
```

```
    public Boolean isConfirmed() { return confirmed; }

    public void setConfirmed(Boolean confirmed) {
        this.confirmed = confirmed;
        try {
            DatabaseManager.logActivity(customerName, "BOOKING_UPDATE",
                "Booking " + id + " status changed to " + (confirmed ? "Confirmed" : "Pending"));
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void display() {
        System.out.println("\nBooking ID: " + id);
        System.out.println("Customer: " + customerName);
        System.out.println("Contact: " + contact);
        System.out.println("University: " + university);
        System.out.println("Status: " + (confirmed ? "Confirmed" : "Pending"));
        System.out.println("Accommodation Details:");
        accommodation.displayBasicInformation();
    }
}

class HousingManager {
    private List<Accommodation> accommodations;
    private UserAuthenticator authenticator;
    private String currentUser;
    private List<Booking> bookings;
    private Integer nextBookingId;
    private Map<String, List<Accommodation>> locationMap;
    private Map<String, List<Accommodation>> typeMap;

    // Swing components
    private JFrame loginFrame, mainFrame, adminFrame;
    private JTextField usernameField, passwordField;
    private JTextArea outputArea;

    public HousingManager() {
        DatabaseManager.initializeDatabase();
        accommodations = new ArrayList<>();
        authenticator = new UserAuthenticator();
        currentUser = null;
        bookings = new ArrayList<>();
        nextBookingId = 1;
        locationMap = new HashMap<>();
        typeMap = new HashMap<>();
        loadBookingsFromDatabase();
    }

    public void loadAccommodationsFromDatabase() {
        try (Connection conn = DatabaseManager.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM accommodations")) {
```

```
        while (rs.next()) {
            Accommodation acc = createAccommodationFromResultSet(rs);
            if (acc != null) {
                accommodations.add(acc);
                locationMap.computeIfAbsent(acc.getLocation(), k -> new ArrayList<>()).add(acc);
                typeMap.computeIfAbsent(acc.getType(), k -> new ArrayList<>()).add(acc);
            }
        }
        System.out.println("Loaded " + accommodations.size() + " accommodations from database.");
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Error loading accommodations from database: " + e.getMessage(),
                                      "Database Error", JOptionPane.ERROR_MESSAGE);
    }
}

private void loadBookingsFromDatabase() {
    try (Connection conn = DatabaseManager.getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT * FROM bookings")) {

        while (rs.next()) {
            int accId = rs.getInt("accommodation_id");
            Accommodation acc = findAccommodationById(accId);
            if (acc != null) {
                Booking booking = new Booking(
                    rs.getInt("id"),
                    rs.getString("customer_name"),
                    rs.getString("contact"),
                    rs.getString("university"),
                    acc,
                    rs.getBoolean("confirmed")
                );
                bookings.add(booking);
                if (rs.getInt("id") >= nextBookingId) {
                    nextBookingId = rs.getInt("id") + 1;
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private Accommodation findAccommodationById(int id) {
    try (Connection conn = DatabaseManager.getConnection();
         PreparedStatement stmt = conn.prepareStatement("SELECT * FROM accommodations WHERE id = ?")) {

        stmt.setInt(1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            return createAccommodationFromResultSet(rs);
        }
```

```
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}

private Accommodation createAccommodationFromResultSet(ResultSet rs) throws SQLException {
    String type = rs.getString("type");
    String name = rs.getString("name");
    String location = rs.getString("location");
    int rent = rs.getInt("rent");
    boolean wifi = rs.getBoolean("wifi");
    boolean meals = rs.getBoolean("meals");
    boolean security = rs.getBoolean("security");
    double rating = rs.getDouble("rating");
    int roomSharing = rs.getInt("room_sharing");
    boolean laundry = rs.getBoolean("laundry");
    boolean mess = rs.getBoolean("mess");
    boolean ac = rs.getBoolean("ac");
    boolean geyser = rs.getBoolean("geyser");
    boolean kitchen = rs.getBoolean("kitchen");
    String curfew = rs.getString("curfew");
    String gender = rs.getString("gender");

    if (type.equalsIgnoreCase("Hostel")) {
        boolean studyRoom = rs.getBoolean("study_room");
        boolean sportsFacility = rs.getBoolean("sports_facility");
        return new Hostel(name, location, rent, wifi, meals, security, rating,
                          roomSharing, laundry, mess, ac, geyser, kitchen,
                          curfew, gender, studyRoom, sportsFacility);
    } else if (type.equalsIgnoreCase("PG")) {
        boolean parking = rs.getBoolean("parking");
        boolean powerBackup = rs.getBoolean("power_backup");
        return new PG(name, location, rent, wifi, meals, security, rating,
                      roomSharing, laundry, mess, ac, geyser, kitchen,
                      curfew, gender, parking, powerBackup);
    } else if (type.equalsIgnoreCase("Flat")) {
        boolean furnished = rs.getBoolean("furnished");
        boolean maintenance = rs.getBoolean("maintenance_included");
        return new Flat(name, location, rent, wifi, meals, security, rating,
                        laundry, mess, ac, geyser, kitchen, gender,
                        furnished, maintenance);
    } else {
        return new Accommodation(name, type, location, rent, wifi, meals, security,
                                 rating, roomSharing, laundry, mess, ac, geyser,
                                 kitchen, curfew, gender);
    }
}

public void showLoginFrame() {
    loginFrame = new JFrame("Student Housing Management System - Login");
    loginFrame.setSize(400, 350);
    loginFrame.setLayout(null);
    loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
    JLabel titleLabel = new JLabel("Student Housing Management System");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    loginFrame.add(titleLabel);

    JLabel userLabel = new JLabel("Username:");
    userLabel.setBounds(50, 70, 100, 25);
    loginFrame.add(userLabel);

    usernameField = new JTextField();
    usernameField.setBounds(150, 70, 200, 25);
    loginFrame.add(usernameField);

    JLabel passLabel = new JLabel("Password:");
    passLabel.setBounds(50, 110, 100, 25);
    loginFrame.add(passLabel);

    passwordField = new JPasswordField();
    passwordField.setBounds(150, 110, 200, 25);
    loginFrame.add(passwordField);

    JButton loginButton = new JButton("Login");
    loginButton.setBounds(100, 160, 100, 30);
    loginButton.addActionListener(e -> handleLogin());
    loginFrame.add(loginButton);

    JButton registerButton = new JButton("Register");
    registerButton.setBounds(220, 160, 100, 30);
    registerButton.addActionListener(e -> handleRegistration());
    loginFrame.add(registerButton);

    JButton adminButton = new JButton("Admin Login");
    adminButton.setBounds(150, 200, 120, 30);
    adminButton.addActionListener(e -> handleAdminLogin());
    loginFrame.add(adminButton);

    JButton forgotPassButton = new JButton("Forgot Password?");
    forgotPassButton.setBounds(150, 240, 120, 30);
    forgotPassButton.addActionListener(e -> handleForgotPassword());
    loginFrame.add(forgotPassButton);

    loginFrame.setVisible(true);
}

private void handleLogin() {
    String username = usernameField.getText();
    String password = passwordField.getText();

    if (authenticator.login(username, password)) {
        currentUser = username;
        JOptionPane.showMessageDialog(loginFrame, "Login successful! Welcome, " + username + "!");
        loginFrame.dispose();
        if (authenticator.isAdmin()) {
```

```
            showAdminMenu();
        } else {
            showMainMenu();
        }
    } else {
        JOptionPane.showMessageDialog(loginFrame, "Invalid username or password. Please try again.");
    }
}

private void handleAdminLogin() {
    AdminAuthenticator adminAuth = new AdminAuthenticator();
    String username = usernameField.getText();
    String password = passwordField.getText();

    if (adminAuth.login(username, password)) {
        currentUser = username;
        JOptionPane.showMessageDialog(loginFrame, "Admin login successful! Welcome, " + username + "!");
        loginFrame.dispose();
        showAdminMenu();
    } else {
        JOptionPane.showMessageDialog(loginFrame, "Invalid admin credentials. Please try again.");
    }
}

private void handleRegistration() {
    String username = usernameField.getText();
    String password = passwordField.getText();

    if (username.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(loginFrame, "Username and password cannot be empty!");
        return;
    }

    try {
        authenticator.register(username, password);
        JOptionPane.showMessageDialog(loginFrame, "Registration successful! You can now login with your credentials.");
    } catch (RuntimeException e) {
        JOptionPane.showMessageDialog(loginFrame, "Registration failed: " + e.getMessage());
    }
}

private void handleForgotPassword() {
    JFrame forgotPassFrame = new JFrame("Password Recovery");
    forgotPassFrame.setSize(400, 250);
    forgotPassFrame.setLayout(null);

    JLabel titleLabel = new JLabel("Password Recovery");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    forgotPassFrame.add(titleLabel);

    JLabel userLabel = new JLabel("Username:");
    userLabel.setBounds(50, 70, 100, 25);
    forgotPassFrame.add(userLabel);
```

```
JTextField userField = new JTextField();
userField.setBounds(150, 70, 200, 25);
forgotPassFrame.add(userField);

JButton requestButton = new JButton("Request Reset Link");
requestButton.setBounds(50, 120, 150, 30);
requestButton.addActionListener(e -> {
    String username = userField.getText();
    if (username.isEmpty()) {
        JOptionPane.showMessageDialog(forgotPassFrame, "Please enter your username");
        return;

    String token = generateRandomToken();
    try {
        DatabaseManager.addPasswordResetRequest(username, token);
        JOptionPane.showMessageDialog(forgotPassFrame,
            "Password reset link has been sent (simulated).\n" +
            "Token: " + token + "\n" +
            "This token will expire in 1 hour.");
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(forgotPassFrame, "Error: " + ex.getMessage());
    }
});
forgotPassFrame.add(requestButton);

JButton resetButton = new JButton("Reset Password");
resetButton.setBounds(220, 120, 150, 30);
resetButton.addActionListener(e -> {
    String username = userField.getText();
    if (username.isEmpty()) {
        JOptionPane.showMessageDialog(forgotPassFrame, "Please enter your username");
        return;
    }

    String token = JOptionPane.showInputDialog(forgotPassFrame, "Enter your reset token:");
    if (token == null || token.isEmpty()) {
        return;
    }

    try {
        if (DatabaseManager.validatePasswordResetToken(username, token)) {
            String newPassword = JOptionPane.showInputDialog(forgotPassFrame, "Enter new password:");
            if (newPassword != null && !newPassword.isEmpty()) {
                if (updatePassword(username, newPassword)) {
                    DatabaseManager.deletePasswordResetToken(username);
                    JOptionPane.showMessageDialog(forgotPassFrame, "Password reset successfully!");
                    forgotPassFrame.dispose();
                } else {
                    JOptionPane.showMessageDialog(forgotPassFrame, "Failed to reset password.");
                }
            }
        } else {
            JOptionPane.showMessageDialog(forgotPassFrame, "Invalid or expired token.");
```

```
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(forgotPassFrame, "Error: " + ex.getMessage());
        }
    });
    forgotPassFrame.add(resetButton);

    JButton backButton = new JButton("Back to Login");
    backButton.setBounds(150, 170, 150, 30);
    backButton.addActionListener(e -> forgotPassFrame.dispose());
    forgotPassFrame.add(backButton);

    forgotPassFrame.setVisible(true);
}

private String generateRandomToken() {
    return Long.toHexString(Double.doubleToLongBits(Math.random()));
}

private void showMainMenu() {
    mainFrame = new JFrame("Student Housing System - Main Menu");
    mainFrame.setSize(700, 500);
    mainFrame.setLayout(null);
    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel titleLabel = new JLabel("Welcome, " + currentUser + "!");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    mainFrame.add(titleLabel);

    JLabel accLabel = new JLabel("Accommodation Operations:");
    accLabel.setBounds(50, 60, 200, 20);
    mainFrame.add(accLabel);

    JButton searchButton = new JButton("Search Accommodations");
    searchButton.setBounds(50, 90, 200, 30);
    searchButton.addActionListener(e -> searchAccommodations());
    mainFrame.add(searchButton);

    JButton viewAllButton = new JButton("View All Accommodations");
    viewAllButton.setBounds(50, 130, 200, 30);
    viewAllButton.addActionListener(e -> viewAllAccommodations());
    mainFrame.add(viewAllButton);

    JButton filterButton = new JButton("Filter Accommodations");
    filterButton.setBounds(50, 170, 200, 30);
    filterButton.addActionListener(e -> filterAccommodations());
    mainFrame.add(filterButton);

    JLabel userLabel = new JLabel("User Operations:");
    userLabel.setBounds(50, 210, 200, 20);
    mainFrame.add(userLabel);

    JButton viewBookingsButton = new JButton("View My Bookings");
```

```
    viewBookingsButton.setBounds(50, 240, 200, 30);
    viewBookingsButton.addActionListener(e -> viewUserBookings());
    mainFrame.add(viewBookingsButton);

    JButton profileButton = new JButton("My Profile");
    profileButton.setBounds(50, 280, 200, 30);
    profileButton.addActionListener(e -> showUserProfile());
    mainFrame.add(profileButton);

    JButton logoutButton = new JButton("Logout");
    logoutButton.setBounds(50, 320, 200, 30);
    logoutButton.addActionListener(e -> {
        try {
            DatabaseManager.logActivity(currentUser, "LOGOUT", "User logged out");
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        currentUser = null;
        mainFrame.dispose();
        showLoginFrame();
    });
    mainFrame.add(logoutButton);

    outputArea = new JTextArea();
    outputArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(outputArea);
    scrollPane.setBounds(300, 90, 350, 350);
    mainFrame.add(scrollPane);

    mainFrame.setVisible(true);
}

private void showUserProfile() {
    JFrame profileFrame = new JFrame("My Profile");
    profileFrame.setSize(400, 300);
    profileFrame.setLayout(null);

    JLabel titleLabel = new JLabel("User Profile: " + currentUser);
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    profileFrame.add(titleLabel);

    JButton changePassButton = new JButton("Change Password");
    changePassButton.setBounds(50, 70, 200, 30);
    changePassButton.addActionListener(e -> changePassword());
    profileFrame.add(changePassButton);

    JButton viewBookingsButton = new JButton("View My Bookings");
    viewBookingsButton.setBounds(50, 120, 200, 30);
    viewBookingsButton.addActionListener(e -> {
        profileFrame.dispose();
        viewUserBookings();
    });
    profileFrame.add(viewBookingsButton);
```

