SIX WEEKS SUMMER TRAINING REPORT

on

HOSTEL ALLOTMENT MANAGEMENT SYSTEM

Submitted by

KOMAL KUMARI

Registration No: 12114903

Program Name: B.Tech.(CSE)

Under the Guidance of

Mr. Ravi Kant Sahu

Assistant Professor

School of Computer Science and Engineering
Lovely Professional University. Phagwara
(June- July, 2023)

DECLARATION

I hereby declare that I have completed my six weeks summer training at **Human Resource Development Center, Lovely Professional University, Phagwara (Punjab)** from **6th June 2023** to **17th July 2023** under the guidance of **Mr. Ravi Kant Sahu**. I declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of **B.Tech.(Computer Science & Engineering)** at Lovely Professional University, Phagwara.

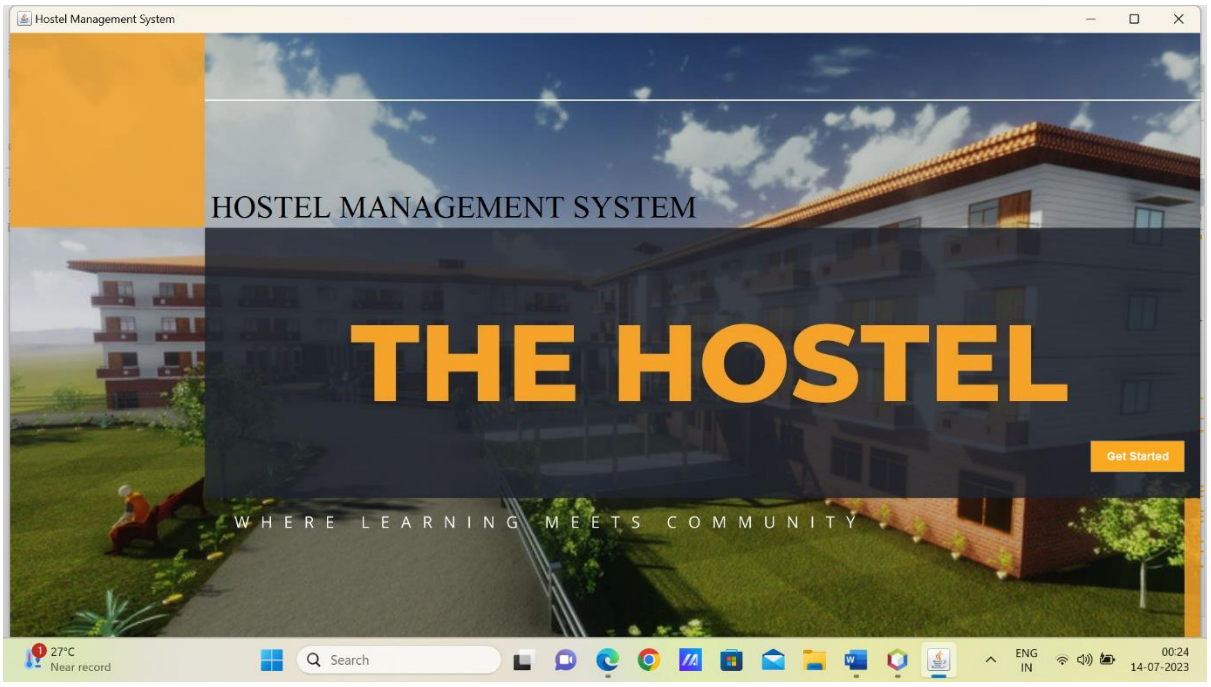
Komal Kumari

Registration No: 12114903

Welcome to the Hostel Allotment Management System!

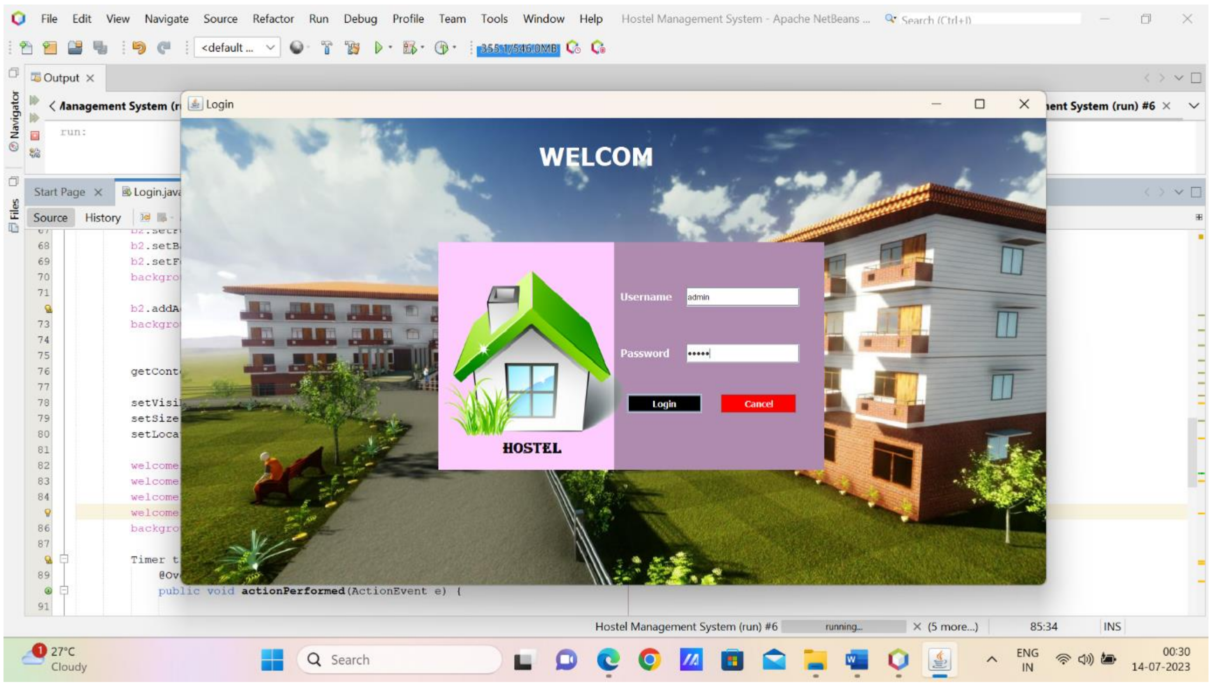
- The 'HostelManagementSystem' class is the main class representing the first page of your Hostel Management System.

  The main functionality of this page is to display an image, a title, and a "Get Started" button. When the button is clicked, it hides the current page and opens the login page. The code also includes a continuous blinking effect for the title by toggling its visibility every 500 milliseconds.



- The login page is an integral component of the Hostel Management System, serving as the initial point of entry for users to access the system. It plays a crucial role in ensuring the security and authentication of users before granting them access to the system's functionalities.

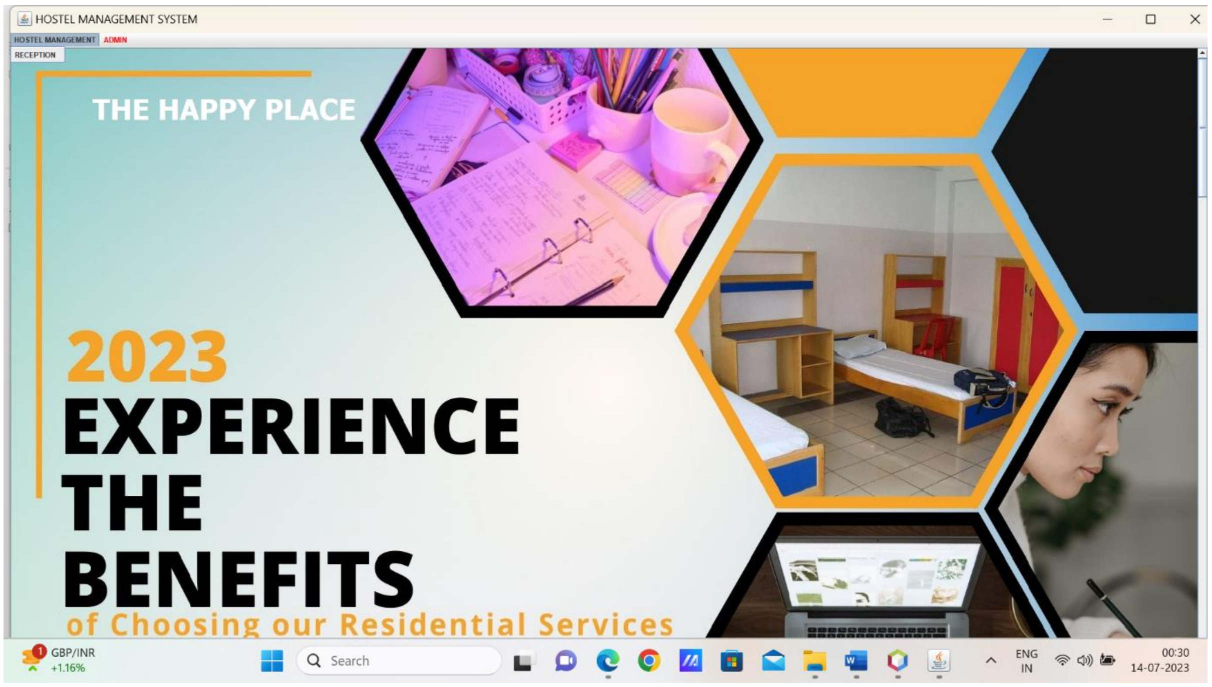
  When the user clicks the "Login" button, the entered username and password are sent to the system's database for authentication. If the credentials are valid, the system grants access to the user; otherwise, an error message is displayed, indicating an invalid login attempt.

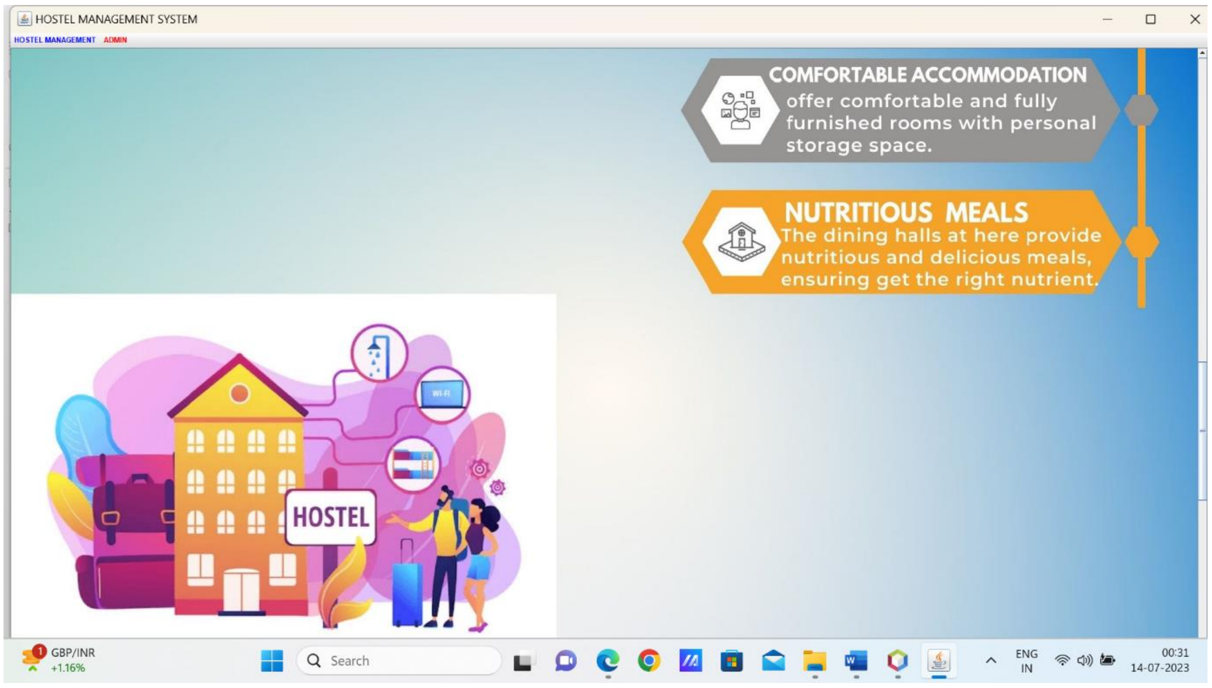


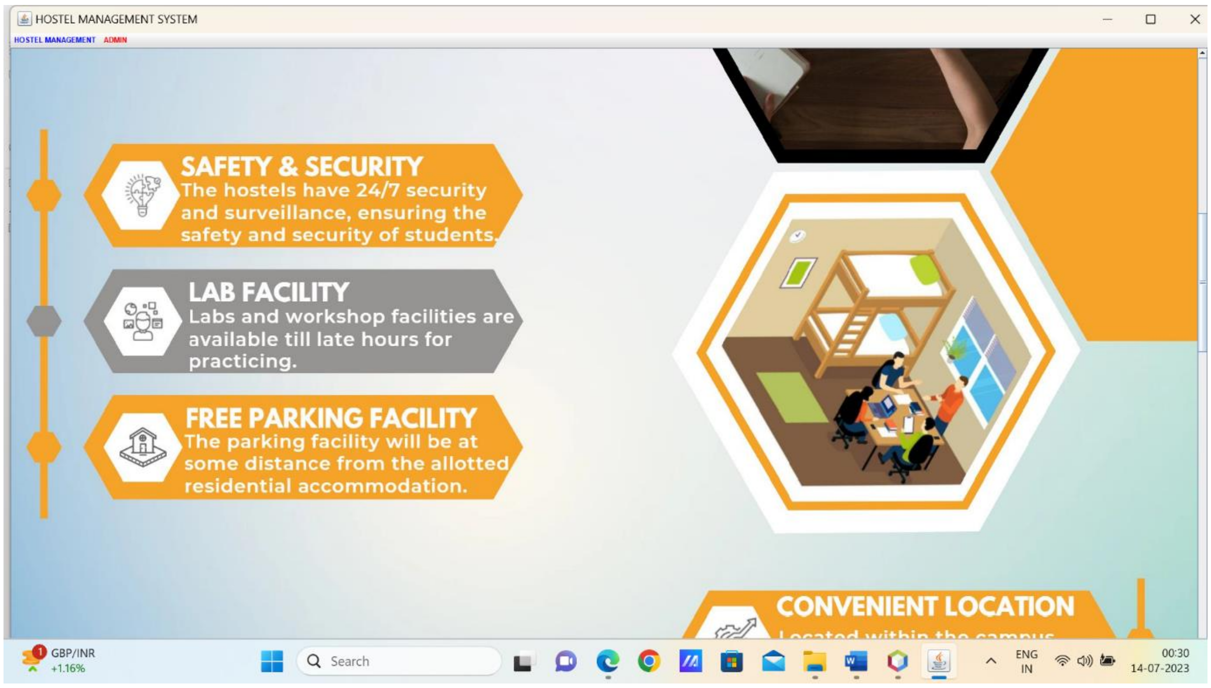
Username: admin
Password: •••••

- The Dashboard page of the Hostel Management System provides an overview of crucial information and serves as a central control panel for managing hostel operations. With its visually appealing design, interactive elements, and seamless navigation, it enables hostel staff to efficiently manage tasks and access relevant functionalities.

  The Dashboard page consists of multiple content panels, each displaying specific information or functionality related to the hostel management system. These panels may include sections for reception management, administrative tasks, room allocations, and more. This page offers a "Log Out" option in the "ADMIN" menu. When selected, it allows users to securely log out from the system, ensuring the privacy and security of their data.




HOSTEL MANAGEMENT SYSTEM
HOSTEL MANAGEMENT
ADMIN
COMFORTABLE ACCOMMODATION
offer comfortable and fully furnished rooms with personal storage space.
NUTRITIOUS MEALS
The dining halls at here provide nutritious and delicious meals, ensuring get the right nutrient.
HOSTEL
Search
00:31
14-07-2023

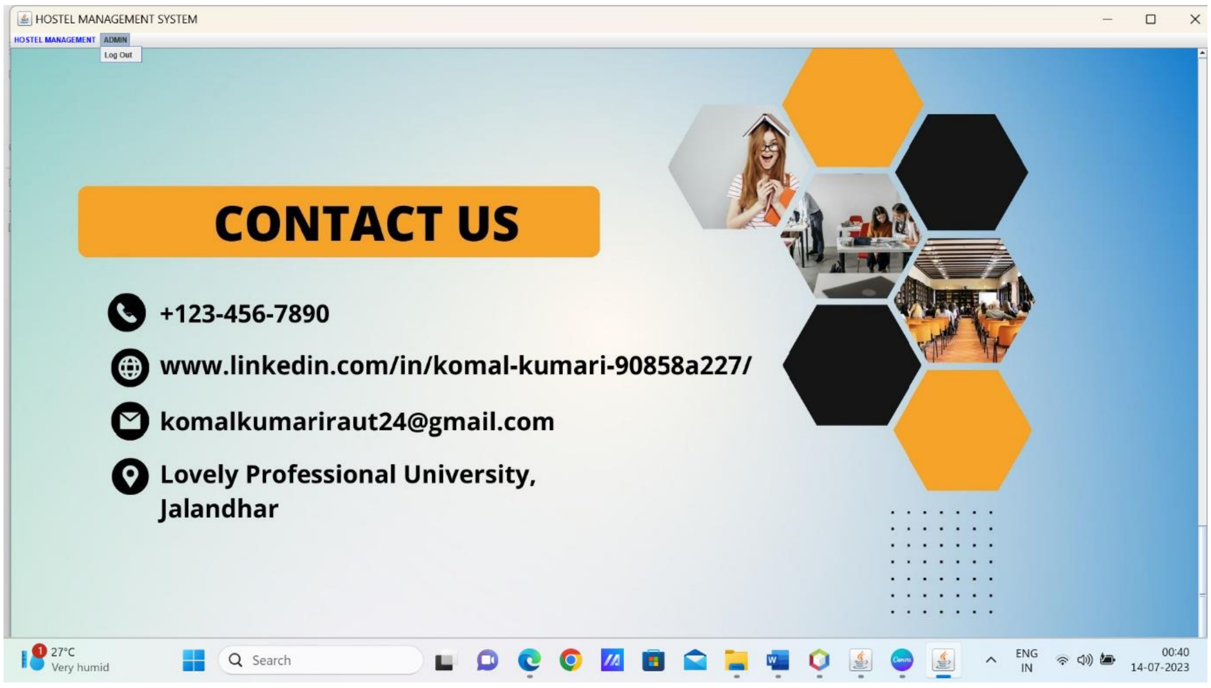


SAFETY & SECURITY
The hostels have 24/7 security and surveillance, ensuring the safety and security of students.

LAB FACILITY
Labs and workshop facilities are available till late hours for practicing.

FREE PARKING FACILITY
The parking facility will be at some distance from the allotted residential accommodation.