```
        JButton closeButton = new JButton("Close");
        closeButton.setBounds(50, 170, 200, 30);
        closeButton.addActionListener(e -> profileFrame.dispose());
        profileFrame.add(closeButton);

        profileFrame.setVisible(true);
    }

    private void changePassword() {
        JFrame passFrame = new JFrame("Change Password");
        passFrame.setSize(400, 250);
        passFrame.setLayout(null);

        JLabel titleLabel = new JLabel("Change Password for " + currentUser);
        titleLabel.setBounds(50, 20, 300, 30);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 14));
        passFrame.add(titleLabel);

        JLabel oldPassLabel = new JLabel("Current Password:");
        oldPassLabel.setBounds(50, 60, 150, 25);
        passFrame.add(oldPassLabel);

        JPasswordField oldPassField = new JPasswordField();
        oldPassField.setBounds(200, 60, 150, 25);
        passFrame.add(oldPassField);

        JLabel newPassLabel = new JLabel("New Password:");
        newPassLabel.setBounds(50, 100, 150, 25);
        passFrame.add(newPassLabel);

        JPasswordField newPassField = new JPasswordField();
        newPassField.setBounds(200, 100, 150, 25);
        passFrame.add(newPassField);

        JLabel confirmPassLabel = new JLabel("Confirm New Password:");
        confirmPassLabel.setBounds(50, 140, 150, 25);
    confirmPassField.setBounds(200, 140, 150, 25);
        passFrame.add(confirmPassField);

        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(50, 180, 100, 30);
        submitButton.addActionListener(e -> {
            String oldPass = new String(oldPassField.getPassword());
            String newPass = new String(newPassField.getPassword());
            String confirmPass = new String(confirmPassField.getPassword());

            if (newPass.isEmpty() || !newPass.equals(confirmPass)) {
                JOptionPane.showMessageDialog(passFrame, "New passwords don't match or are empty!");
                return;
            }

            if (updatePassword(currentUser, oldPass, newPass)) {
                try {
```

```
                DatabaseManager.logActivity(currentUser, "PASSWORD_CHANGE", "Password changed");
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            JOptionPane.showMessageDialog(passFrame, "Password changed successfully!");
            passFrame.dispose();
        } else {
            JOptionPane.showMessageDialog(passFrame, "Failed to change password. Check your current password.");
        }
    });
    passFrame.add(submitButton);

    JButton cancelButton = new JButton("Cancel");
    cancelButton.setBounds(200, 180, 100, 30);
    cancelButton.addActionListener(e -> passFrame.dispose());
    passFrame.add(cancelButton);

    passFrame.setVisible(true);
}

private boolean updatePassword(String username, String oldPassword, String newPassword) {
    try (Connection conn = DatabaseManager.getConnection()) {
        try (PreparedStatement verifyStmt = conn.prepareStatement(
                "SELECT * FROM users WHERE username = ? AND password = ?")) {
            verifyStmt.setString(1, username);
            verifyStmt.setString(2, oldPassword);
            ResultSet rs = verifyStmt.executeQuery();

            if (!rs.next()) {
                return false; // Old password doesn't match
            }
        }

        try (PreparedStatement updateStmt = conn.prepareStatement(
                "UPDATE users SET password = ? WHERE username = ?")) {
            updateStmt.setString(1, newPassword);
            updateStmt.setString(2, username);
            int rowsAffected = updateStmt.executeUpdate();
            return rowsAffected > 0;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

private boolean updatePassword(String username, String newPassword) {
    try (Connection conn = DatabaseManager.getConnection();
         PreparedStatement stmt = conn.prepareStatement(
             "UPDATE users SET password = ? WHERE username = ?")) {

        stmt.setString(1, newPassword);
        stmt.setString(2, username);
        int rowsAffected = stmt.executeUpdate();
```

```
            return rowsAffected > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    private void viewUserBookings() {
        List<Booking> userBookings = getUserBookings(currentUser);

        outputArea.setText("");
        outputArea.append("=== My Bookings ===\n");

        if (userBookings.isEmpty()) {
            outputArea.append("No bookings found.");
        } else {
            for (Booking booking : userBookings) {
                outputArea.append("\nBooking ID: " + booking.getId() + "\n");
                outputArea.append("Accommodation: " + booking.getAccommodation().getName() +
                        " (" + booking.getAccommodation().getType() + ")\n");
                outputArea.append("Location: " + booking.getAccommodation().getLocation() + "\n");
                outputArea.append("Rent: Rs" + booking.getAccommodation().getRent() + "\n");
                outputArea.append("Status: " + (booking.isConfirmed() ? "Confirmed" : "Pending") + "\n");
                outputArea.append("Contact: " + booking.getContact() + "\n");
                outputArea.append("University: " + booking.getUniversity() + "\n");
            }

            JButton cancelButton = new JButton("Cancel Booking");
            cancelButton.setBounds(50, 360, 200, 30);
            cancelButton.addActionListener(e -> cancelBooking(userBookings));
            mainFrame.add(cancelButton);
            mainFrame.revalidate();
            mainFrame.repaint();
        }
    }

    private List<Booking> getUserBookings(String username) {
        List<Booking> userBookings = new ArrayList<>();
        for (Booking booking : bookings) {
            if (booking.getCustomerName().equalsIgnoreCase(username)) {
                userBookings.add(booking);
            }
        }
        return userBookings;
    }

    private void cancelBooking(List<Booking> userBookings) {
        String[] bookingOptions = new String[userBookings.size()];
        for (int i = 0; i < userBookings.size(); i++) {
            Booking b = userBookings.get(i);
            bookingOptions[i] = "ID: " + b.getId() + " - " + b.getAccommodation().getName() +
                    " (" + (b.isConfirmed() ? "Confirmed" : "Pending") + ")";
        }
```

```
    String selected = (String) JOptionPane.showInputDialog(
        mainFrame,
        "Select booking to cancel:",
        "Cancel Booking",
        JOptionPane.QUESTION_MESSAGE,
        null,
        bookingOptions,
        bookingOptions[0]);

    if (selected == null) return;

    int index = -1;
    for (int i = 0; i < bookingOptions.length; i++) {
        if (bookingOptions[i].equals(selected)) {
            index = i;
            break;
        }
    }

    if (index == -1) return;

    Booking booking = userBookings.get(index);

    int confirm = JOptionPane.showConfirmDialog(
        mainFrame,
        "Are you sure you want to cancel booking ID " + booking.getId() + "?",
        "Confirm Cancellation",
        JOptionPane.YES_NO_OPTION);

    if (confirm == JOptionPane.YES_OPTION) {
        deleteBookingFromDatabase(booking);
        bookings.remove(booking);
        try {
            DatabaseManager.logActivity(currentUser, "BOOKING_CANCELLED",
                "Cancelled booking ID " + booking.getId());
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        JOptionPane.showMessageDialog(mainFrame, "Booking cancelled successfully!");
        viewUserBookings(); // Refresh the view
    }
}

private void deleteBookingFromDatabase(Booking booking) {
    try (Connection conn = DatabaseManager.getConnection();
         PreparedStatement stmt = conn.prepareStatement(
             "DELETE FROM bookings WHERE id = ?")) {

        stmt.setInt(1, booking.getId());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
private void showAdminMenu() {
    adminFrame = new JFrame("Admin Panel");
    adminFrame.setSize(700, 500);
    adminFrame.setLayout(null);
    adminFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel titleLabel = new JLabel("Admin Panel - Welcome, " + currentUser + "!");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    adminFrame.add(titleLabel);

    JLabel accLabel = new JLabel("Accommodation Management:");
    accLabel.setBounds(50, 60, 200, 20);
    adminFrame.add(accLabel);

    JButton addButton = new JButton("Add New Accommodation");
    addButton.setBounds(50, 90, 200, 30);
    addButton.addActionListener(e -> addNewAccommodation());
    adminFrame.add(addButton);

    JButton updateButton = new JButton("Update Accommodation");
    updateButton.setBounds(50, 130, 200, 30);
    updateButton.addActionListener(e -> updateAccommodation());
    adminFrame.add(updateButton);

    JButton deleteButton = new JButton("Delete Accommodation");
    deleteButton.setBounds(50, 170, 200, 30);
    deleteButton.addActionListener(e -> deleteAccommodation());
    adminFrame.add(deleteButton);

    JLabel bookingLabel = new JLabel("Booking Management:");
    bookingLabel.setBounds(50, 210, 200, 20);
    adminFrame.add(bookingLabel);

    JButton viewBookingsButton = new JButton("View All Bookings");
    viewBookingsButton.setBounds(50, 240, 200, 30);
    viewBookingsButton.addActionListener(e -> viewAllBookings());
    adminFrame.add(viewBookingsButton);

    JButton confirmBookingButton = new JButton("Confirm Booking");
    confirmBookingButton.setBounds(50, 280, 200, 30);
    confirmBookingButton.addActionListener(e -> confirmBooking());
    adminFrame.add(confirmBookingButton);

    JButton searchBookingButton = new JButton("Search Bookings");
    searchBookingButton.setBounds(50, 320, 200, 30);
    searchBookingButton.addActionListener(e -> searchBookings());
    adminFrame.add(searchBookingButton);

    JLabel userLabel = new JLabel("User Management:");
    userLabel.setBounds(50, 360, 200, 20);
    adminFrame.add(userLabel);
```

```
    JButton manageUsersButton = new JButton("Manage Users");
    manageUsersButton.setBounds(50, 390, 200, 30);
    manageUsersButton.addActionListener(e -> manageUsers());
    adminFrame.add(manageUsersButton);

    JButton logoutButton = new JButton("Logout");
    logoutButton.setBounds(50, 430, 200, 30);
    logoutButton.addActionListener(e -> {
        try {
            DatabaseManager.logActivity(currentUser, "ADMIN_LOGOUT", "Admin logged out");
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        currentUser = null;
        adminFrame.dispose();
        showLoginFrame();
    });
    adminFrame.add(logoutButton);

    outputArea = new JTextArea();
    outputArea.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(outputArea);
    scrollPane.setBounds(300, 90, 350, 350);
    adminFrame.add(scrollPane);

    adminFrame.setVisible(true);
}

private void searchBookings() {
    JFrame searchFrame = new JFrame("Search Bookings");
    searchFrame.setSize(400, 250);
    searchFrame.setLayout(null);

    JLabel titleLabel = new JLabel("Search Bookings");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    searchFrame.add(titleLabel);

    JButton byUserButton = new JButton("By User");
    byUserButton.setBounds(50, 70, 150, 30);
    byUserButton.addActionListener(e -> {
        searchFrame.dispose();
        searchBookingsByUser();
    });
    searchFrame.add(byUserButton);

    JButton byAccButton = new JButton("By Accommodation");
    byAccButton.setBounds(220, 70, 150, 30);
    byAccButton.addActionListener(e -> {
        searchFrame.dispose();
        searchBookingsByAccommodation();
    });
    searchFrame.add(byAccButton);
```

```
    JButton byStatusButton = new JButton("By Status");
    byStatusButton.setBounds(50, 120, 150, 30);
    byStatusButton.addActionListener(e -> {
        searchFrame.dispose();
        searchBookingsByStatus();
    });
    searchFrame.add(byStatusButton);

    JButton backButton = new JButton("Back");
    backButton.setBounds(220, 120, 150, 30);
    backButton.addActionListener(e -> searchFrame.dispose());
    searchFrame.add(backButton);

    searchFrame.setVisible(true);
}

private void searchBookingsByUser() {
    List<String> userNames = new ArrayList<>();
    for (Booking booking : bookings) {
        if (!userNames.contains(booking.getCustomerName())) {
            userNames.add(booking.getCustomerName());
        }
    }

    if (userNames.isEmpty()) {
        JOptionPane.showMessageDialog(adminFrame, "No bookings found.");
        return;
    }

    String selected = (String) JOptionPane.showInputDialog(
        adminFrame,
        "Select user:",
        "Search Bookings by User",
        JOptionPane.QUESTION_MESSAGE,
        null,
        userNames.toArray(),
        userNames.get(0));

    if (selected == null) return;

    List<Booking> filtered = new ArrayList<>();
    for (Booking booking : bookings) {
        if (booking.getCustomerName().equals(selected)) {
            filtered.add(booking);
        }
    }

    displayBookingList(filtered, "Bookings for " + selected);
}

private void searchBookingsByAccommodation() {
    List<String> accNames = new ArrayList<>();
    for (Booking booking : bookings) {
        String accName = booking.getAccommodation().getName() + " (" + booking.getAccommodation().getType() + ")";
```

```
            if (!accNames.contains(accName)) {
                accNames.add(accName);
            }
        }

        if (accNames.isEmpty()) {
            JOptionPane.showMessageDialog(adminFrame, "No bookings found.");
            return;
        }

        String selected = (String) JOptionPane.showInputDialog(
            adminFrame,
            "Select accommodation:",
            "Search Bookings by Accommodation",
            JOptionPane.QUESTION_MESSAGE,
            null,
            accNames.toArray(),
            accNames.get(0));

        if (selected == null) return;

        List<Booking> filtered = new ArrayList<>();
        for (Booking booking : bookings) {
            String accName = booking.getAccommodation().getName() + " (" + booking.getAccommodation().getType() + ")";
            if (accName.equals(selected)) {
                filtered.add(booking);
            }
        }

        displayBookingList(filtered, "Bookings for " + selected);
    }

    private void searchBookingsByStatus() {
        String[] options = {"Confirmed", "Pending"};
        String selected = (String) JOptionPane.showInputDialog(
            adminFrame,
            "Select status:",
            "Search Bookings by Status",
            JOptionPane.QUESTION_MESSAGE,
            null,
            options,
            options[0]);

        if (selected == null) return;

        boolean status = selected.equals("Confirmed");
        List<Booking> filtered = new ArrayList<>();
        for (Booking booking : bookings) {
            if (booking.isConfirmed() == status) {
                filtered.add(booking);
            }
        }

        displayBookingList(filtered, selected + " Bookings");
```

```
}

private void displayBookingList(List<Booking> bookingList, String title) {
    outputArea.setText(""); // Clear the output area
    outputArea.append("=== " + title + " ===\n");

    if (bookingList.isEmpty()) {
        outputArea.append("No bookings found.");
    } else {
        for (Booking booking : bookingList) {
            outputArea.append("\nBooking ID: " + booking.getId() + "\n");
            outputArea.append("Customer: " + booking.getCustomerName() + "\n");
            outputArea.append("Accommodation: " + booking.getAccommodation().getName() +
                    " (" + booking.getAccommodation().getType() + ")\n");
            outputArea.append("Status: " + (booking.isConfirmed() ? "Confirmed" : "Pending") + "\n");
            outputArea.append("Contact: " + booking.getContact() + "\n");
            outputArea.append("University: " + booking.getUniversity() + "\n");
        }
    }
}

private void manageUsers() {
    List<String> users = getAllUsers();

    if (users.isEmpty()) {
        JOptionPane.showMessageDialog(adminFrame, "No users found.");
        return;
    }

    String selected = (String) JOptionPane.showInputDialog(
        adminFrame,
        "Select user to manage:",
        "User Management",
        JOptionPane.QUESTION_MESSAGE,
        null,
        users.toArray(),
        users.get(0));

    if (selected == null) return;

    if (selected.equals("admin")) {
        JOptionPane.showMessageDialog(adminFrame, "Cannot modify admin account.");
        return;
    }

    String[] options = {"Reset Password", "Delete User", "Cancel"};
    int choice = JOptionPane.showOptionDialog(
        adminFrame,
        "Manage user: " + selected,
        "User Options",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.QUESTION_MESSAGE,
        null,
        options,
```

```
        options[0]);

    if (choice == 0) { // Reset Password
        resetUserPassword(selected);
    } else if (choice == 1) { // Delete User
        deleteUser(selected);
    }
}

private List<String> getAllUsers() {
    List<String> users = new ArrayList<>();
    try (Connection conn = DatabaseManager.getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT username FROM users")) {

        while (rs.next()) {
            users.add(rs.getString("username"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return users;
}

private void resetUserPassword(String username) {
    String newPassword = JOptionPane.showInputDialog(
        adminFrame,
        "Enter new password for " + username + ":",
        "Reset Password",
        JOptionPane.QUESTION_MESSAGE);

    if (newPassword == null || newPassword.isEmpty()) {
        JOptionPane.showMessageDialog(adminFrame, "Password cannot be empty!");
        return;
    }

    if (updatePassword(username, newPassword)) {
        try {
            DatabaseManager.logActivity(currentUser, "ADMIN_PASSWORD_RESET",
                "Reset password for user " + username);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        JOptionPane.showMessageDialog(adminFrame, "Password reset successfully!");
    } else {
        JOptionPane.showMessageDialog(adminFrame, "Failed to reset password.");
    }
}

private void deleteUser(String username) {
    int confirm = JOptionPane.showConfirmDialog(
        adminFrame,
        "Are you sure you want to delete user: " + username + "?",
        "Confirm User Deletion",
```

```
            JOptionPane.YES_NO_OPTION);

        if (confirm != JOptionPane.YES_OPTION) return;

        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                 "DELETE FROM users WHERE username = ?")) {

            stmt.setString(1, username);
            int rowsAffected = stmt.executeUpdate();

            if (rowsAffected > 0) {
                try {
                    DatabaseManager.logActivity(currentUser, "USER_DELETED",
                        "Deleted user " + username);
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
                JOptionPane.showMessageDialog(adminFrame, "User deleted successfully!");
            } else {
                JOptionPane.showMessageDialog(adminFrame, "Failed to delete user.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(adminFrame, "Error deleting user: " + e.getMessage());
        }
    }


}

    private void searchAccommodations() {
        JFrame searchFrame = new JFrame("Search Accommodations");
        searchFrame.setSize(400, 300);
        searchFrame.setLayout(null);

        JLabel titleLabel = new JLabel("Search Accommodations");
        titleLabel.setBounds(50, 20, 300, 30);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
        searchFrame.add(titleLabel);

        JButton pgButton = new JButton("PG (Paying Guest)");
        pgButton.setBounds(50, 70, 150, 30);
        pgButton.addActionListener(e -> {
            List<Accommodation> filtered = typeMap.getOrDefault("PG", new ArrayList<>());
            displayAccommodationList(filtered, "Available PG Accommodations");
            searchFrame.dispose();
        });
        searchFrame.add(pgButton);

        JButton hostelButton = new JButton("Hostel");
        hostelButton.setBounds(50, 120, 150, 30);
        hostelButton.addActionListener(e -> {
            List<Accommodation> filtered = typeMap.getOrDefault("Hostel", new ArrayList<>());
```

```
        displayAccommodationList(filtered, "Available Hostel Accommodations");
        searchFrame.dispose();
    });
    searchFrame.add(hostelButton);

    JButton flatButton = new JButton("Flat");
    flatButton.setBounds(50, 170, 150, 30);
    flatButton.addActionListener(e -> {
        List<Accommodation> filtered = typeMap.getOrDefault("Flat", new ArrayList<>());
        displayAccommodationList(filtered, "Available Flat Accommodations");
        searchFrame.dispose();
    });
    searchFrame.add(flatButton);

    JButton allButton = new JButton("View All Types");
    allButton.setBounds(220, 70, 150, 30);
    allButton.addActionListener(e -> {
        viewAllAccommodations();
        searchFrame.dispose();
    });
    searchFrame.add(allButton);

    JButton backButton = new JButton("Back");
    backButton.setBounds(220, 170, 150, 30);
    backButton.addActionListener(e -> searchFrame.dispose());
    searchFrame.add(backButton);

    searchFrame.setVisible(true);
}

private void viewAllAccommodations() {
    displayAccommodationList(accommodations, "All Available Accommodations");
}

private void displayAccommodationList(List<Accommodation> accList, String title) {
    outputArea.setText(""); // Clear the output area
    outputArea.append("=== " + title + " ===\n");

    if (accList.isEmpty()) {
        outputArea.append("No accommodations found.");
    } else {
        for (int i = 0; i < accList.size(); i++) {
            Accommodation acc = accList.get(i);
            outputArea.append((i+1) + ". " + acc.getName() +
                              " (" + acc.getType() + ") - " +
                              acc.getLocation() +
                              " (Rs" + acc.getRent() + ")\n");
        }
    }

    if (!accList.isEmpty()) {
        JButton viewDetailsButton = new JButton("View Details");
        viewDetailsButton.setBounds(50, 270, 200, 30);
        viewDetailsButton.addActionListener(e -> viewAccommodationDetails(accList));
```

```
        if (mainFrame != null) {
            mainFrame.add(viewDetailsButton);
            mainFrame.revalidate();
            mainFrame.repaint();
        } else if (adminFrame != null) {
            adminFrame.add(viewDetailsButton);
            adminFrame.revalidate();
            adminFrame.repaint();
        }
    }
}

private void viewAccommodationDetails(List<Accommodation> accList) {
    String input = JOptionPane.showInputDialog(mainFrame != null ? mainFrame : adminFrame,
        "Enter the number to view details (1-" + accList.size() + "):");

    try {
        int choice = Integer.parseInt(input);
        if (choice >= 1 && choice <= accList.size()) {
            Accommodation acc = accList.get(choice-1);

            outputArea.setText(""); // Clear the output area
            outputArea.append("\n=== Accommodation Details ===\n");
            outputArea.append("Type: " + acc.getType() + "\n");
            outputArea.append("Name: " + acc.getName() + "\n");
            outputArea.append("Location: " + acc.getLocation() + "\n");
            outputArea.append("Rent: Rs" + acc.getRent() + "\n");
            outputArea.append("Rating: " + acc.getRating() + "/5\n");
            outputArea.append("Gender: " + acc.getGender() + "\n");

            if (!acc.getType().equalsIgnoreCase("Flat")) {
                outputArea.append("Room Sharing: " + acc.getRoomSharing() + " sharing\n");
                outputArea.append("Curfew: " + (acc.getCurfew().equals("None") ? "No curfew" : acc.getCurfew()) + "\n");
            } else {
                outputArea.append("Private apartment\n");
            }

            outputArea.append("\nFacilities:\n");
            outputArea.append("- WiFi: " + (acc.hasWifi() ? "Yes" : "No") + "\n");
            outputArea.append("- Meals: " + (acc.hasMeals() ? "Yes" : "No") + "\n");
            outputArea.append("- Security: " + (acc.hasSecurity() ? "Yes" : "No") + "\n");
            outputArea.append("- Laundry: " + (acc.hasLaundry() ? "Yes" : "No") + "\n");
            outputArea.append("- Mess: " + (acc.hasMess() ? "Yes" : "No") + "\n");
            outputArea.append("- AC: " + (acc.hasAC() ? "Yes" : "No") + "\n");
            outputArea.append("- Geyser: " + (acc.hasGeyser() ? "Yes" : "No") + "\n");
            outputArea.append("- Kitchen: " + (acc.hasKitchen() ? "Yes" : "No") + "\n");

            if (acc instanceof Hostel) {
                Hostel h = (Hostel)acc;
                outputArea.append("\nHostel-specific Facilities:\n");
                outputArea.append("- Study Room: " + (h.hasStudyRoom() ? "Yes" : "No") + "\n");
                outputArea.append("- Sports Facility: " + (h.hasSportsFacility() ? "Yes" : "No") + "\n");
            } else if (acc instanceof PG) {
                PG pg = (PG)acc;
```

```
            outputArea.append("\nPG-specific Facilities:\n");
            outputArea.append("- Parking: " + (pg.hasParking() ? "Yes" : "No") + "\n");
            outputArea.append("- Power Backup: " + (pg.hasPowerBackup() ? "Yes" : "No") + "\n");
        } else if (acc instanceof Flat) {
            Flat f = (Flat)acc;
            outputArea.append("\nFlat-specific Facilities:\n");
            outputArea.append("- Furnished: " + (f.isFurnished() ? "Yes" : "No") + "\n");
            outputArea.append("- Maintenance Included: " + (f.isMaintenanceIncluded() ? "Yes" : "No") + "\n");
        }

        if (mainFrame != null) {
            int option = JOptionPane.showConfirmDialog(mainFrame,
                "Would you like to book this accommodation?", "Book Accommodation",
                JOptionPane.YES_NO_OPTION);

            if (option == JOptionPane.YES_OPTION) {
                bookAccommodation(acc);
            }
        }
    }
  } catch (NumberFormatException e) {
      JOptionPane.showMessageDialog(mainFrame != null ? mainFrame : adminFrame,
        "Please enter a valid number.");
  }
}

private void bookAccommodation(Accommodation accommodation) {
  JFrame bookFrame = new JFrame("Booking Process");
  bookFrame.setBounds(0,0,400, 350);
  bookFrame.setLayout(null);

  JLabel titleLabel = new JLabel("Booking: " + accommodation.getName());
  titleLabel.setBounds(50, 20, 300, 30);
  titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
  bookFrame.add(titleLabel);

  JLabel rentLabel = new JLabel("Rent: Rs" + accommodation.getRent());
  rentLabel.setBounds(50, 50, 200, 25);
  bookFrame.add(rentLabel);

  JLabel nameLabel = new JLabel("Your Full Name:");
  nameLabel.setBounds(50, 90, 150, 25);
  bookFrame.add(nameLabel);

  JTextField nameField = new JTextField();
  nameField.setBounds(200, 90, 150, 25);
  bookFrame.add(nameField);

  JLabel contactLabel = new JLabel("Contact Number:");
  contactLabel.setBounds(50, 130, 150, 25);
  bookFrame.add(contactLabel);

  JTextField contactField = new JTextField();
  contactField.setBounds(200, 130, 150, 25);
```

```
        bookFrame.add(contactField);

        JLabel uniLabel = new JLabel("University:");
        uniLabel.setBounds(50, 170, 150, 25);
        bookFrame.add(uniLabel);

        JTextField uniField = new JTextField();
        uniField.setBounds(200, 170, 150, 25);
        bookFrame.add(uniField);

        JButton submitButton = new JButton("Submit Booking");
        submitButton.setBounds(100, 220, 200, 30);
        submitButton.addActionListener(e -> {
            String name = nameField.getText();
            String contact = contactField.getText();
            String university = uniField.getText();

            if (name.isEmpty() || contact.isEmpty() || university.isEmpty()) {
                JOptionPane.showMessageDialog(bookFrame, "Please fill in all fields!");
                return;
            }

            Booking booking = new Booking(nextBookingId++, name, contact, university, accommodation);
            bookings.add(booking);
            saveBookingToDatabase(booking);

            try {
                DatabaseManager.logActivity(currentUser, "BOOKING_CREATED",
                    "Created booking ID " + booking.getId() + " for " + accommodation.getName());
            } catch (SQLException ex) {
                ex.printStackTrace();
            }

            JOptionPane.showMessageDialog(bookFrame,
                "Booking request submitted successfully!\n" +
                "Your booking ID is: " + booking.getId() + "\n" +
                "Please wait for admin confirmation.");
            bookFrame.dispose();
        });
        bookFrame.add(submitButton);

        JButton cancelButton = new JButton("Cancel");
        cancelButton.setBounds(100, 260, 200, 30);
        cancelButton.addActionListener(e -> bookFrame.dispose());
        bookFrame.add(cancelButton);

        bookFrame.setVisible(true);
    }

    private void saveBookingToDatabase(Booking booking) {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                "INSERT INTO bookings (customer_name, contact, university, accommodation_id) " +
                "VALUES (?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS)) {
```

```
            int accId = getAccommodationId(booking.getAccommodation());
            if (accId == -1) {
                JOptionPane.showMessageDialog(null, "Error: Accommodation not found in database", "Database Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            stmt.setString(1, booking.getCustomerName());
            stmt.setString(2, booking.getContact());
            stmt.setString(3, booking.getUniversity());
            stmt.setInt(4, accId);
            stmt.executeUpdate();

            try (ResultSet rs = stmt.getGeneratedKeys()) {
                if (rs.next()) {
                    booking.id = rs.getInt(1);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private int getAccommodationId(Accommodation acc) {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                 "SELECT id FROM accommodations WHERE name = ? AND type = ? AND location = ?")) {

            stmt.setString(1, acc.getName());
            stmt.setString(2, acc.getType());
            stmt.setString(3, acc.getLocation());
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                return rs.getInt("id");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return -1;
    }

    private void filterAccommodations() {
        JFrame filterFrame = new JFrame("Filter Accommodations");
        filterFrame.setSize(400, 400);
        filterFrame.setLayout(null);