CONVENIENT LOCATION



CONTACT US

+123-456-7890

www.linkedin.com/in/komal-kumari-90858a227/

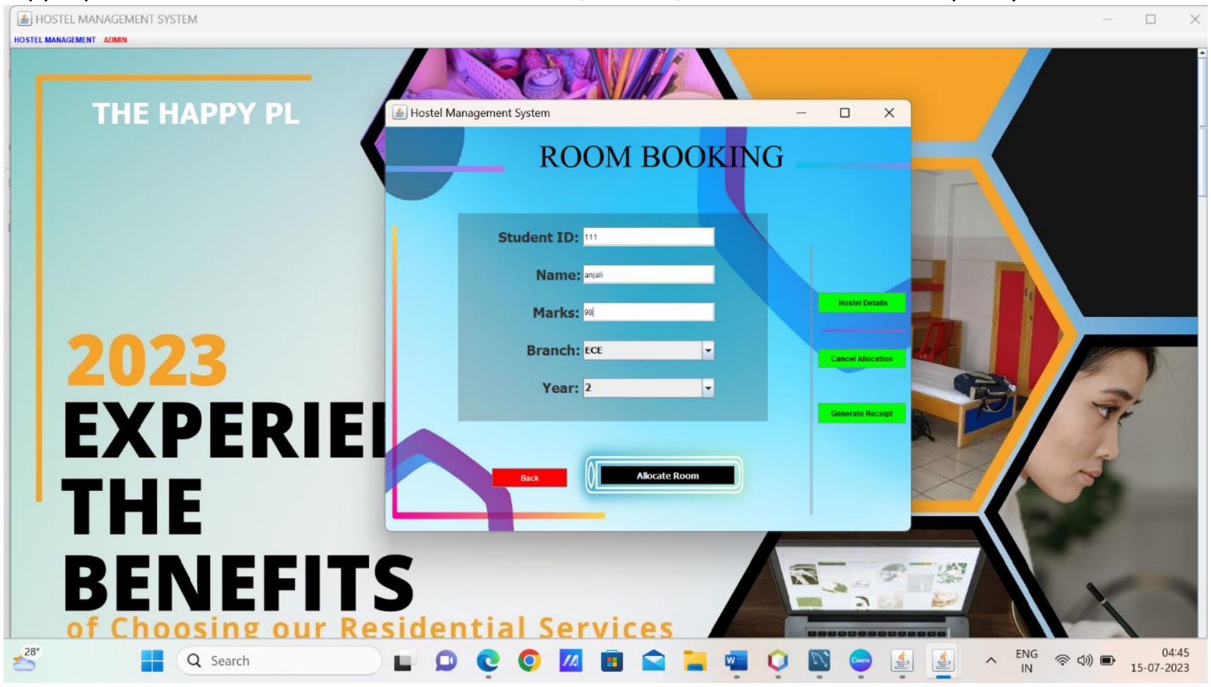
komalkumariraut24@gmail.com

Lovely Professional University, Jalandhar

- **Room Allocation GUI Page of the Hostel Management System!**

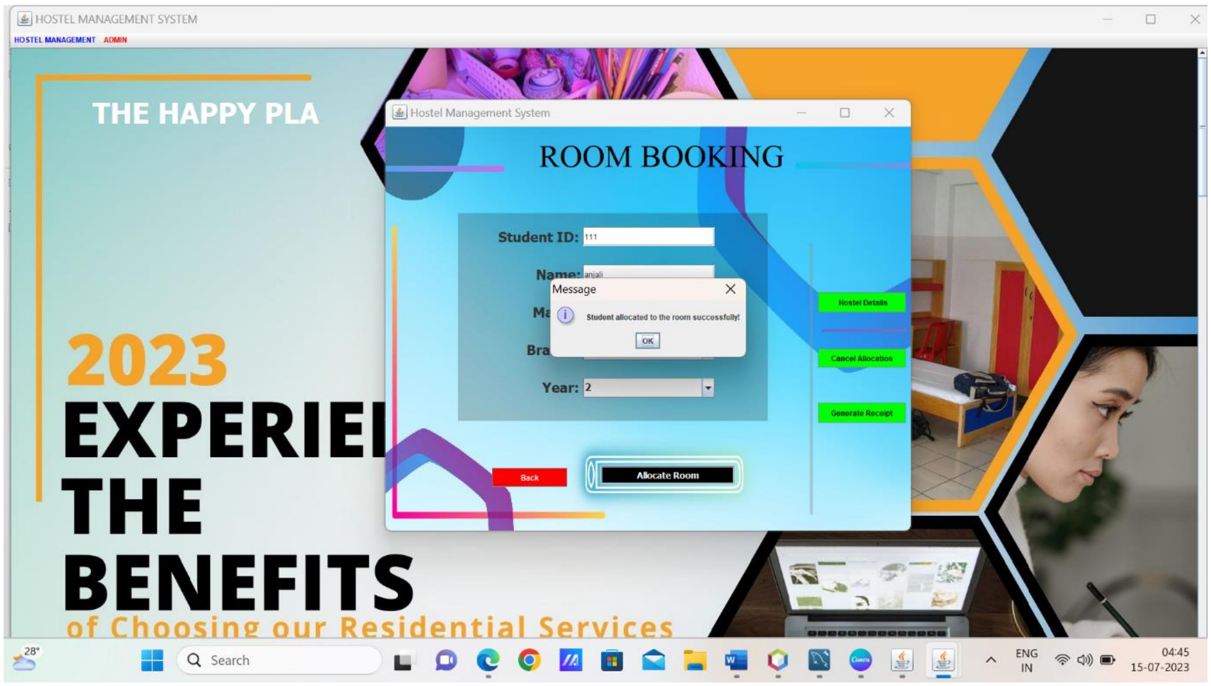
  The Room Allocation GUI page allows hostel administrators to allocate rooms to students based on their information. The page includes input fields and labels to capture student details. These fields typically include the student's name, marks, branch, year, and student ID.

  The "**Allocate Room**" button triggers the room allocation process based on the student's information provided in the input fields. When clicked, the system checks the student's eligibility and assigns an appropriate room based on the student's marks, branch, and available room capacity.



Student ID: 111
Name: anjali
Marks: 98
Branch: ECE
Year: 2

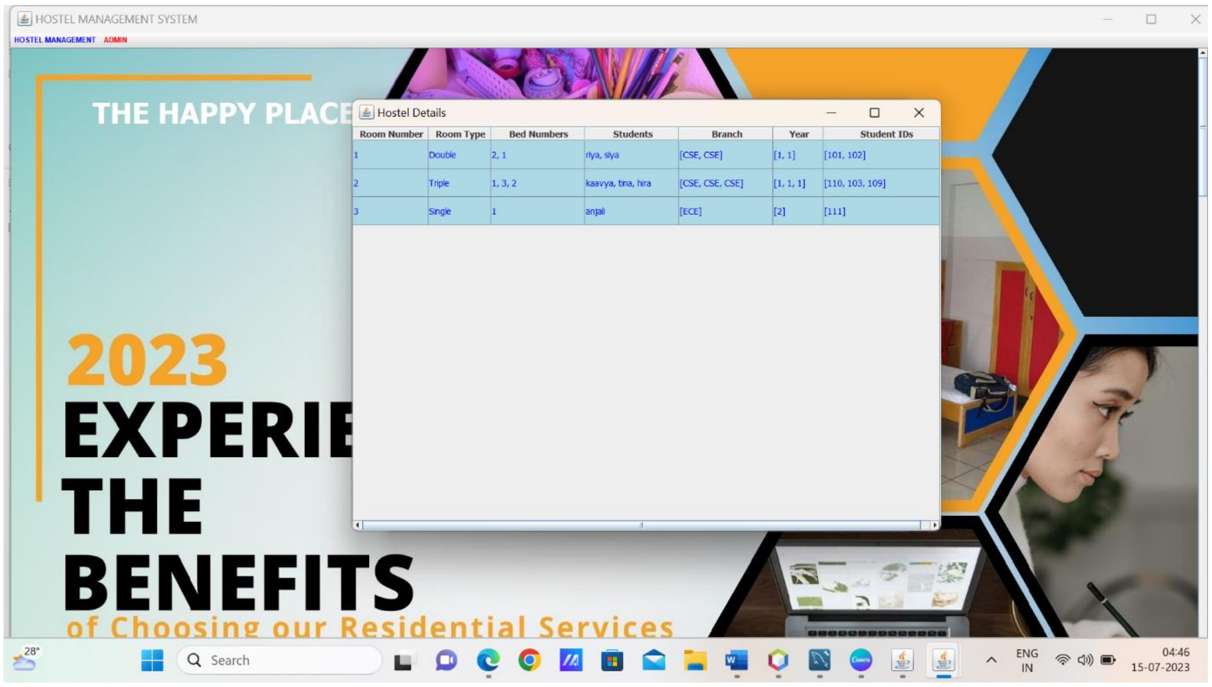
Generate Receipt Button: The "Generate Receipt" button allows administrators to generate a receipt for the allocated room. Clicking this button opens a receipt generation page, where administrators can enter additional details and generate a receipt for the allocated room.



Student ID: 111
Name: anjali
Year: 2

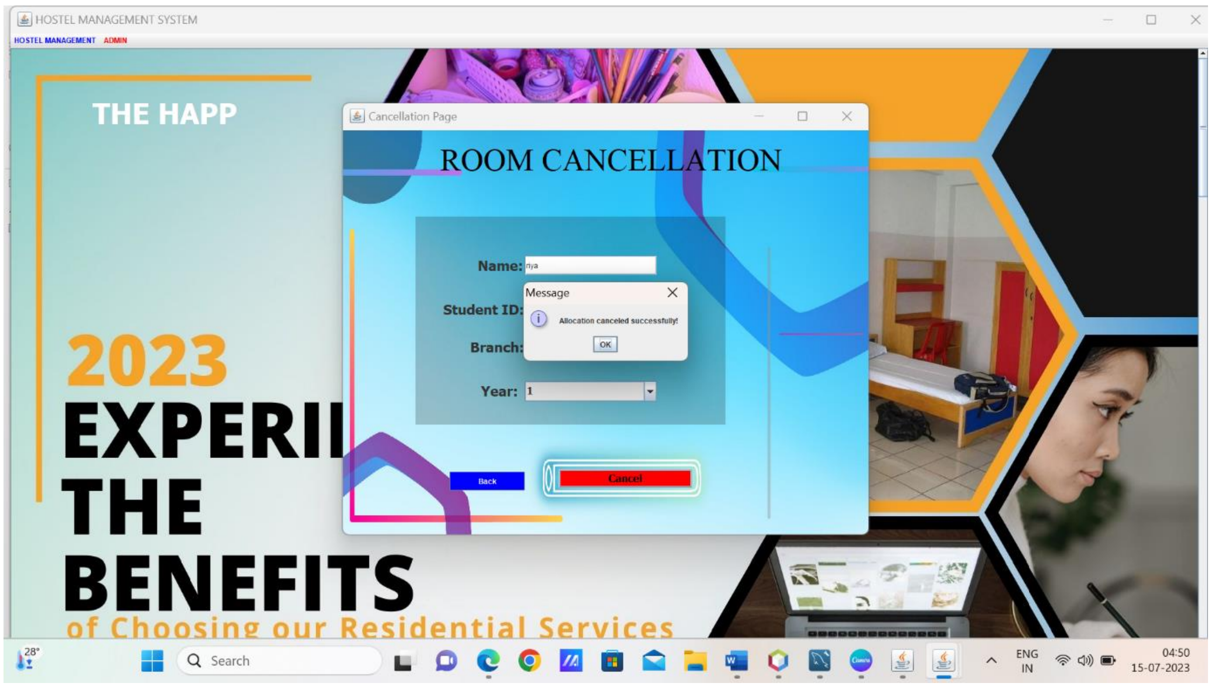
Student allocated to the room successfully!