        JLabel titleLabel = new JLabel("Filter Accommodations");
        titleLabel.setBounds(50, 20, 300, 30);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
        filterFrame.add(titleLabel);

        JButton locationButton = new JButton("By Location");
```

```
        locationButton.setBounds(50, 70, 150, 30);
        locationButton.addActionListener(e -> {
            filterFrame.dispose();
            filterByLocation();
        });
        filterFrame.add(locationButton);

        JButton rentButton = new JButton("By Rent Range");
        rentButton.setBounds(220, 70, 150, 30);
        rentButton.addActionListener(e -> {
            filterFrame.dispose();
            filterByRent();
        });
        filterFrame.add(rentButton);

        JButton facilitiesButton = new JButton("By Facilities");
        facilitiesButton.setBounds(50, 120, 150, 30);
        facilitiesButton.addActionListener(e -> {
            filterFrame.dispose();
            filterByFacilities();
        });
        filterFrame.add(facilitiesButton);

        JButton ratingButton = new JButton("By Rating");
        ratingButton.setBounds(220, 120, 150, 30);
        ratingButton.addActionListener(e -> {
            filterFrame.dispose();
            filterByRating();
        });
        filterFrame.add(ratingButton);

        JButton genderButton = new JButton("By Gender");
        genderButton.setBounds(50, 170, 150, 30);
        genderButton.addActionListener(e -> {
            filterFrame.dispose();
            filterByGender();
        });
        filterFrame.add(genderButton);

        JButton backButton = new JButton("Back");
        backButton.setBounds(220, 170, 150, 30);
        backButton.addActionListener(e -> filterFrame.dispose());
        filterFrame.add(backButton);

        filterFrame.setVisible(true);
    }

    private void filterByLocation() {
        String[] locations = locationMap.keySet().toArray(new String[0]);
        String selectedLocation = (String) JOptionPane.showInputDialog(
            mainFrame != null ? mainFrame : adminFrame,
            "Select location:",
            "Filter by Location",
            JOptionPane.QUESTION_MESSAGE,
```

```
            null,
            locations,
            locations[0]);

        if (selectedLocation != null) {
            List<Accommodation> filtered = locationMap.get(selectedLocation);
            displayAccommodationList(filtered, "Accommodations in " + selectedLocation);
        }
    }

    private void filterByRent() {
        JFrame rentFrame = new JFrame("Filter by Rent");
        rentFrame.setSize(350, 200);
        rentFrame.setLayout(null);

        JLabel minLabel = new JLabel("Minimum Rent:");
        minLabel.setBounds(50, 30, 100, 25);
        rentFrame.add(minLabel);

        JTextField minField = new JTextField();
        minField.setBounds(160, 30, 100, 25);
        rentFrame.add(minField);

        JLabel maxLabel = new JLabel("Maximum Rent:");
        maxLabel.setBounds(50, 70, 100, 25);
        rentFrame.add(maxLabel);

        JTextField maxField = new JTextField();
        maxField.setBounds(160, 70, 100, 25);
        rentFrame.add(maxField);

        JButton filterButton = new JButton("Filter");
        filterButton.setBounds(50, 120, 100, 30);
        filterButton.addActionListener(e -> {
            try {
                int min = Integer.parseInt(minField.getText());
                int max = Integer.parseInt(maxField.getText());

                if (min > max) {
                    JOptionPane.showMessageDialog(rentFrame, "Minimum rent cannot be greater than maximum rent!");
                    return;
                }

                List<Accommodation> filtered = new ArrayList<>();
                for (Accommodation acc : accommodations) {
                    if (acc.getRent() >= min && acc.getRent() <= max) {
                        filtered.add(acc);
                    }
                }

                displayAccommodationList(filtered, "Accommodations between Rs" + min + " and Rs" + max);
                rentFrame.dispose();
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(rentFrame, "Please enter valid numbers for rent!");
```

```
        }
    });
    rentFrame.add(filterButton);

    JButton cancelButton = new JButton("Cancel");
    cancelButton.setBounds(160, 120, 100, 30);
    cancelButton.addActionListener(e -> rentFrame.dispose());
    rentFrame.add(cancelButton);

    rentFrame.setVisible(true);
}

private void filterByFacilities() {
    JFrame facilitiesFrame = new JFrame("Filter by Facilities");
    facilitiesFrame.setSize(300, 400);
    facilitiesFrame.setLayout(null);

    JLabel titleLabel = new JLabel("Select Facilities:");
    titleLabel.setBounds(50, 20, 200, 25);
    facilitiesFrame.add(titleLabel);

    JCheckBox wifiCheck = new JCheckBox("WiFi");
    wifiCheck.setBounds(50, 50, 100, 25);
    facilitiesFrame.add(wifiCheck);

    JCheckBox mealsCheck = new JCheckBox("Meals");
    mealsCheck.setBounds(50, 80, 100, 25);
    facilitiesFrame.add(mealsCheck);

    JCheckBox securityCheck = new JCheckBox("Security");
    securityCheck.setBounds(50, 110, 100, 25);
    facilitiesFrame.add(securityCheck);

    JCheckBox laundryCheck = new JCheckBox("Laundry");
    laundryCheck.setBounds(50, 140, 100, 25);
    facilitiesFrame.add(laundryCheck);

    JCheckBox acCheck = new JCheckBox("AC");
    acCheck.setBounds(50, 170, 100, 25);
    facilitiesFrame.add(acCheck);

    JCheckBox geyserCheck = new JCheckBox("Geyser");
    geyserCheck.setBounds(50, 200, 100, 25);
    facilitiesFrame.add(geyserCheck);

    JCheckBox kitchenCheck = new JCheckBox("Kitchen");
    kitchenCheck.setBounds(50, 230, 100, 25);
    facilitiesFrame.add(kitchenCheck);

    JButton filterButton = new JButton("Filter");
    filterButton.setBounds(50, 260, 100, 30);
    filterButton.addActionListener(e -> {
        List<Accommodation> filtered = new ArrayList<>();
        for (Accommodation acc : accommodations) {
```

```
            boolean matches = true;

            if (wifiCheck.isSelected() && !acc.hasWifi()) matches = false;
            if (mealsCheck.isSelected() && !acc.hasMeals()) matches = false;
            if (securityCheck.isSelected() && !acc.hasSecurity()) matches = false;
            if (laundryCheck.isSelected() && !acc.hasLaundry()) matches = false;
            if (acCheck.isSelected() && !acc.hasAC()) matches = false;
            if (geyserCheck.isSelected() && !acc.hasGeyser()) matches = false;
            if (kitchenCheck.isSelected() && !acc.hasKitchen()) matches = false;

            if (matches) filtered.add(acc);
        }

        displayAccommodationList(filtered, "Accommodations with selected facilities");
        facilitiesFrame.dispose();
    });
    facilitiesFrame.add(filterButton);

    JButton cancelButton = new JButton("Cancel");
    cancelButton.setBounds(160, 260, 100, 30);
    cancelButton.addActionListener(e -> facilitiesFrame.dispose());
    facilitiesFrame.add(cancelButton);

    facilitiesFrame.setVisible(true);
}

private void filterByRating() {
    String input = JOptionPane.showInputDialog(mainFrame != null ? mainFrame : adminFrame,
        "Enter minimum rating (1-5):");

    try {
        double minRating = Double.parseDouble(input);
        if (minRating >= 1.0 && minRating <= 5.0) {
            List<Accommodation> filtered = new ArrayList<>();
            for (Accommodation acc : accommodations) {
                if (acc.getRating() >= minRating) {
                    filtered.add(acc);
                }
            }
            displayAccommodationList(filtered, "Accommodations with rating " + minRating + "+");
        } else {
            JOptionPane.showMessageDialog(mainFrame != null ? mainFrame : adminFrame,
                "Please enter a rating between 1 and 5.");
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(mainFrame != null ? mainFrame : adminFrame,
            "Please enter a valid number.");
    }
}

private void filterByGender() {
    String[] options = {"Male", "Female", "Unisex"};
    String selected = (String) JOptionPane.showInputDialog(
        mainFrame != null ? mainFrame : adminFrame,
```

```
        "Select gender preference:",
        "Filter by Gender",
        JOptionPane.QUESTION_MESSAGE,
        null,
        options,
        options[0]);

    if (selected != null) {
        List<Accommodation> filtered = new ArrayList<>();
        for (Accommodation acc : accommodations) {
            if (acc.getGender().equalsIgnoreCase(selected)) {
                filtered.add(acc);
            }
        }
        displayAccommodationList(filtered, selected + " Accommodations");
    }
}

private void addNewAccommodation() {
    JFrame addFrame = new JFrame("Add New Accommodation");
    addFrame.setSize(500, 600);
    addFrame.setLayout(null);

    JLabel titleLabel = new JLabel("Add New Accommodation");
    titleLabel.setBounds(50, 20, 300, 30);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    addFrame.add(titleLabel);

    JLabel typeLabel = new JLabel("Accommodation Type:");
    typeLabel.setBounds(50, 60, 150, 25);
    addFrame.add(typeLabel);

    String[] types = {"Hostel", "PG", "Flat"};
    JComboBox<String> typeCombo = new JComboBox<>(types);
    typeCombo.setBounds(200, 60, 150, 25);
    addFrame.add(typeCombo);

    JLabel nameLabel = new JLabel("Name:");
    nameLabel.setBounds(50, 100, 150, 25);
    addFrame.add(nameLabel);

    JTextField nameField = new JTextField();
    nameField.setBounds(200, 100, 150, 25);
    addFrame.add(nameField);

    JLabel locationLabel = new JLabel("Location:");
    locationLabel.setBounds(50, 140, 150, 25);
    addFrame.add(locationLabel);

    JTextField locationField = new JTextField();
    locationField.setBounds(200, 140, 150, 25);
    addFrame.add(locationField);

    JLabel rentLabel = new JLabel("Rent:");
```

```
rentLabel.setBounds(50, 180, 150, 25);
addFrame.add(rentLabel);

JTextField rentField = new JTextField();
rentField.setBounds(200, 180, 150, 25);
addFrame.add(rentField);

JLabel ratingLabel = new JLabel("Rating (1-5):");
ratingLabel.setBounds(50, 220, 150, 25);
addFrame.add(ratingLabel);

JTextField ratingField = new JTextField();
ratingField.setBounds(200, 220, 150, 25);
addFrame.add(ratingField);

JLabel genderLabel = new JLabel("Gender:");
genderLabel.setBounds(50, 260, 150, 25);
addFrame.add(genderLabel);

String[] genders = {"Male", "Female", "Unisex"};
JComboBox<String> genderCombo = new JComboBox<>(genders);
genderCombo.setBounds(200, 260, 150, 25);
addFrame.add(genderCombo);

JLabel facilitiesLabel = new JLabel("Facilities:");
facilitiesLabel.setBounds(50, 300, 150, 25);
addFrame.add(facilitiesLabel);

JCheckBox wifiCheck = new JCheckBox("WiFi");
wifiCheck.setBounds(200, 300, 100, 25);
addFrame.add(wifiCheck);

JCheckBox mealsCheck = new JCheckBox("Meals");
mealsCheck.setBounds(200, 330, 100, 25);
addFrame.add(mealsCheck);

JCheckBox securityCheck = new JCheckBox("Security");
securityCheck.setBounds(200, 360, 100, 25);
addFrame.add(securityCheck);

JCheckBox laundryCheck = new JCheckBox("Laundry");
laundryCheck.setBounds(200, 390, 100, 25);
addFrame.add(laundryCheck);

JCheckBox acCheck = new JCheckBox("AC");
acCheck.setBounds(200, 420, 100, 25);
addFrame.add(acCheck);

JCheckBox geyserCheck = new JCheckBox("Geyser");
geyserCheck.setBounds(200, 450, 100, 25);
addFrame.add(geyserCheck);

JCheckBox kitchenCheck = new JCheckBox("Kitchen");
kitchenCheck.setBounds(200, 480, 100, 25);
```

```
        addFrame.add(kitchenCheck);

        JButton submitButton = new JButton("Submit");
        submitButton.setBounds(100, 520, 100, 30);
        submitButton.addActionListener(e -> {
            try {
                String type = (String) typeCombo.getSelectedItem();
                String name = nameField.getText();
                String location = locationField.getText();
                int rent = Integer.parseInt(rentField.getText());
                double rating = Double.parseDouble(ratingField.getText());
                String gender = (String) genderCombo.getSelectedItem();

                boolean wifi = wifiCheck.isSelected();
                boolean meals = mealsCheck.isSelected();
                boolean security = securityCheck.isSelected();
                boolean laundry = laundryCheck.isSelected();
                boolean ac = acCheck.isSelected();
                boolean geyser = geyserCheck.isSelected();
                boolean kitchen = kitchenCheck.isSelected();

                // Create the appropriate accommodation type
                Accommodation newAcc;
                if (type.equals("Hostel")) {
                    int roomSharing = Integer.parseInt(JOptionPane.showInputDialog(addFrame, "Enter room sharing (number of
people):"));
                    String curfew = JOptionPane.showInputDialog(addFrame, "Enter curfew time (or 'None'):");

                    boolean studyRoom = JOptionPane.showConfirmDialog(addFrame, "Has study room?", "Study Room",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;
                    boolean sportsFacility = JOptionPane.showConfirmDialog(addFrame, "Has sports facility?", "Sports Facility",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;

                    newAcc = new Hostel(name, location, rent, wifi, meals, security, rating,
                            roomSharing, laundry, false, ac, geyser, kitchen,
                            curfew, gender, studyRoom, sportsFacility);
                }
                else if (type.equals("PG")) {
                    int roomSharing = Integer.parseInt(JOptionPane.showInputDialog(addFrame, "Enter room sharing (number of
people):"));
                    String curfew = JOptionPane.showInputDialog(addFrame, "Enter curfew time (or 'None'):");

                    boolean parking = JOptionPane.showConfirmDialog(addFrame, "Has parking?", "Parking",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;
                    boolean powerBackup = JOptionPane.showConfirmDialog(addFrame, "Has power backup?", "Power Backup",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;

                    newAcc = new PG(name, location, rent, wifi, meals, security, rating,
                            roomSharing, laundry, false, ac, geyser, kitchen,
                            curfew, gender, parking, powerBackup);
                }
                else { // Flat
                    boolean furnished = JOptionPane.showConfirmDialog(addFrame, "Is furnished?", "Furnished",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;
```