Hostel Details Button: The "Hostel Details" button provides access to additional information and details about the hostel. Clicking this button opens a page with relevant hostel information, such as available rooms, room types, facilities, and more.



Room Number	Room Type	Bed Numbers	Students	Branch	Year	Student IDs
1	Double	2, 1	riya, siya	[CSE, CSE]	[1, 1]	[101, 102]
2	Triple	1, 3, 2	kaavya, tina, hira	[CSE, CSE, CSE]	[1, 1, 1]	[110, 103, 109]
3	Single	1	anjali	[ECE]	[2]	[111]

Cancel Allocation Button: The "Cancel Allocation" button enables administrators to cancel room allocations if needed. Clicking this button opens a cancellation page, where administrators can select the allocated room and cancel the allocation for a specific student.



Name: riya

Student ID:

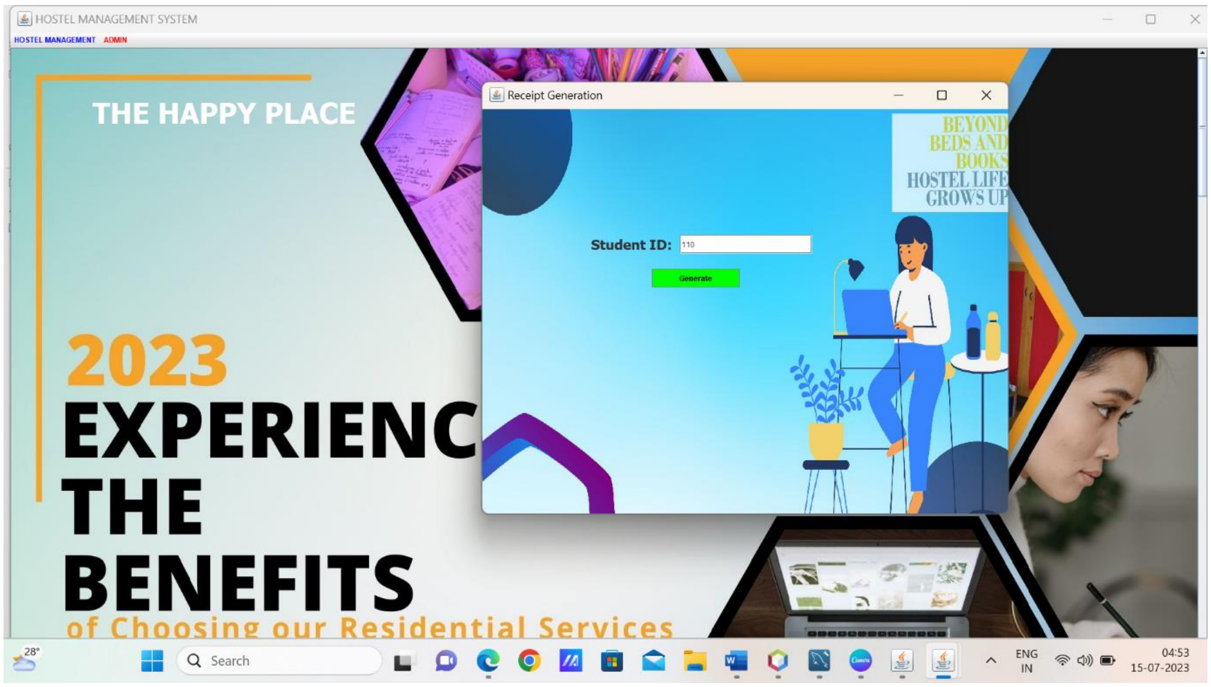
Branch:

Year: 1

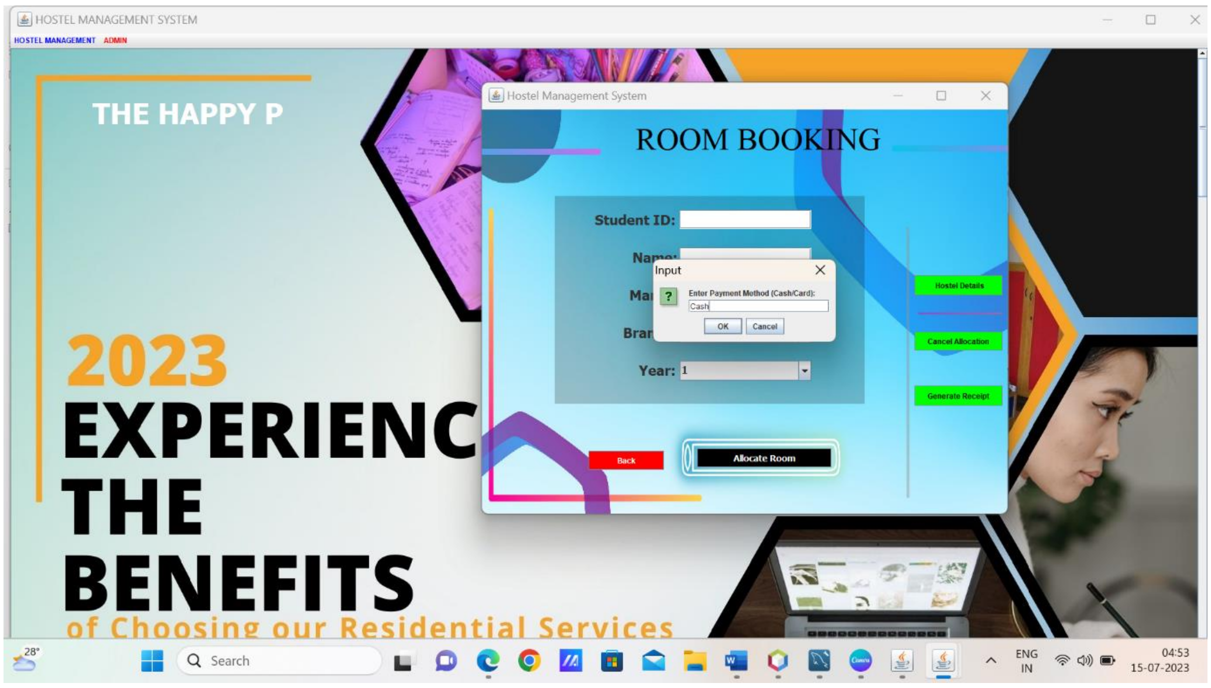
Message

Allocation canceled successfully!

- **This Frame allows** the generation of a receipt based on the student ID, including the student's information, room details, payment method, total amount, and a randomly generated receipt number.



Student ID: 110



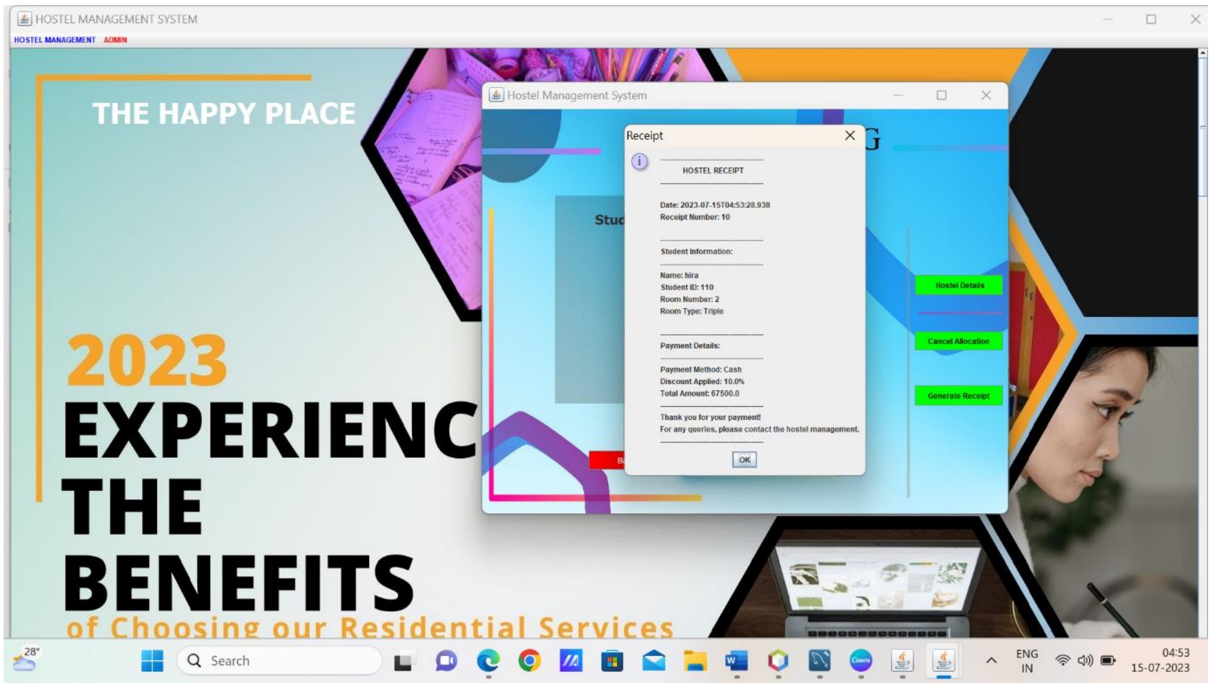
Student ID:

Name:

Enter Payment Method (Cash/Card): Cash

Year: 1

- By incorporating the **discount logic**, the hostel management system offers discounts to students based on their marks and the chosen room type. The discount percentage is calculated and applied to the room price, resulting in a discounted price for the student. This ensures fair pricing and encourages academic achievements among students.



```
-------------------------------------
          HOSTEL RECEIPT
-------------------------------------

Date: 2023-07-15T04:53:28.938
Receipt Number: 10

-------------------------------------
Student Information:
-------------------------------------
Name: hira
Student ID: 110
Room Number: 2
Room Type: Triple

-------------------------------------
Payment Details:
-------------------------------------
Payment Method: Cash
Discount Applied: 10.0%
Total Amount: 67500.0
-------------------------------------
Thank you for your payment!
For any queries, please contact the hostel management.
-------------------------------------
```

```java
package hostel.management.system;


import java.awt.*;

import javax.swing.*;

import java.awt.event.*;


public class HostelManagementSystem extends JFrame implements ActionListener {


  public HostelManagementSystem() {

    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

    int screenWidth = (int) screenSize.getWidth();

    int screenHeight = (int) screenSize.getHeight();


    setSize(screenWidth, screenHeight);

    setLocation(0, 0);
```

```
        setLayout(null);

        setTitle("Hostel Management System");
        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/1.1.jpg"));
        JLabel image = new JLabel(i1);
        image.setBounds(0, 0, screenWidth, screenHeight);
        add(image);

        JLabel text = new JLabel("HOSTEL MANAGEMENT SYSTEM");
        text.setBounds(320, 230, 1000, 90);
        text.setForeground(Color.black);
        text.setFont(new Font("Serif", Font.PLAIN, 50));
        image.add(text);



       JButton next = new JButton("Get Started");
       next.setBounds(screenWidth - 197, 650, 150, 50);
       next.setBackground(new Color(249, 170, 33));
       next.setForeground(Color.WHITE);
       next.setFocusPainted(false);
       next.addActionListener(this);
       next.setFont(new Font("Arial", Font.BOLD, 18));

       next.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        next.setBackground(new Color(102, 255, 102));
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        next.setBackground(new Color(249, 170, 33));
    }
});
```

```
image.add(next);



        setVisible(true);


        while (true) {
            text.setVisible(false);
            try {
                Thread.sleep(500);
            } catch (Exception e) {
                e.printStackTrace();
            }
            text.setVisible(true);
            try {
                Thread.sleep(500);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public void actionPerformed(ActionEvent ae) {
        setVisible(false);
        new Login();
    }

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

```
package hostel.management.system;


import java.awt.*;

import javax.swing.*;

import java.awt.event.*;

import java.sql.*;


public class Login extends JFrame implements ActionListener{


    JLabel l1,l2, background,welcomeLabel;

    JTextField t1;

    JPasswordField t2;

    JButton b1,b2;

      int welcomeIndex = 0;


    Login(){


        super("Login");


        setLayout(null);


        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/login
background.PNG"));

        Image i2 = i1.getImage().getScaledInstance(1400,800,Image.SCALE_DEFAULT);

        ImageIcon i3 =  new ImageIcon(i2);

        background = new JLabel(i3);

        background.setBounds(0,0,1400,800);

        add(background);


        JPanel login=new JPanel();
```

```
login.setSize(200,300);

login.setBackground(new Color(0,0,0,80));

login.setBounds(690,198,335,363);


l1 = new JLabel("Username");

l1.setForeground(Color.WHITE);

l1.setFont(new Font("Tahoma", Font.BOLD, 16));

l1.setBounds(700,270,100,30);

background.add(l1);


l2 = new JLabel("Password");

l2.setForeground(Color.WHITE);

l2.setFont(new Font("Tahoma", Font.BOLD, 16));

l2.setBounds(700,360,100,30);

background.add(l2);


t1=new JTextField();

t1.setBounds(805,270,180,30);

background.add(t1);


t2=new JPasswordField();

t2.setBounds(805,360,180,30);

background.add(t2);




b1 = new JButton("Login");

b1.setBounds(710,440,120,30);

b1.setFont(new Font("Tahoma", Font.BOLD, 14));

b1.addActionListener(this);

b1.setBackground(Color.BLACK);

b1.setForeground(Color.WHITE);
```

```
background.add(b1);



b2=new JButton("Cancel");
b2.setBounds(860,440,120,30);
b2.setFont(new Font("Tahoma", Font.BOLD, 14));
b2.setBackground(Color.RED);
b2.setForeground(Color.WHITE);
background.add(b2);

b2.addActionListener(this);
background.add(login);



getContentPane().setBackground(Color.WHITE);

setVisible(true);
setSize(1400,800);
setLocation(260,170);

welcomeLabel = new JLabel("WELCOME");
welcomeLabel.setForeground(Color.WHITE);
welcomeLabel.setFont(new Font("Tahoma", Font.BOLD, 40));
welcomeLabel.setBounds(570, 30, 400, 60);
background.add(welcomeLabel);

Timer timer = new Timer(500, new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {

    String welcomeText = "WELCOME";
```

```
         welcomeIndex = (welcomeIndex + 1) % welcomeText.length();
         welcomeLabel.setText(welcomeText.substring(0, welcomeIndex + 1));
      }
   });
   timer.start();

}
public void actionPerformed(ActionEvent ae){
   if(ae.getSource()==b1){
   try{
      conn c1 = new conn();
      String u = t1.getText();
      String v = t2.getText();

      String q = "select * from login where username='"+u+"' and password='"+v+"'";

      ResultSet rs = c1.s.executeQuery(q);
      if(rs.next()){
         new Dashboard().setVisible(true);
         setVisible(false);
      }else{
         JOptionPane.showMessageDialog(null, "Invalid login");
         setVisible(false);
      }
   }catch(Exception e){
      e.printStackTrace();
   }
   }else if(ae.getSource()==b2){
      System.exit(0);
   }
}
```

```
   public static void main(String[] arg){
      new Login();
   }
}
package hostel.management.system;


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


public class Dashboard extends JFrame {


   public static void main(String[] args) {
      new Dashboard().setVisible(true);
   }


   int welcomeIndex = 0;


   public Dashboard() {
      super("HOSTEL MANAGEMENT SYSTEM");


     setForeground(Color.CYAN);
      setLayout(null);


      ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/2.2.jpg"));
      Image i2 = i1.getImage().getScaledInstance(1910, 1090, Image.SCALE_DEFAULT);
      ImageIcon i3 = new ImageIcon(i2);
      JLabel NewLabel = new JLabel(i3);
      NewLabel.setBounds(0, 0, 1910, 1090);


      ImageIcon q1 = new ImageIcon(ClassLoader.getSystemResource("icons/3.3.jpg"));
```

```
Image q2 = q1.getImage().getScaledInstance(1910, 1090, Image.SCALE_DEFAULT);
ImageIcon q3 = new ImageIcon(q2);
JLabel NewLabel2 = new JLabel(q3);
NewLabel2.setBounds(0, 1090, 1910, 1090);

ImageIcon w1 = new ImageIcon(ClassLoader.getSystemResource("icons/4.4.jpg"));
Image w2 = w1.getImage().getScaledInstance(1910, 1090, Image.SCALE_DEFAULT);
ImageIcon w3 = new ImageIcon(w2);
JLabel NewLabel3 = new JLabel(w3);
NewLabel3.setBounds(0, 2180, 1910, 1090);

ImageIcon e1 = new ImageIcon(ClassLoader.getSystemResource("icons/5.5.jpg"));
Image e2 = e1.getImage().getScaledInstance(1910, 1090, Image.SCALE_DEFAULT);
ImageIcon e3 = new ImageIcon(e2);
JLabel NewLabel4 = new JLabel(e3);
NewLabel4.setBounds(0, 3270, 1910, 1090);



JPanel contentPanel = new JPanel();
contentPanel.setLayout(null);
contentPanel.setPreferredSize(new Dimension(1950, 4360));
contentPanel.add(NewLabel);
contentPanel.add(NewLabel2);
contentPanel.add(NewLabel3);
 contentPanel.add(NewLabel4);



JScrollPane scrollPane = new JScrollPane(contentPanel);
scrollPane.setBounds(0, 0, 1910, 1000);
scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
```

```
add(scrollPane);



JLabel HostelManagementSystem = new JLabel("THE HAPPY PLACE");
HostelManagementSystem.setForeground(Color.WHITE);
HostelManagementSystem.setFont(new Font("Tahoma", Font.BOLD, 46));
HostelManagementSystem.setBounds(130, 60, 1000, 85);
NewLabel.add(HostelManagementSystem);

Timer timer = new Timer(500, new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      String welcomeText = "THE HAPPY PLACE";
      welcomeIndex = (welcomeIndex + 1) % welcomeText.length();
      HostelManagementSystem.setText(welcomeText.substring(0, welcomeIndex + 1));
   }
});
timer.start();

JMenuBar menuBar = new JMenuBar();
setJMenuBar(menuBar);

JMenu HostelSystem = new JMenu("HOSTEL MANAGEMENT");
HostelSystem.setForeground(Color.BLUE);
menuBar.add(HostelSystem);

JMenuItem HostelSystemDetails = new JMenuItem("RECEPTION");
HostelSystem.add(HostelSystemDetails);

JMenu HostelSystemHello = new JMenu("ADMIN");
```

```
HostelSystemHello.setForeground(Color.RED);
 menuBar.add(HostelSystemHello);

 JMenuItem HostelSystemDetailshello1 = new JMenuItem("Log Out");
 HostelSystemHello.add(HostelSystemDetailshello1);

 HostelSystemDetailshello1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
       try {
          new Login().setVisible(true);
          setVisible(false);
       } catch (Exception e) {
       }
    }
 });



 HostelSystemDetails.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        try{
                                RoomAllocationGUI custom = new RoomAllocationGUI();
                                custom.setVisible(true);
                        }catch(Exception e1){
                                e1.printStackTrace();
                        }
                        }
                });


 setSize(1950, 1090);
```

```
        setVisible(true);
        getContentPane().setBackground(Color.WHITE);
    }
}

package hostel.management.system;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.HashSet;
import java.util.Set;
import java.util.Random;
import java.time.LocalDateTime;
import java.util.Map;
import java.util.HashMap;
import javax.swing.table.TableColumnModel;


public class RoomAllocationGUI extends JFrame {
private JTextField nameTextField;
private JTextField marksTextField;
private final JLabel backgroundPanel;
private  JLabel backgroundPanel1;
private JTextField studentIdTextField;
private JTextField nameTextField1;
JLabel studentIdLabel;
private JButton generateReceiptButton;
private JFrame receiptFrame;


    private JComboBox<String> branchComboBox;

    private JComboBox<String> yearComboBox;

    private final JButton allocateButton;

    private final JButton showDetailsButton;

    private int nextRoomNumber;

    private int nextBedNumber;

    private final Connection connection;

    private final JTextField jTextFieldAllocatedBeds;

    private final JButton cancelAllocationButton;


    private double discountPercentage;


    private Set<String> studentIdSet;


    private Map<Integer, Room> allocatedRooms;






    public RoomAllocationGUI() {


        setTitle("Hostel Management System");

        setSize(860, 700);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLocationRelativeTo(null);


        ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons/7.jpg"));

        Image i2 = i1.getImage().getScaledInstance(850,650,Image.SCALE_DEFAULT);

        ImageIcon i3 =  new ImageIcon(i2);
```

```
        backgroundPanel = new JLabel(i3);
        backgroundPanel.setBounds(0,0,850,650);
        add(backgroundPanel);

        setLayout(null);
setContentPane(backgroundPanel);

        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
        nameLabel.setBounds(240,220,100,30);
        nameTextField = new JTextField(15);
         nameTextField.setBounds(315,220,210,30);
         backgroundPanel.add(nameLabel);
        backgroundPanel.add(nameTextField);



        JLabel studentIdLabel = new JLabel("Student ID:");
        studentIdLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
        studentIdLabel.setBounds(180, 160, 130, 30);
        backgroundPanel.add(studentIdLabel);

        studentIdTextField = new JTextField(15);
        studentIdTextField.setBounds(315, 160, 210, 30);
        backgroundPanel.add(studentIdTextField);

        JLabel text = new JLabel("ROOM BOOKING");
        text.setBounds(245, 0, 1000, 90);
        text.setForeground(Color.black);
        text.setFont(new Font("Serif", Font.PLAIN, 50));
        backgroundPanel.add(text);
```

```
JLabel marksLabel = new JLabel("Marks:");
marksTextField = new JTextField(15);
        marksTextField.setBounds(315,280,210,30);

marksLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
marksLabel.setBounds(235,280,100,30);
backgroundPanel.add(marksLabel);
backgroundPanel.add(marksTextField);

JLabel branchLabel = new JLabel("Branch:");
branchLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
branchLabel.setBounds(225,340,100,30);
String[] branches = {"CSE", "ECE", "Mechanical"};
branchComboBox = new JComboBox<>(branches);
branchComboBox.setForeground(Color.black);
branchComboBox.setFont(new Font("Tahoma", Font.BOLD, 14));
branchComboBox.setBounds(315,340,210,32);

backgroundPanel.add(branchLabel);
backgroundPanel.add(branchComboBox);

JLabel yearLabel = new JLabel("Year:");
String[] years = {"1", "2", "3", "4"};
yearLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
yearLabel.setBounds(250,400,100,30);
yearComboBox = new JComboBox<>(years);
yearComboBox.setFont(new Font("Tahoma", Font.BOLD, 16));
yearComboBox.setBounds(315,400,210,32);
backgroundPanel.add(yearLabel);
backgroundPanel.add(yearComboBox);
```

```
    allocateButton = new JButton("Allocate Room");
    allocateButton.setBounds(343,540,214,30);
    allocateButton.setFont(new Font("Tahoma", Font.BOLD, 14));
    allocateButton.setBackground(Color.BLACK);
    allocateButton.setForeground(Color.WHITE);
    backgroundPanel.add(allocateButton);

    generateReceiptButton = new JButton("Generate Receipt");
    generateReceiptButton.setBounds(690, 440, 140, 32);
    generateReceiptButton.setBackground(Color.green);
    generateReceiptButton.setForeground(Color.black);
    backgroundPanel.add(generateReceiptButton);

    generateReceiptButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        showReceiptGenerationPage();
    }
});
    showDetailsButton = new JButton(" Hostel Details");
    backgroundPanel.add(showDetailsButton);
    showDetailsButton.setBounds(690, 264, 140, 32);
    showDetailsButton.setBackground(Color.green);
    showDetailsButton.setForeground(Color.black);

    allocatedRooms = new HashMap<>();
    nextRoomNumber = 1;
    nextBedNumber = 1;

    jTextFieldAllocatedBeds = new JTextField();
    backgroundPanel.add(jTextFieldAllocatedBeds);
```

```
JButton btnExit = new JButton("Back");
btnExit.setBounds(170, 544, 120, 30);
    btnExit.setBackground(Color.RED);
    btnExit.setForeground(Color.WHITE);
backgroundPanel.add(btnExit );
          btnExit.addActionListener(new ActionListener() {
      @Override

                 public void actionPerformed(ActionEvent e) {
          new Dashboard().setVisible(true);
          setVisible(false);
                 }
          });

allocateButton.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    String name = nameTextField.getText();
    String marksStr = marksTextField.getText();
    String branch = (String) branchComboBox.getSelectedItem();
    String year = (String) yearComboBox.getSelectedItem();
    String studentId = studentIdTextField.getText();



    if (name.isEmpty() || marksStr.isEmpty()) {
      JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Please enter Name and
Marks.");
    } else {
      try {
        int marks = Integer.parseInt(marksStr);
```

```
            allocateRoom(name, marks, branch, year, studentId);                 }
          catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(RoomAllocationGUI.this,
                "Invalid Marks. Please enter a numeric value.");
          }
        }
      }
    });

    cancelAllocationButton = new JButton("Cancel Allocation");
    backgroundPanel.add(cancelAllocationButton);
    cancelAllocationButton.setBounds(690, 354, 140, 30);
        cancelAllocationButton.setBackground(Color.green);
        cancelAllocationButton.setForeground(Color.black);

    cancelAllocationButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        showCancellationPage();
      }
    });

    showDetailsButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        showHostelDetails();
      }
    });

    connection = getConnection();
    if (connection == null) {
            JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Failed to connect to the 
database.");
            System.exit(0);
        }

        loadAllocatedRooms();
          studentIdSet = new HashSet<>();
    }

    private Connection getConnection() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection("jdbc:mysql://localhost/hostelmanagementsystem", 
"root", "Kaavya@24");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    private String getRoomType(int marks, String branch) {
        if (marks >= 90) {
            return "Single";
        } else if (marks >= 80) {
            return "Double";
        } else if (branch.equals("CSE")) {
            return "Triple";
        } else {
            return "Quadruple";
        }
    }
 private Room allocateNextRoom(String roomType) {
    for (Room room : allocatedRooms.values()) {
        if (room.getRoomType().equals(roomType) && room.hasAvailableBed()) {
            int nextBedNumber = room.getNextBedNumber();
            room.allocateBed(nextBedNumber);
            return room;
        }
    }
    for (Room room : allocatedRooms.values()) {
        if (room.hasAvailableBed()) {
            int nextBedNumber = room.getNextBedNumber();
            room.allocateBed(nextBedNumber);
            return room;
        }
    }
    if (nextRoomNumber > 15) {
        JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Maximum number of rooms
reached.");
        throw new IllegalStateException("Maximum number of rooms reached.");
    }
    for (int i = 1; i < nextRoomNumber; i++) {
        if (!allocatedRooms.containsKey(i)) {
            Room vacantRoom = new Room(i, roomType);
            vacantRoom.allocateBed(1);
            allocatedRooms.put(i, vacantRoom);
            return vacantRoom;
        }
    }
    Room newRoom = new Room(nextRoomNumber, roomType);
    newRoom.allocateBed(1);
    allocatedRooms.put(nextRoomNumber, newRoom);
    nextRoomNumber++;
```

```
  return newRoom;
}
private void allocateRoom(String name, int marks, String branch, String year,String studentId) {
   if (marks < 45) {
     JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Not eligible for room allocation
for this Term");
     return;
  }
  String roomType = getRoomType(marks, branch);


  try {
     PreparedStatement statement = connection.prepareStatement(
          "SELECT COUNT(*) FROM room_allocation WHERE student_id = ?"
     );
     statement.setString(1, studentId);
     ResultSet resultSet = statement.executeQuery();
     resultSet.next();
     int count = resultSet.getInt(1);
     if (count > 0) {
        JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Student ID already exists.");
        nameTextField.setText("");
        marksTextField.setText("");
        branchComboBox.setSelectedIndex(0);
        yearComboBox.setSelectedIndex(0);
        studentIdTextField.setText("");
        return;
     }
  } catch (SQLException e) {
     e.printStackTrace();
  }
  Room room = allocateNextRoom(roomType);
 if (room == null) {
    JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Maximum number of rooms
reached.");
    return;
  }
  room.addStudent(name,marks);
  room.addStudentMarks(name, marks);
  room.addStudentBranch(name, branch);
  room.addStudentYear(name, year);
  room.addStudentId(studentId);
  room.setRoomType(roomType);
    allocatedRooms.put(room.getRoomNumber(), room);

  try {
    PreparedStatement statement = connection.prepareStatement(
        "INSERT INTO room_allocation (room_number, bed_number, name, marks, branch, year,
room_type,student_id) VALUES (?,?, ?, ?, ?, ?, ?, ?)"
    );
    statement.setInt(1, room.getRoomNumber());
    List<Integer> allocatedBedNumber = room.getAllocatedBedNumber();
    if (!allocatedBedNumber.isEmpty()) {
      statement.setInt(2, allocatedBedNumber.get(allocatedBedNumber.size() - 1));
    } else {
      throw new IllegalStateException("No allocated bed numbers for the room.");
    }
    statement.setString(3, name);
    statement.setInt(4, marks);
    statement.setString(5, branch);
    statement.setString(6, year);
    statement.setString(7, roomType);
    statement.setString(8, studentId);
```

```
        statement.executeUpdate();

        JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Student allocated to the room
successfully!");

          studentIdSet.add(studentId);

          nameTextField.setText("");
          marksTextField.setText("");
          branchComboBox.setSelectedIndex(0);
          yearComboBox.setSelectedIndex(0);
          studentIdTextField.setText("");
    } catch (SQLException e) {
        e.printStackTrace();
    }

    updateRoomAllocation();
}




    private void updateRoomAllocation() {
        StringBuilder allocatedBeds = new StringBuilder();
        for (Room room : allocatedRooms.values()) {
            allocatedBeds.append(room.getBedNumbersAsString()).append(", ");
        }
        jTextFieldAllocatedBeds.setText(allocatedBeds.toString());
    }

    private void loadAllocatedRooms() {
        try {
```

```
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery("SELECT * FROM room_allocation");
studentIdSet = new HashSet<>();
    while (resultSet.next()) {
        int roomNumber = resultSet.getInt("room_number");
        int bedNumber = resultSet.getInt("bed_number");
        String name = resultSet.getString("name");
        int marks = resultSet.getInt("marks");
        String branch = resultSet.getString("branch");
        String year = resultSet.getString("year");
        String studentId = resultSet.getString("student_id");
        String roomType = resultSet.getString("room_type");