```
                boolean maintenance = JOptionPane.showConfirmDialog(addFrame, "Maintenance included?", "Maintenance",
JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;

                newAcc = new Flat(name, location, rent, wifi, meals, security, rating,
                        laundry, false, ac, geyser, kitchen, gender,
                        furnished, maintenance);
            }

            accommodations.add(newAcc);
            locationMap.computeIfAbsent(location, k -> new ArrayList<>()).add(newAcc);
            typeMap.computeIfAbsent(type, k -> new ArrayList<>()).add(newAcc);
            saveAccommodationToDatabase(newAcc);

            try {
                DatabaseManager.logActivity(currentUser, "ACCOMMODATION_ADDED",
                    "Added " + type + ": " + name);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }

            JOptionPane.showMessageDialog(addFrame, "Accommodation added successfully!");
            addFrame.dispose();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(addFrame, "Please enter valid numbers for rent and rating!");
        }
    });
    addFrame.add(submitButton);

    JButton cancelButton = new JButton("Cancel");
    cancelButton.setBounds(220, 520, 100, 30);
    cancelButton.addActionListener(e -> addFrame.dispose());
    addFrame.add(cancelButton);

    addFrame.setVisible(true);
}

private void saveAccommodationToDatabase(Accommodation acc) {
    try (Connection conn = DatabaseManager.getConnection();
         PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO accommodations (name, type, location, rent, wifi, meals, security, " +
            "rating, room_sharing, laundry, mess, ac, geyser, kitchen, curfew, gender, " +
            "study_room, sports_facility, parking, power_backup, furnished, maintenance_included) " +
            "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)")) {

        stmt.setString(1, acc.getName());
        stmt.setString(2, acc.getType());
        stmt.setString(3, acc.getLocation());
        stmt.setInt(4, acc.getRent());
        stmt.setBoolean(5, acc.hasWifi());
        stmt.setBoolean(6, acc.hasMeals());
        stmt.setBoolean(7, acc.hasSecurity());
        stmt.setDouble(8, acc.getRating());
        stmt.setInt(9, acc.getRoomSharing());
        stmt.setBoolean(10, acc.hasLaundry());
```

```
            stmt.setBoolean(11, acc.hasMess());
            stmt.setBoolean(12, acc.hasAC());
            stmt.setBoolean(13, acc.hasGeyser());
            stmt.setBoolean(14, acc.hasKitchen());
            stmt.setString(15, acc.getCurfew());
            stmt.setString(16, acc.getGender());

            if (acc instanceof Hostel) {
                Hostel h = (Hostel) acc;
                stmt.setBoolean(17, h.hasStudyRoom());
                stmt.setBoolean(18, h.hasSportsFacility());
                stmt.setNull(19, Types.BOOLEAN);
                stmt.setNull(20, Types.BOOLEAN);
                stmt.setNull(21, Types.BOOLEAN);
                stmt.setNull(22, Types.BOOLEAN);
            } else if (acc instanceof PG) {
                PG pg = (PG) acc;
                stmt.setNull(17, Types.BOOLEAN);
                stmt.setNull(18, Types.BOOLEAN);
                stmt.setBoolean(19, pg.hasParking());
                stmt.setBoolean(20, pg.hasPowerBackup());
                stmt.setNull(21, Types.BOOLEAN);
                stmt.setNull(22, Types.BOOLEAN);
            } else if (acc instanceof Flat) {
                Flat f = (Flat) acc;
                stmt.setNull(17, Types.BOOLEAN);
                stmt.setNull(18, Types.BOOLEAN);
                stmt.setNull(19, Types.BOOLEAN);
                stmt.setNull(20, Types.BOOLEAN);
                stmt.setBoolean(21, f.isFurnished());
                stmt.setBoolean(22, f.isMaintenanceIncluded());
            } else {
                stmt.setNull(17, Types.BOOLEAN);
                stmt.setNull(18, Types.BOOLEAN);
                stmt.setNull(19, Types.BOOLEAN);
                stmt.setNull(20, Types.BOOLEAN);
                stmt.setNull(21, Types.BOOLEAN);
                stmt.setNull(22, Types.BOOLEAN);
            }

            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void updateAccommodation() {
        if (accommodations.isEmpty()) {
            JOptionPane.showMessageDialog(adminFrame, "No accommodations available to update.");
            return;
        }

        String[] accNames = new String[accommodations.size()];
        for (int i = 0; i < accommodations.size(); i++) {
```

```
    accNames[i] = accommodations.get(i).getName() + " (" + accommodations.get(i).getType() + ")";
}

String selected = (String) JOptionPane.showInputDialog(
    adminFrame,
    "Select accommodation to update:",
    "Update Accommodation",
    JOptionPane.QUESTION_MESSAGE,
    null,
    accNames,
    accNames[0]);

if (selected == null) return;

int index = -1;
for (int i = 0; i < accNames.length; i++) {
    if (accNames[i].equals(selected)) {
        index = i;
        break;
    }
}

if (index == -1) return;

Accommodation acc = accommodations.get(index);

String[] options = {"Name", "Location", "Rent", "Rating", "Facilities", "Cancel"};
int choice = JOptionPane.showOptionDialog(
    adminFrame,
    "Select field to update for " + acc.getName() + ":",
    "Update Field",
    JOptionPane.DEFAULT_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,
    options,
    options[0]);

if (choice == 5 || choice == -1) return;

switch (choice) {
    case 0:
        String newName = JOptionPane.showInputDialog(adminFrame, "Enter new name:", acc.getName());
        if (newName != null && !newName.isEmpty()) {
            accommodations.set(index, createUpdatedAccommodation(acc, "name", newName));
            updateAccommodationInDatabase(accommodations.get(index));
            JOptionPane.showMessageDialog(adminFrame, "Name updated successfully!");
        }
        break;
    case 1:
        String newLocation = JOptionPane.showInputDialog(adminFrame, "Enter new location:", acc.getLocation());
        if (newLocation != null && !newLocation.isEmpty()) {
            accommodations.set(index, createUpdatedAccommodation(acc, "location", newLocation));
            updateAccommodationInDatabase(accommodations.get(index));
            JOptionPane.showMessageDialog(adminFrame, "Location updated successfully!");
```

```
            }
            break;
        case 2:
            String rentStr = JOptionPane.showInputDialog(adminFrame, "Enter new rent:", acc.getRent());
            try {
                int newRent = Integer.parseInt(rentStr);
                accommodations.set(index, createUpdatedAccommodation(acc, "rent", newRent));
                updateAccommodationInDatabase(accommodations.get(index));
                JOptionPane.showMessageDialog(adminFrame, "Rent updated successfully!");
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(adminFrame, "Please enter a valid number for rent!");
            }
            break;
        case 3:
String ratingStr = JOptionPane.showInputDialog(adminFrame, "Enter new rating (1-5):", acc.getRating());
            try {
                double newRating = Double.parseDouble(ratingStr);
                if (newRating >= 1.0 && newRating <= 5.0) {
                    accommodations.set(index, createUpdatedAccommodation(acc, "rating", newRating));
                    updateAccommodationInDatabase(accommodations.get(index));
                    JOptionPane.showMessageDialog(adminFrame, "Rating updated successfully!");
                } else {
                    JOptionPane.showMessageDialog(adminFrame, "Please enter a rating between 1 and 5!");
                }
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(adminFrame, "Please enter a valid number for rating!");
            }
            break;
        case 4: // Facilities
            updateFacilities(index);
            break;
    }
  }

  private void updateAccommodationInDatabase(Accommodation acc) {
     try (Connection conn = DatabaseManager.getConnection();
          PreparedStatement stmt = conn.prepareStatement(
             "UPDATE accommodations SET name = ?, location = ?, rent = ?, rating = ?, " +
             "wifi = ?, meals = ?, security = ?, laundry = ?, ac = ?, geyser = ?, kitchen = ? " +
             "WHERE name = ? AND type = ? AND location = ?")) {

         stmt.setString(1, acc.getName());
         stmt.setString(2, acc.getLocation());
         stmt.setInt(3, acc.getRent());
         stmt.setDouble(4, acc.getRating());
         stmt.setBoolean(5, acc.hasWifi());
         stmt.setBoolean(6, acc.hasMeals());
         stmt.setBoolean(7, acc.hasSecurity());
         stmt.setBoolean(8, acc.hasLaundry());
         stmt.setBoolean(9, acc.hasAC());
         stmt.setBoolean(10, acc.hasGeyser());
         stmt.setBoolean(11, acc.hasKitchen());

         // Original values for WHERE clause
```

```
            stmt.setString(12, acc.getName());
            stmt.setString(13, acc.getType());
            stmt.setString(14, acc.getLocation());

            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void deleteAccommodation() {
        if (accommodations.isEmpty()) {
            JOptionPane.showMessageDialog(adminFrame, "No accommodations available to delete.");
            return;
        }

        String[] accNames = new String[accommodations.size()];
        for (int i = 0; i < accommodations.size(); i++) {
            accNames[i] = accommodations.get(i).getName() + " (" + accommodations.get(i).getType() + ")";
        }

        String selected = (String) JOptionPane.showInputDialog(
            adminFrame,
            "Select accommodation to delete:",
            "Delete Accommodation",
            JOptionPane.QUESTION_MESSAGE,
            null,
            accNames,
            accNames[0]);

        if (selected == null) return;

        int index = -1;
        for (int i = 0; i < accNames.length; i++) {
            if (accNames[i].equals(selected)) {
                index = i;
                break;
            }
        }

        if (index == -1) return;

        Accommodation acc = accommodations.get(index);

        int confirm = JOptionPane.showConfirmDialog(
            adminFrame,
            "Are you sure you want to delete:\n" + acc.getName() + " (" + acc.getType() + ")",
            "Confirm Deletion",
            JOptionPane.YES_NO_OPTION);

        if (confirm == JOptionPane.YES_OPTION) {
            deleteAccommodationFromDatabase(acc);

            // Remove from main list
```

```
        Accommodation removed = accommodations.remove(index);
        // Remove from location map
        locationMap.get(removed.getLocation()).remove(removed);
        // Remove from type map
        typeMap.get(removed.getType()).remove(removed);

        JOptionPane.showMessageDialog(adminFrame, "Accommodation deleted successfully!");
    }
}

private void deleteAccommodationFromDatabase(Accommodation acc) {
    try (Connection conn = DatabaseManager.getConnection();
         PreparedStatement stmt = conn.prepareStatement(
             "DELETE FROM accommodations WHERE name = ? AND type = ? AND location = ?")) {

        stmt.setString(1, acc.getName());
        stmt.setString(2, acc.getType());
        stmt.setString(3, acc.getLocation());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private Accommodation createUpdatedAccommodation(Accommodation original, String field, Object value) {
    if (original instanceof Hostel) {
        Hostel h = (Hostel)original;
        return new Hostel(
            field.equals("name") ? (String)value : h.getName(),
            field.equals("location") ? (String)value : h.getLocation(),
            field.equals("rent") ? (Integer)value : h.getRent(),
            h.hasWifi(), h.hasMeals(), h.hasSecurity(),
            field.equals("rating") ? (Double)value : h.getRating(),
            h.getRoomSharing(), h.hasLaundry(), h.hasMess(),
            h.hasAC(), h.hasGeyser(), h.hasKitchen(),
            h.getCurfew(), h.getGender(),
            h.hasStudyRoom(), h.hasSportsFacility()
        );
    }
    else if (original instanceof PG) {
        PG pg = (PG)original;
        return new PG(
            field.equals("name") ? (String)value : pg.getName(),
            field.equals("location") ? (String)value : pg.getLocation(),
            field.equals("rent") ? (Integer)value : pg.getRent(),
            pg.hasWifi(), pg.hasMeals(), pg.hasSecurity(),
            field.equals("rating") ? (Double)value : pg.getRating(),
            pg.getRoomSharing(), pg.hasLaundry(), pg.hasMess(),
            pg.hasAC(), pg.hasGeyser(), pg.hasKitchen(),
            pg.getCurfew(), pg.getGender(),
            pg.hasParking(), pg.hasPowerBackup()
        );
    }
    else { // Flat
```

```
        Flat f = (Flat)original;
        return new Flat(
            field.equals("name") ? (String)value : f.getName(),
            field.equals("location") ? (String)value : f.getLocation(),
            field.equals("rent") ? (Integer)value : f.getRent(),
            f.hasWifi(), f.hasMeals(), f.hasSecurity(),
            field.equals("rating") ? (Double)value : f.getRating(),
            f.hasLaundry(), f.hasMess(), f.hasAC(),
            f.hasGeyser(), f.hasKitchen(), f.getGender(),
            f.isFurnished(), f.isMaintenanceIncluded()
        );
    }
}

private void updateFacilities(Integer index) {
    Accommodation acc = accommodations.get(index);

    JFrame facilitiesFrame = new JFrame("Update Facilities");
    facilitiesFrame.setSize(300, 300);
    facilitiesFrame.setLayout(null);

    JLabel titleLabel = new JLabel("Update Facilities for " + acc.getName());
    titleLabel.setBounds(50, 20, 250, 25);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 14));
    facilitiesFrame.add(titleLabel);

    JCheckBox wifiCheck = new JCheckBox("WiFi", acc.hasWifi());
    wifiCheck.setBounds(50, 60, 100, 25);
    facilitiesFrame.add(wifiCheck);

    JCheckBox mealsCheck = new JCheckBox("Meals", acc.hasMeals());
    mealsCheck.setBounds(50, 90, 100, 25);
    facilitiesFrame.add(mealsCheck);

    JCheckBox securityCheck = new JCheckBox("Security", acc.hasSecurity());
    securityCheck.setBounds(50, 120, 100, 25);
    facilitiesFrame.add(securityCheck);

    JCheckBox laundryCheck = new JCheckBox("Laundry", acc.hasLaundry());
    laundryCheck.setBounds(50, 150, 100, 25);
    facilitiesFrame.add(laundryCheck);

    JCheckBox acCheck = new JCheckBox("AC", acc.hasAC());
    acCheck.setBounds(50, 180, 100, 25);
    facilitiesFrame.add(acCheck);

    JCheckBox geyserCheck = new JCheckBox("Geyser", acc.hasGeyser());
    geyserCheck.setBounds(50, 210, 100, 25);
    facilitiesFrame.add(geyserCheck);

    JCheckBox kitchenCheck = new JCheckBox("Kitchen", acc.hasKitchen());
    kitchenCheck.setBounds(50, 240, 100, 25);
    facilitiesFrame.add(kitchenCheck);
```

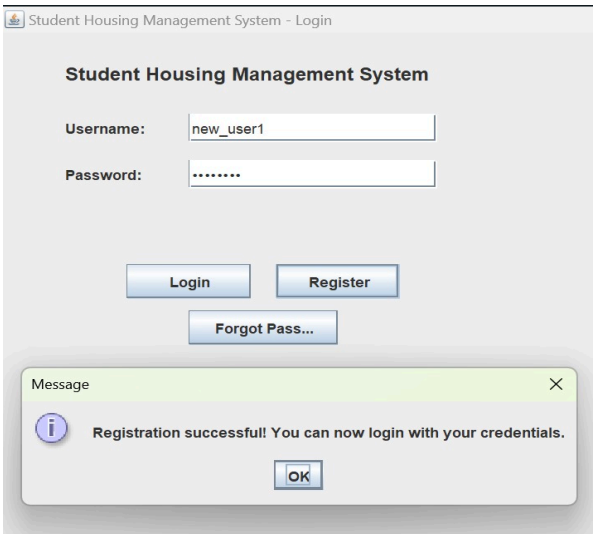
```
    JButton updateButton = new JButton("Update");
    updateButton.setBounds(50, 280, 100, 30);
    updateButton.addActionListener(e -> {
        accommodations.set(index, toggleFacility(acc,
            wifiCheck.isSelected(), mealsCheck.isSelected(),
            securityCheck.isSelected(), laundryCheck.isSelected(),
            acCheck.isSelected(), geyserCheck.isSelected(),
            kitchenCheck.isSelected()));
        updateAccommodationInDatabase(accommodations.get(index));
        JOptionPane.showMessageDialog(facilitiesFrame, "Facilities updated successfully!");
        facilitiesFrame.dispose();
    });
    facilitiesFrame.add(updateButton);

    JButton cancelButton = new JButton("Cancel");
    cancelButton.setBounds(160, 280, 100, 30);
    cancelButton.addActionListener(e -> facilitiesFrame.dispose());
    facilitiesFrame.add(cancelButton);

    facilitiesFrame.setVisible(true);
}

private Accommodation toggleFacility(Accommodation original,
                                     boolean wifi, boolean meals, boolean security,
                                     boolean laundry, boolean ac, boolean geyser,
                                     boolean kitchen) {
    if (original instanceof Hostel) {
        Hostel h = (Hostel)original;
        return new Hostel(
            h.getName(), h.getLocation(), h.getRent(),
            wifi, meals, security,
            h.getRating(), h.getRoomSharing(),
            laundry, h.hasMess(), ac, geyser, kitchen,
            h.getCurfew(), h.getGender(),
            h.hasStudyRoom(), h.hasSportsFacility()
        );
    }
    else if (original instanceof PG) {
        PG pg = (PG)original;
        return new PG(
            pg.getName(), pg.getLocation(), pg.getRent(),
            wifi, meals, security,
            pg.getRating(), pg.getRoomSharing(),
            laundry, pg.hasMess(), ac, geyser, kitchen,
            pg.getCurfew(), pg.getGender(),
            pg.hasParking(), pg.hasPowerBackup()
        );
    }
    else { // Flat
        Flat f = (Flat)original;
        return new Flat(
            f.getName(), f.getLocation(), f.getRent(),
            wifi, meals, security,
            f.getRating(),
```

```
            laundry, f.hasMess(), ac, geyser, kitchen,
            f.getGender(),
            f.isFurnished(), f.isMaintenanceIncluded()
        );
    }
}

private void viewAllBookings() {
    outputArea.setText(""); // Clear the output area
    outputArea.append("=== All Bookings ===\n");

    if (bookings.isEmpty()) {
        outputArea.append("No bookings found.");
    } else {
        for (Booking booking : bookings) {
            outputArea.append("\nBooking ID: " + booking.getId() + "\n");
            outputArea.append("Customer: " + booking.getCustomerName() + "\n");
            outputArea.append("Contact: " + booking.getContact() + "\n");
            outputArea.append("University: " + booking.getUniversity() + "\n");
            outputArea.append("Status: " + (booking.isConfirmed() ? "Confirmed" : "Pending") + "\n");
            outputArea.append("Accommodation: " + booking.getAccommodation().getName() +
                    " (" + booking.getAccommodation().getType() + ")\n");
        }
    }
}

private void confirmBooking() {
    if (bookings.isEmpty()) {
        JOptionPane.showMessageDialog(adminFrame, "No bookings found.");
        return;
    }

    String[] bookingIds = new String[bookings.size()];
    for (int i = 0; i < bookings.size(); i++) {
        bookingIds[i] = "ID: " + bookings.get(i).getId() + " - " +
                bookings.get(i).getCustomerName() + " (" +
                bookings.get(i).getAccommodation().getName() + ")";
    }

    String selected = (String) JOptionPane.showInputDialog(
        adminFrame,
        "Select booking to confirm:",
        "Confirm Booking",
        JOptionPane.QUESTION_MESSAGE,
        null,
        bookingIds,
        bookingIds[0]);

    if (selected == null) return;

    int index = -1;
    for (int i = 0; i < bookingIds.length; i++) {
        if (bookingIds[i].equals(selected)) {
            index = i;
```

```
                break;
            }
        }

        if (index == -1) return;

        Booking booking = bookings.get(index);
        booking.setConfirmed(true);
        updateBookingConfirmation(booking);

        JOptionPane.showMessageDialog(adminFrame,
            "Booking ID " + booking.getId() + " confirmed successfully!");
    }

    private void updateBookingConfirmation(Booking booking) {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement stmt = conn.prepareStatement(
                 "UPDATE bookings SET confirmed = ? WHERE id = ?")) {

            stmt.setBoolean(1, booking.isConfirmed());
            stmt.setInt(2, booking.getId());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

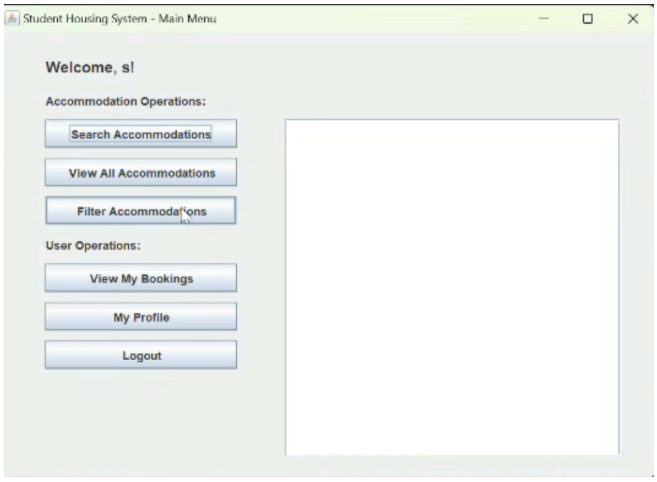
8. Working with screenshots

-> Login System



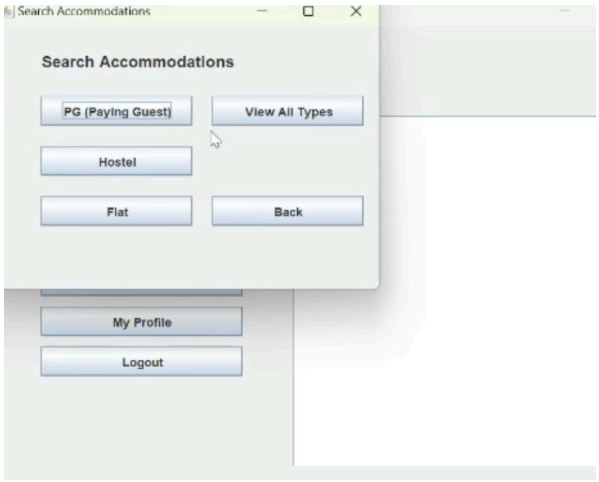
-> Main Interface

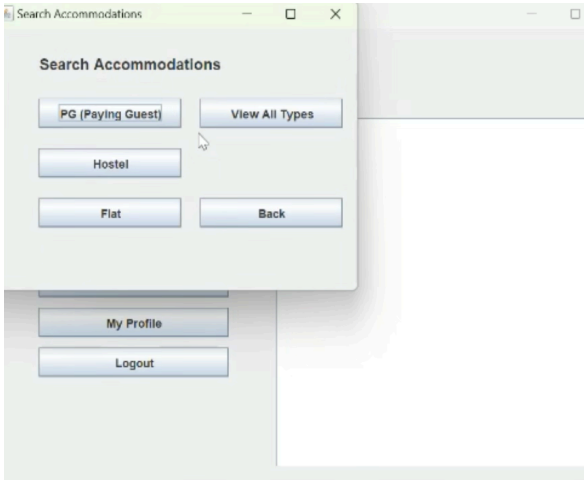
- To view all properties

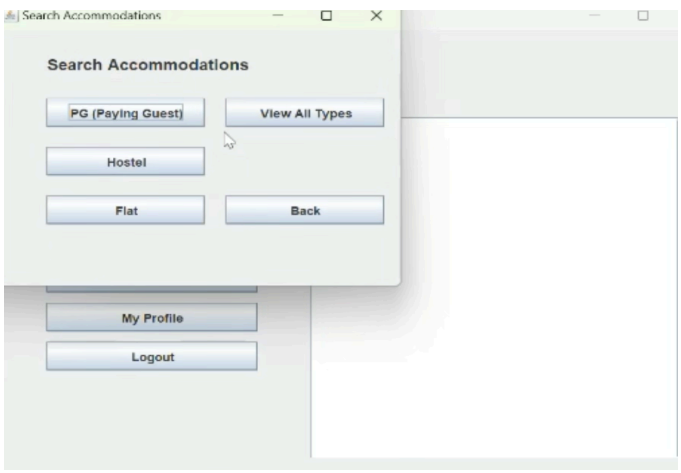


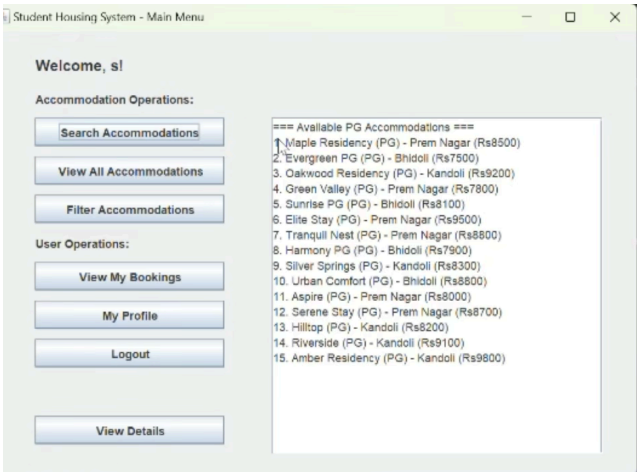
- Filter using search tools

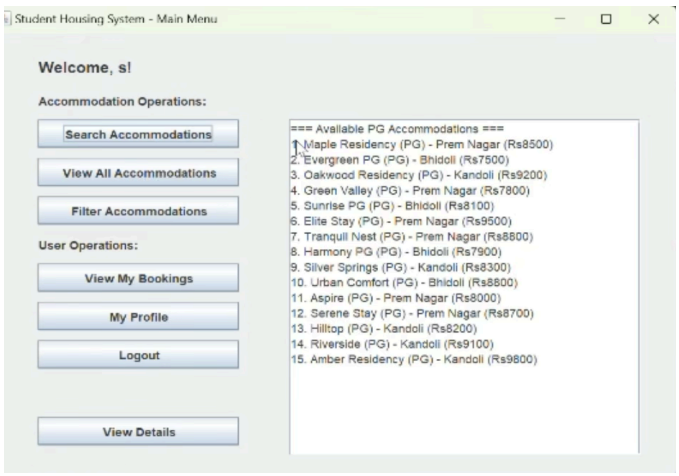
Filters based upon type of accommodation