        Room room = findOrCreateRoom(roomNumber);
        room.allocateBed(bedNumber);
        room.addStudent(name, marks);
        room.addStudentMarks(name, marks);
        room.addStudentBranch(name, branch);
        room.addStudentYear(name, year);
       room.addStudentId(studentId);
       studentIdSet.add(studentId);
        room.setRoomType(roomType);
        if (bedNumber >= nextBedNumber && roomNumber >= nextRoomNumber) {
        nextBedNumber = bedNumber + 1;
        nextRoomNumber = roomNumber + 1;
    }
    }
} catch (SQLException e) {
    e.printStackTrace();
```

```
        }
    }

    private Room findOrCreateRoom(int roomNumber) {
        for (Room room : allocatedRooms.values()) {
            if (room.getRoomNumber() == roomNumber) {
                return room;
            }
        }

        Room room = new Room(roomNumber);
        allocatedRooms.put(roomNumber, room);
        if (roomNumber >= nextRoomNumber) {
            nextRoomNumber = roomNumber + 1;
        }
        return room;
    }

    private void showCancellationPage() {

        ImageIcon p1 = new ImageIcon(ClassLoader.getSystemResource("icons/7.jpg"));
        Image p2 = p1.getImage().getScaledInstance(850,650,Image.SCALE_DEFAULT);
        ImageIcon p3 =  new ImageIcon(p2);
        backgroundPanel1 = new JLabel(p3);
        backgroundPanel1.setBounds(0,0,850,650);
        add(backgroundPanel1);



setTitle("Cancellation Page");
        setSize(860, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        setLayout(null);
        setContentPane(backgroundPanel1);



        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
        nameLabel.setBounds(215,200,100,30);
        nameTextField = new JTextField(15);
         nameTextField.setBounds(290,200,210,30);
        backgroundPanel1.add(nameLabel);
        backgroundPanel1.add(nameTextField);

        studentIdLabel = new JLabel("Student ID:");
studentIdLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
studentIdLabel.setBounds(160, 270, 400, 30);
backgroundPanel1.add(studentIdLabel);



 nameTextField1 = new JTextField(15);
         nameTextField1.setBounds(290,270,210,30);
         backgroundPanel1.add(nameTextField1);



        JLabel text1 = new JLabel("ROOM CANCELLATION");
        text1.setBounds(155, 0, 1000, 90);
        text1.setForeground(Color.black);
        text1.setFont(new Font("Serif", Font.PLAIN, 50));
        backgroundPanel1.add(text1);
        JLabel branchLabel = new JLabel("Branch:");
        branchLabel.setBounds(203,330,100,30);
        branchLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
        String[] branches = {"CSE", "ECE", "Mechanical"};
        JComboBox<String> branchComboBox = new JComboBox<>(branches);
        branchComboBox.setForeground(Color.black);
        branchComboBox.setFont(new Font("Tahoma", Font.BOLD, 14));
        branchComboBox.setBounds(290,330,210,32);
        backgroundPanel1.add(branchLabel);
        backgroundPanel1.add(branchComboBox);



        JLabel yearLabel = new JLabel("Year:");
        String[] years = {"1", "2", "3", "4"};
        yearLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
        yearLabel.setBounds(220,400,100,30);
        JComboBox<String> yearComboBox = new JComboBox<>(years);
        yearComboBox.setFont(new Font("Tahoma", Font.BOLD, 16));
        yearComboBox.setBounds(290,400,210,32);
        backgroundPanel1.add(yearLabel);
        backgroundPanel1.add(yearComboBox);


        JButton cancelButton = new JButton("Cancel");
        backgroundPanel1.add(cancelButton);
            cancelButton.setFont(new Font("Tahoma", Font.BOLD, 16));
    cancelButton.setBounds(344,540,213,30);
            cancelButton.setBackground(Color.red);
            cancelButton.setForeground(Color.black);
    JButton btnExit = new JButton("Back");
    btnExit.setBounds(170, 544, 120, 30);
            btnExit.setBackground(Color.blue);
            btnExit.setForeground(Color.WHITE);
    backgroundPanel1.add(btnExit );
                    btnExit.addActionListener(new ActionListener() {
            @Override

                            public void actionPerformed(ActionEvent e) {
                    new RoomAllocationGUI().setVisible(true);
                    setVisible(false);
                            }
                    });

    cancelButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String name = nameTextField.getText();
            String branch = (String) branchComboBox.getSelectedItem();
            String year = (String) yearComboBox.getSelectedItem();

            cancelAllocation(name, branch, year);
            backgroundPanel1.setVisible(true);
        }
    });

    backgroundPanel1.setVisible(true);
}
```

```
private void cancelAllocation(String name, String branch, String year) {
String studentId = nameTextField1.getText();
if (studentId.isEmpty()) {
   JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Please enter Student ID.");
   return;
}

boolean cancellationSuccessful = false;
   for (Room room : allocatedRooms.values()) {
   List<String> students = room.getStudents();
   List<String> studentBranches = room.getStudentBranches();
   List<String> studentYears = room.getStudentYears();
   Set<String> studentIds = room.getStudentIds();

   if (studentIds == null) {
      continue;
   }
    List<String> studentIdList = new ArrayList<>(studentIds);

   for (int i = 0; i < students.size(); i++) {
      String student = students.get(i);
      String studentBranch = studentBranches.get(i);
      String studentYear = studentYears.get(i);
      String studentIdFromList = studentIdList.get(i);
```

```
        if (student.equals(name) && studentBranch.equals(branch) && studentYear.equals(year)
&& studentId.equals(studentIdFromList)) {
            students.remove(i);
            studentBranches.remove(i);
            studentYears.remove(i);
             studentIds.remove(studentIdFromList);
            try {
                PreparedStatement statement = connection.prepareStatement(
                        "DELETE FROM room_allocation WHERE room_number = ? AND name = ? AND
branch = ? AND year = ? AND student_id = ?"
                );
                statement.setInt(1, room.getRoomNumber());
                statement.setString(2, name);
                statement.setString(3, branch);
                statement.setString(4, year);
                statement.setString(5, studentId);
                statement.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }

            JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Allocation canceled
successfully!");
            updateRoomAllocation();

            nameTextField.setText("");
            marksTextField.setText("");
            branchComboBox.setSelectedIndex(0);
            yearComboBox.setSelectedIndex(0);
             nameTextField1.setText("");

            cancellationSuccessful = true;
            break;
          }
        }

        if (cancellationSuccessful) {
          break;
        }
      }

      if (!cancellationSuccessful) {
        JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Allocation not found for the given
details or invalid Student ID.");
      }
    }



    private void showHostelDetails() {
    JFrame detailsFrame = new JFrame("Hostel Details");
    detailsFrame.setSize(960, 700);
    detailsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    detailsFrame.setLocationRelativeTo(null);
    detailsFrame.setLayout(new BorderLayout());

    String[] columnNames = {"Room Number", "Room Type", "Bed Numbers", "Students", "Branch",
"Year", "Student IDs"};
    Object[][] rowData = new Object[allocatedRooms.size()][7];
    int i = 0;
    for (Room room : allocatedRooms.values()) {
      String bedNumbers = room.getBedNumbersAsString();
      String students = room.getStudentsAsString();
      List<String> branches = room.getStudentBranches();
      List<String> years = room.getStudentYears();
    Set<String> studentIds = room.getStudentIds();


    rowData[i][0] = room.getRoomNumber();
    rowData[i][1] = room.getRoomType();
    rowData[i][2] = bedNumbers;
    rowData[i][3] = students;
    rowData[i][4] = branches;
    rowData[i][5] = years;
    rowData[i][6] = studentIds;
    i++;
}

JTable table = new JTable(rowData, columnNames);
table.getTableHeader().setFont(new Font("Tahoma", Font.BOLD, 14));
table.setFont(new Font("Tahoma", Font.PLAIN, 14));
table.setRowHeight(45);
table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);

TableColumnModel columnModel = table.getColumnModel();
columnModel.getColumn(0).setPreferredWidth(120);
columnModel.getColumn(1).setPreferredWidth(100);
columnModel.getColumn(2).setPreferredWidth(150);
columnModel.getColumn(3).setPreferredWidth(150);
columnModel.getColumn(4).setPreferredWidth(150);
columnModel.getColumn(5).setPreferredWidth(80);
columnModel.getColumn(6).setPreferredWidth(200);

JScrollPane scrollPane = new JScrollPane(table);
        table.setBackground(new Color(173, 216, 230));
    table.setForeground(Color.BLUE);
detailsFrame.add(scrollPane, BorderLayout.CENTER);
```

```
    detailsFrame.setVisible(true);
}




   private double calculateTotalAmount(String roomType, int studentMarks) {
    double roomPrice = 0.0;

    if (roomType.equals("Single")) {
        roomPrice = 100000.0;
    } else if (roomType.equals("Double")) {
        roomPrice = 85000.0;
    } else if (roomType.equals("Triple")) {
        roomPrice = 75000.0;
    } else if (roomType.equals("Quadruple")) {
        roomPrice = 55000.0;
    }


     discountPercentage = 0.0;
    if (studentMarks >= 90) {
        discountPercentage = 20.0;
    } else if (studentMarks >= 80) {
        discountPercentage = 15.0;
    } else if (studentMarks >= 70) {
        discountPercentage = 10.0;
    }
    double discountAmount = roomPrice * (discountPercentage / 100.0);
    double discountedPrice = roomPrice - discountAmount;



    return discountedPrice;
}



private Receipt generateReceipt(String studentId) {
    try {
        PreparedStatement statement = connection.prepareStatement(
            "SELECT * FROM room_allocation WHERE student_id = ?"
        );
        statement.setString(1, studentId);
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            String studentName = resultSet.getString("name");
            int roomNumber = resultSet.getInt("room_number");
            String roomType = resultSet.getString("room_type");
            int studentMarks = resultSet.getInt("marks");



            double totalAmount = calculateTotalAmount(roomType,studentMarks);