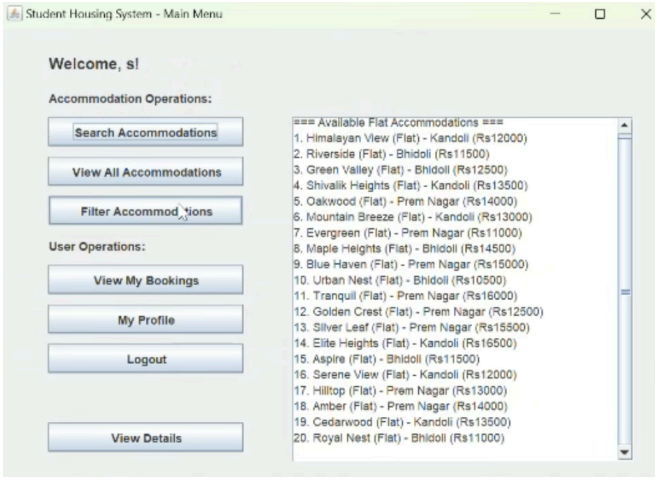




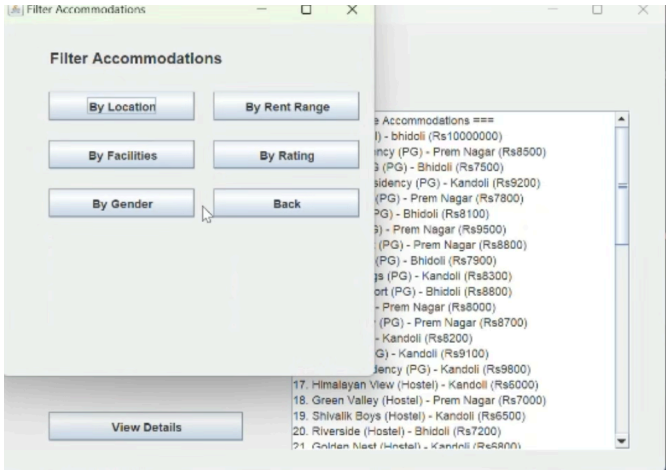


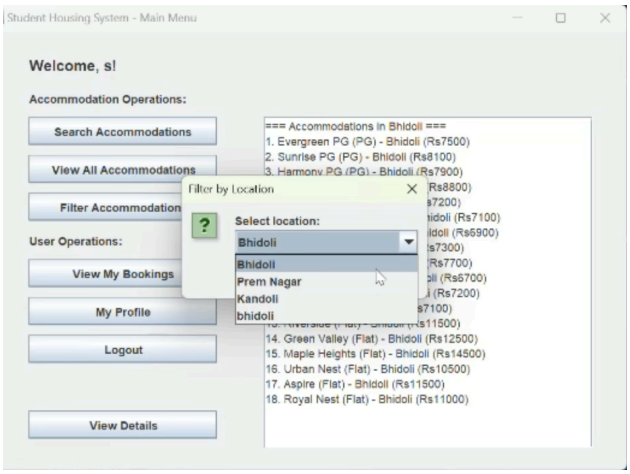


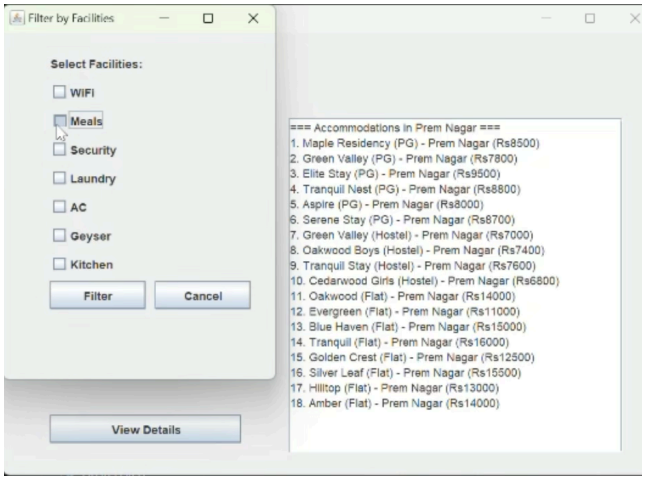


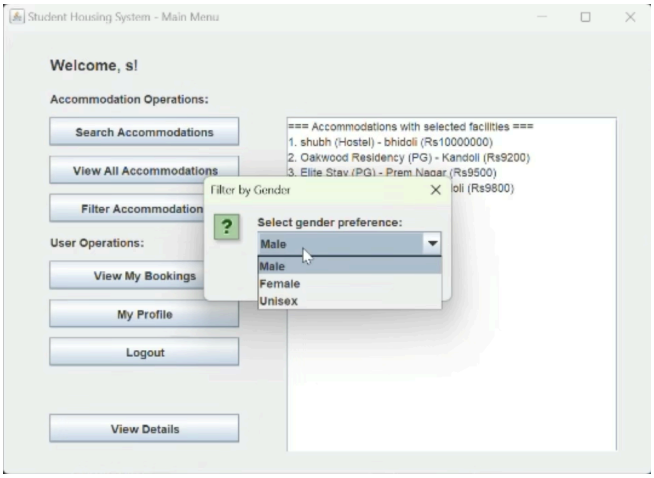


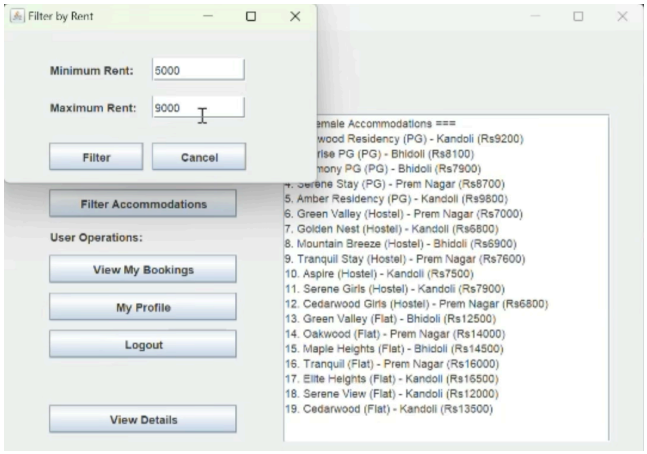
The screens below allow users to filter accommodations based on location, rent, facilities, rating, and gender preference. A Back button lets users return to the main menu**.**

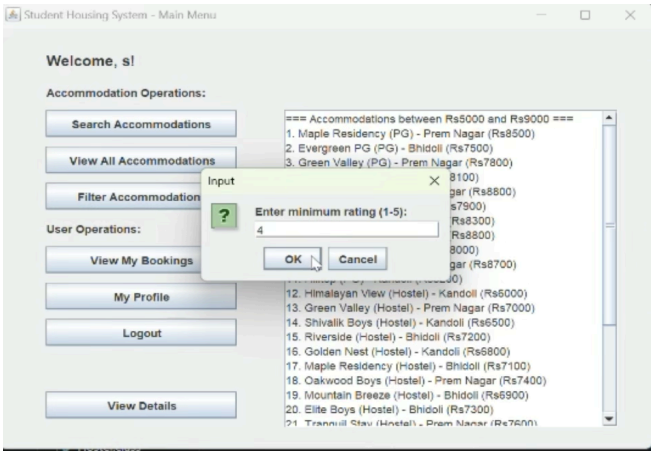




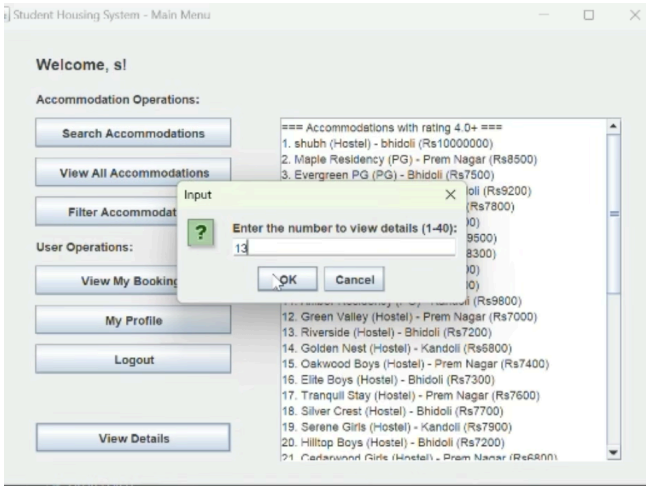






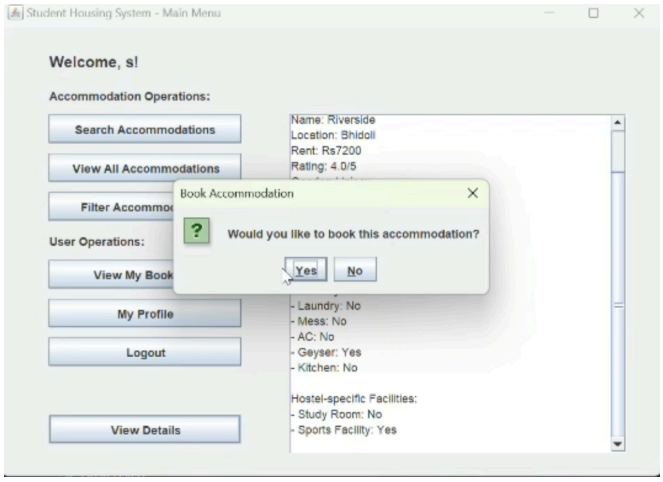


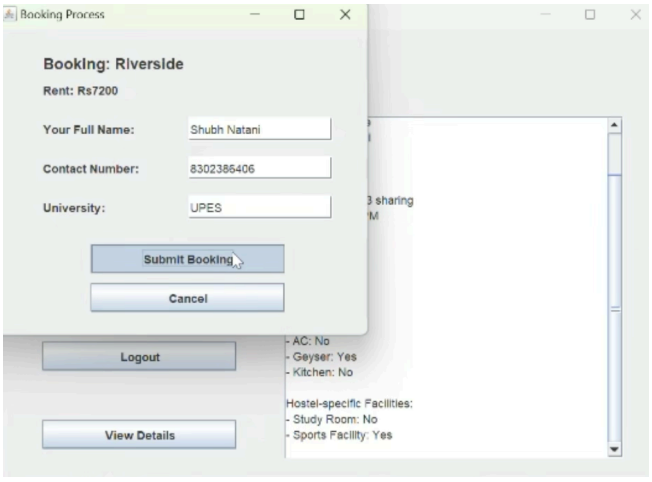
These screens below show the filtered accommodation lists based on selected criteria. Each filter helps the user quickly find suitable options.



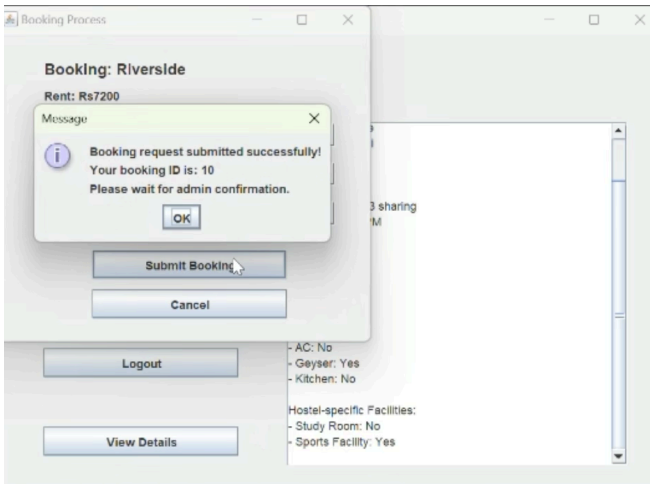
Booking Process :

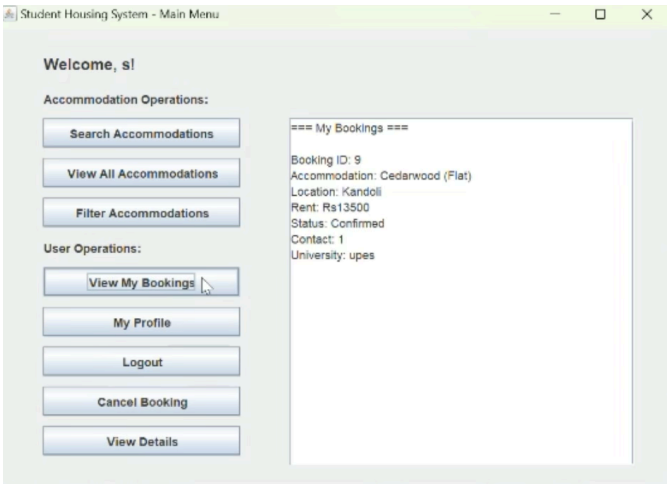
The user selects a hostel, PG, or flat, fills in personal details, and submits the form.





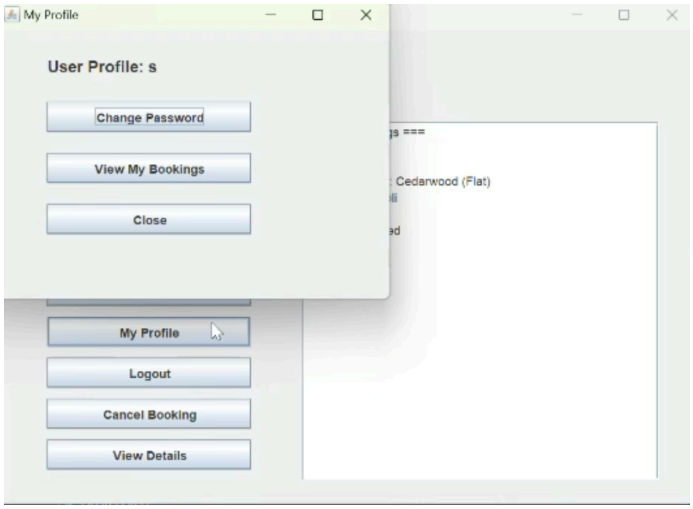
A confirmation message with a unique Booking ID is then shown. This ID can later be used to view the booking details as shown.

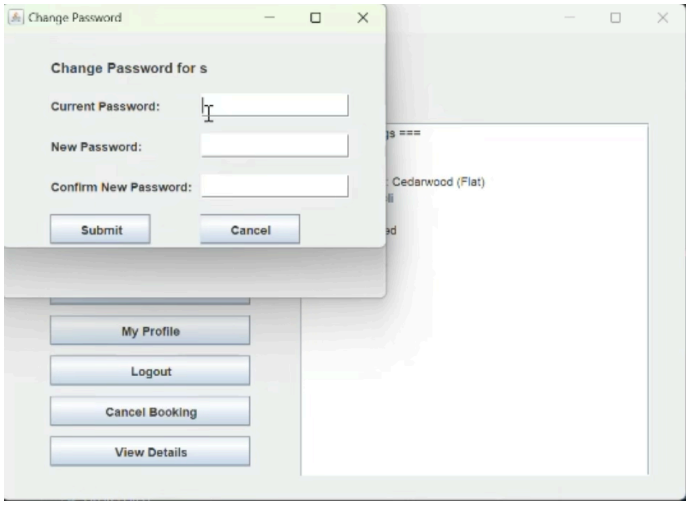




User Profile :

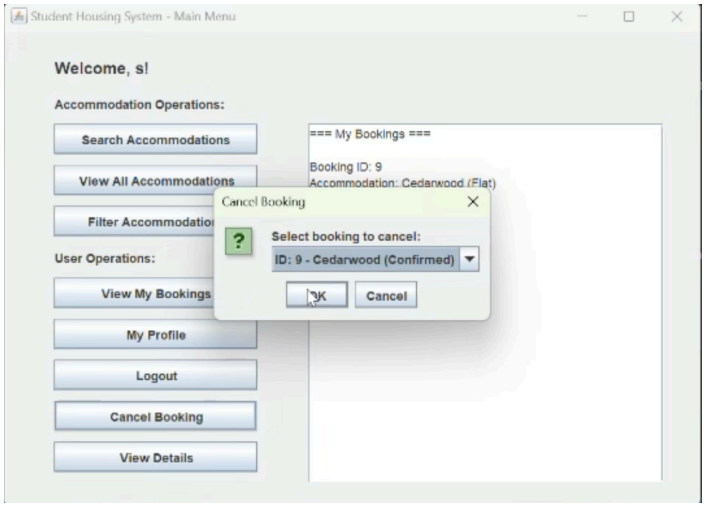
These screenshots show the user profile section, where users can update personal details, change their password, view bookings, or close the profile. One screenshot highlights the “Change Password” feature, allowing secure updates to login credentials.





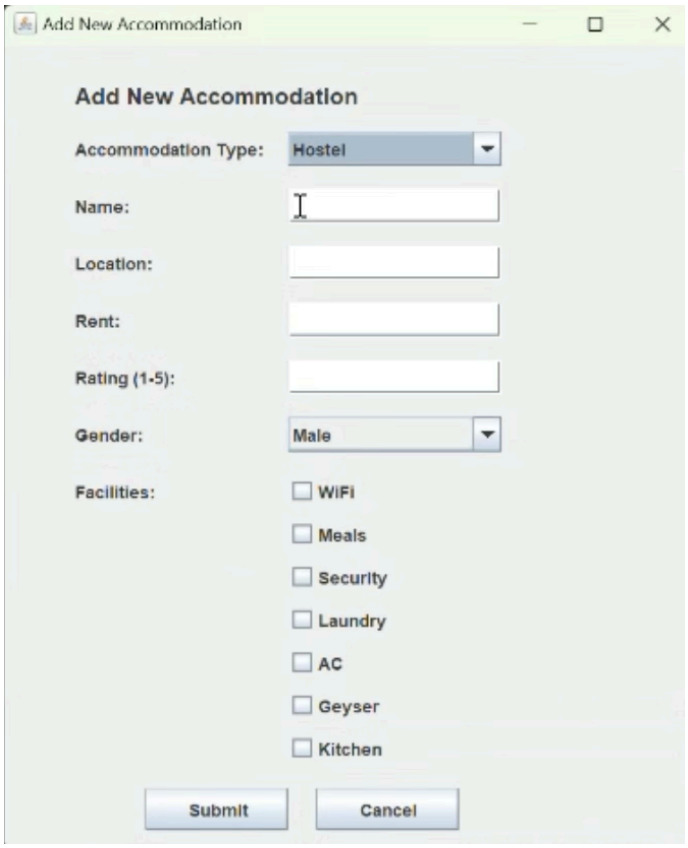
Cancel Booking :

This screenshot shows the option to cancel an existing booking. The user selects their booking and clicks “OK” to remove it from the system. A success message confirms the cancellation.



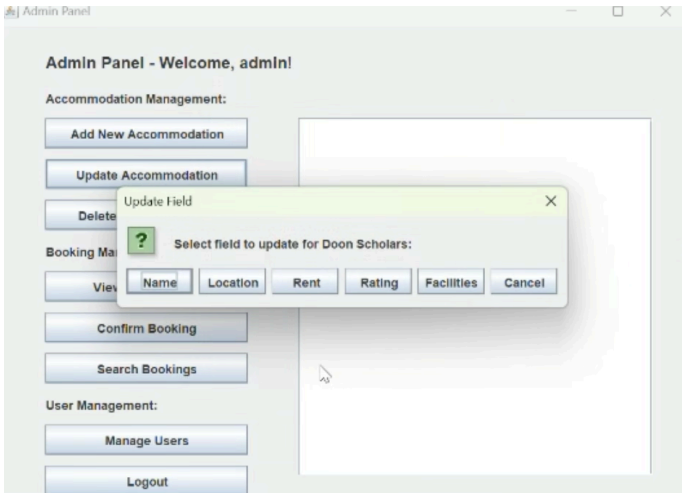
Admin Login :

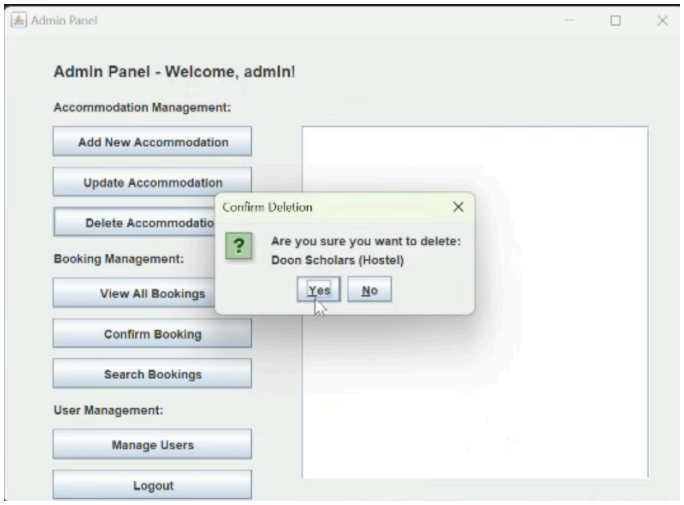
The below screenshots show the admin login and dashboard. After logging in, the admin can add new accommodations by entering details like name, location, rent, rating, gender preference, and facilities.

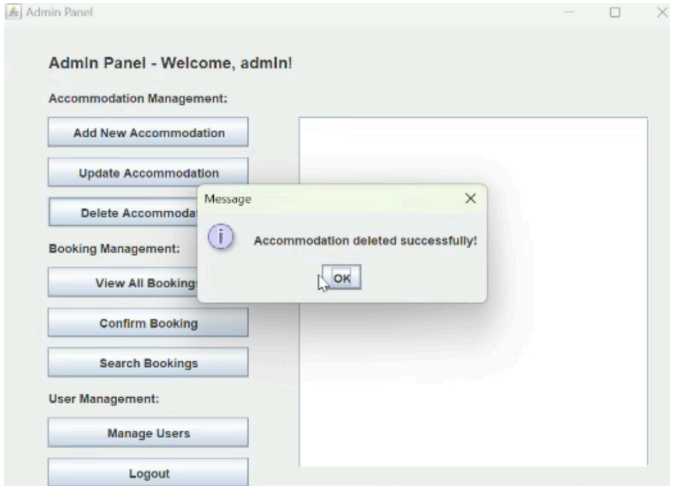


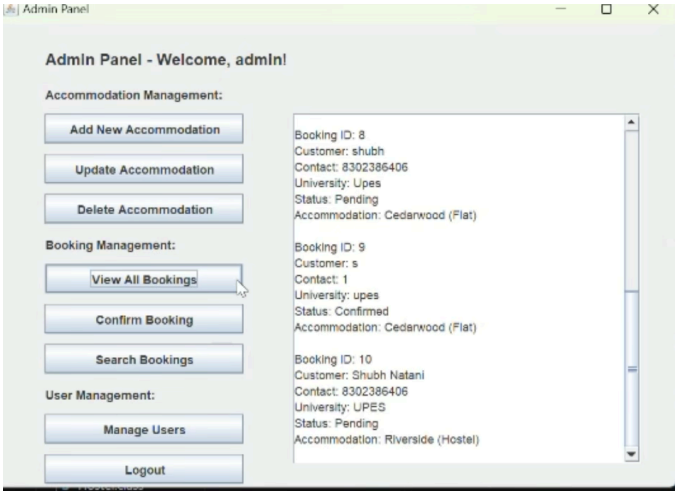
Admin Panel :

The below screenshots show the admin updating accommodation details by selecting a property and modifying fields like location, rent, rating, gender preference, or facilities to keep listings current.



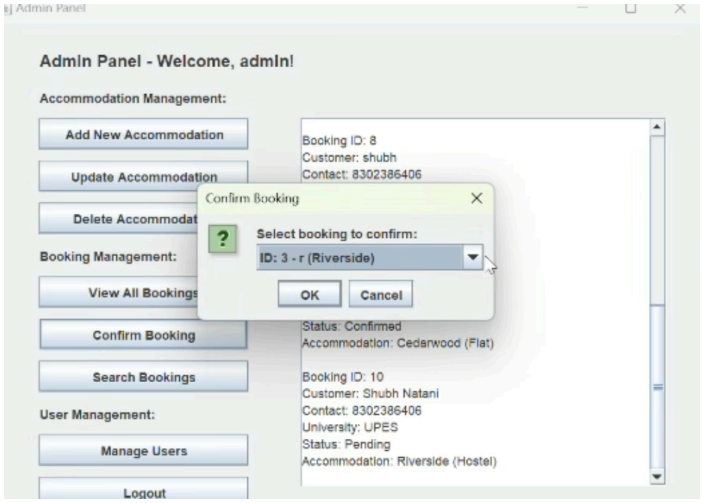


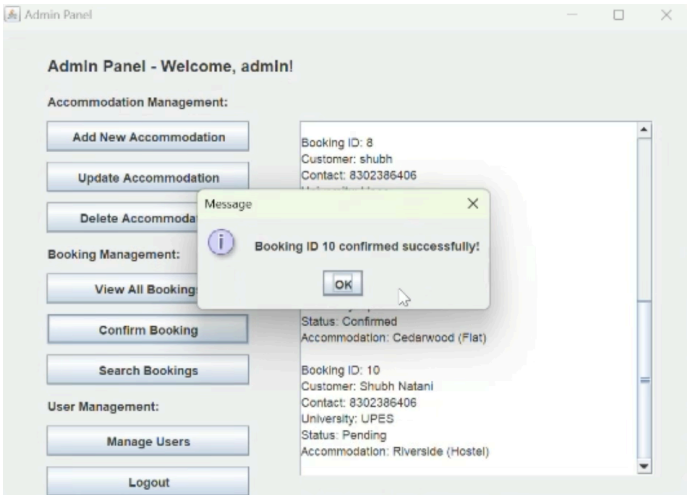




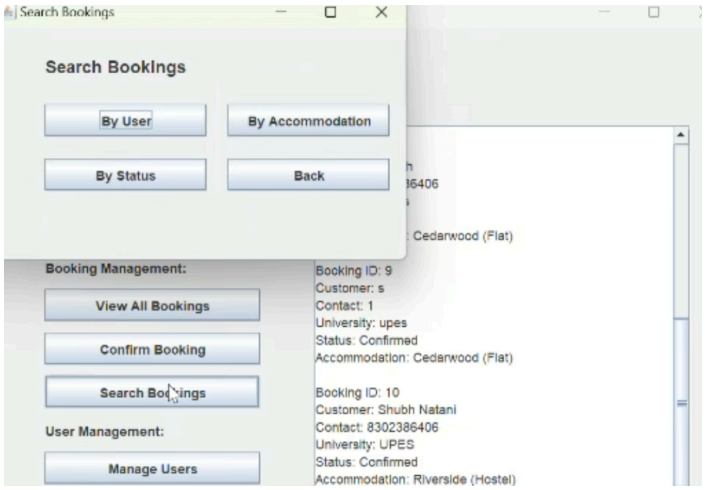
Admin Conforming Booking :

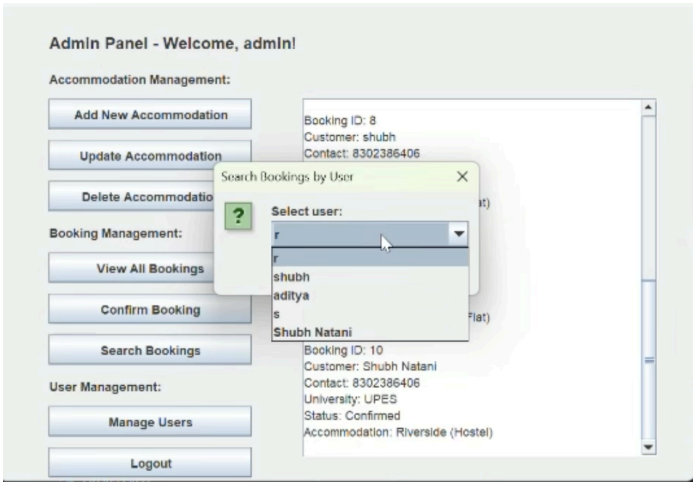
These screenshots show the admin confirming a booking. The admin selects a property and booking ID to verify the details, and upon confirmation, the booking is marked as confirmed in the system.

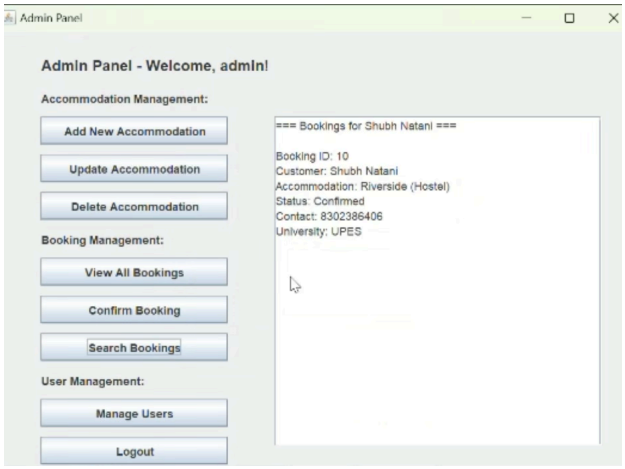


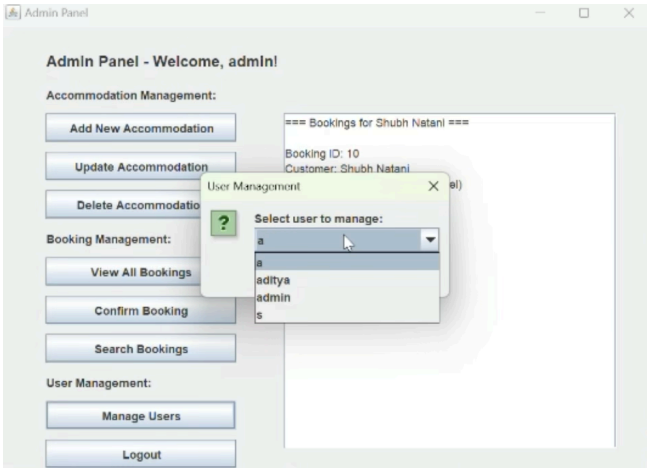


These screenshots show the admin searching bookings by user, accommodation, or status, allowing efficient filtering and management of reservations.









Password Recovery :

These screenshots show the admin managing user accounts by resetting passwords or sending reset links to users' registered emails for secure password recovery.