            String paymentMethod = JOptionPane.showInputDialog(RoomAllocationGUI.this, "Enter
Payment Method (Cash/Card):");
            String paymentDate = LocalDateTime.now().toString();
            int receiptNumber = generateRandomReceiptNumber();
Receipt receipt = new Receipt(studentName, studentId, roomNumber, roomType,
paymentMethod, paymentDate, totalAmount, receiptNumber,discountPercentage);



            return receipt;

        } else {

            JOptionPane.showMessageDialog(RoomAllocationGUI.this, "Allocation not found for the
given Student ID.");

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }


    return null;

}


private int generateRandomReceiptNumber() {

    Random random = new Random();

    int receiptNumber = random.nextInt(9000) + 1000;

    return receiptNumber;

}



    private void showReceipt(Receipt receipt) {

    String receiptText = generateReceiptText(receipt);

    JOptionPane.showMessageDialog(RoomAllocationGUI.this, receiptText, "Receipt",
JOptionPane.INFORMATION_MESSAGE);

}



private String generateReceiptText(Receipt receipt) {
```

```
String studentName = receipt.getStudentName();
String studentId = receipt.getStudentId();
int roomNumber = receipt.getRoomNumber();
String roomType = receipt.getRoomType();
String paymentMethod = receipt.getPaymentMethod();
String paymentDate = receipt.getPaymentDate();
double totalAmount = receipt.getTotalAmount();
int receiptNumber =receipt.getReceiptNumber();

StringBuilder receiptBuilder = new StringBuilder();
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("          HOSTEL RECEIPT\n");
receiptBuilder.append("----------------------------------------\n\n");
receiptBuilder.append("Date: ").append(paymentDate).append("\n");
receiptBuilder.append("Receipt Number: ").append(receiptNumber).append("\n\n");
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("Student Information:\n");
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("Name: ").append(studentName).append("\n");
receiptBuilder.append("Student ID: ").append(studentId).append("\n");
receiptBuilder.append("Room Number: ").append(roomNumber).append("\n");
receiptBuilder.append("Room Type: ").append(roomType).append("\n\n");
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("Payment Details:\n");
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("Payment Method: ").append(paymentMethod).append("\n");
receiptBuilder.append("Discount Applied: ").append(discountPercentage).append("%\n");
receiptBuilder.append("Total Amount: ").append(totalAmount).append("\n");
receiptBuilder.append("----------------------------------------\n");
receiptBuilder.append("Thank you for your payment!\n");
receiptBuilder.append("For any queries, please contact the hostel management.\n");
```

```
   receiptBuilder.append("----------------------------------------\n");
 return receiptBuilder.toString();
}
private void showReceiptGenerationPage() {
   JFrame receiptFrame = new JFrame("Receipt Generation");
   receiptFrame.setSize(860, 700);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   receiptFrame.setLocationRelativeTo(null);
   receiptFrame.setLayout(null);

   ImageIcon p1 = new ImageIcon(ClassLoader.getSystemResource("icons/6.6.jpg"));
   Image p2 = p1.getImage().getScaledInstance(850, 650, Image.SCALE_DEFAULT);
   ImageIcon p3 = new ImageIcon(p2);
   JLabel backgroundPanel1 = new JLabel(p3);
   backgroundPanel1.setBounds(0, 0, 850, 650);
   backgroundPanel1.setLayout(null);

   studentIdLabel = new JLabel("Student ID:");
   studentIdLabel.setFont(new Font("Tahoma", Font.BOLD, 22));
   studentIdLabel.setBounds(173,200,230,30);
   studentIdTextField = new JTextField(15);
   studentIdTextField.setBounds(315,200,210,30);

   JButton generateButton = new JButton("Generate");
   backgroundPanel1.add(studentIdLabel);
   backgroundPanel1.add(studentIdTextField);
   backgroundPanel1.add(generateButton);
       generateButton.setBounds(270, 254, 140, 30);
           generateButton.setBackground(Color.green);
           generateButton.setForeground(Color.black);
```

```
    receiptFrame.add(backgroundPanel1);
    backgroundPanel1.setVisible(true);

    generateButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String studentId = studentIdTextField.getText();
            Receipt receipt = generateReceipt(studentId);
            if (receipt != null) {
                showReceipt(receipt);
                receiptFrame.setVisible(true);
            }
        }
    });

    receiptFrame.setVisible(true);
}
public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new RoomAllocationGUI().setVisible(true);
            }
        });
    }
}
package hostel.management.system;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import java.util.HashSet;


import java.util.StringJoiner;


public class Room {
    private int roomNumber;
    private String roomType;
    private List<Integer> bedNumber;
    private List<String> students;
    private List<Integer> studentMarks;
    private List<String> studentBranches;
    private List<String> studentYears;
     private String bedType;
     private Set<String> studentIds;



    public Room(int roomNumber) {
        this.roomNumber = roomNumber;
        this.roomType = "";
        this.bedNumber = new ArrayList<>();
        this.students = new ArrayList<>();
        this.studentMarks = new ArrayList<>();
        this.studentBranches = new ArrayList<>();
        this.studentYears = new ArrayList<>();
        this.studentIds = new HashSet<>();    }

    public Room(int roomNumber, String roomType) {
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.bedNumber = new ArrayList<>();
        this.students = new ArrayList<>();
```

```
        this.studentMarks = new ArrayList<>();
        this.studentBranches = new ArrayList<>();
        this.studentYears = new ArrayList<>();
         this.studentIds = new HashSet<>();
    }

    public int getRoomNumber() {
        return roomNumber;
    }
    public void setRoomType(String roomType) {
    this.roomType = roomType;
}



    public String getRoomType() {
        return roomType;
    }

    public List<Integer> getBedNumber() {
        return bedNumber;
    }

    public String getBedNumbersAsString() {
        StringJoiner joiner = new StringJoiner(", ");
        for (int bedNumbers : bedNumber) {
            joiner.add(String.valueOf(bedNumbers));
        }
        return joiner.toString();
    }

    public List<String> getStudents() {
```

```
    return students;
}

public int getNextBedNumber() {
    for (int i = 1; i <= getMaxCapacity(); i++) {
        if (!bedNumber.contains(i)) {
            return i;
        }
    }
    return -1;
}



public String getStudentsAsString() {
    return String.join(", ", students);
}

public void allocateBed(int bedNumbers) {
    bedNumber.add(bedNumbers);
}

public List<String> getStudentBranches() {
    return studentBranches;
}

public List<String> getStudentYears() {
    return studentYears;
}



public Set<String> getStudentIds() {
    return studentIds;
}



public List<Integer> getAllocatedBedNumber() {
    return bedNumber;
}

    public boolean hasAvailableBed() {
    return bedNumber.size() < getMaxCapacity();
}

public void addStudentId(String student, int marks) {
    students.add(student);
    studentMarks.add(marks);
}

public void addStudentMarks(String student, int marks) {
    studentMarks.add(marks);
}

public void addStudentBranch(String student, String branch) {
    studentBranches.add(branch);
}

public void addStudentYear(String student, String year) {
    studentYears.add(year);
}

public void addStudentId(String studentId) {
    studentIds.add(studentId);
```

```
  }

public String getStudentIdsAsString() {
      return String.join(", ", studentIds);
  }
public boolean isVacant() {
      return students.isEmpty();
  }
  public int getVacantBedCount() {
      return getMaxCapacity() - bedNumber.size();
  }

   private int getMaxCapacity() {
      if (roomType.equals("Single")) {
         return 1;
      } else if (roomType.equals("Double")) {
         return 2;
      } else if (roomType.equals("Triple")) {
         return 3;
      } else if (roomType.equals("Quadruple")) {
         return 4;
      } else {
         return 0;
      }
  }


  public void deallocateBed(int bedNumber) {
      this.bedNumber.remove(Integer.valueOf(bedNumber));
  }
}
```

```
package hostel.management.system;



import java.util.HashMap;
import java.util.Map;

public class Receipt {
   private Map<String, Object> receiptData;

   public Receipt(String studentName, String studentId, int roomNumber, String roomType,
                String paymentMethod, String paymentDate, double roomPrice,
               double discountPercentage, double receiptNumber) {
      receiptData = new HashMap<>();

      receiptData.put("studentName", studentName);
      receiptData.put("studentId", studentId);
      receiptData.put("roomNumber", roomNumber);
      receiptData.put("roomType", roomType);
      receiptData.put("paymentMethod", paymentMethod);
      receiptData.put("paymentDate", paymentDate);
      receiptData.put("roomPrice", roomPrice);

      receiptData.put("discountPercentage", discountPercentage);
      receiptData.put("receiptNumber", receiptNumber);
   }

   public String getStudentName() {
      return (String) receiptData.get("studentName");
   }

   public String getStudentId() {
      return (String) receiptData.get("studentId");
   }

   public int getRoomNumber() {
      return (int) receiptData.get("roomNumber");
   }

   public String getRoomType() {
      return (String) receiptData.get("roomType");
   }

   public String getPaymentMethod() {
      return (String) receiptData.get("paymentMethod");
   }

   public String getPaymentDate() {
      return (String) receiptData.get("paymentDate");
   }

   public double getTotalAmount() {
   double totalAmount = (double) receiptData.get("roomPrice");
   double discountPercentage = (double) receiptData.get("discountPercentage");
   return totalAmount ;
}



   public double getDiscountApplied() {
      return (double) receiptData.get("discountApplied");
   }

   public int getReceiptNumber() {
      return (int) Math.round((double) receiptData.get("receiptNumber"));
   }
}
```

In this provided code, there are several instances where data structures are used to store and manage data within the Hostel Management System.

HashMap (allocatedRooms): The allocatedRooms variable is a HashMap that stores the allocated rooms in the hostel. The key is the room number, and the value is an instance of the Room class. This data structure allows efficient retrieval and manipulation of allocated rooms based on their room numbers.

Set (studentIdSet): The studentIdSet variable is a HashSet that stores the unique student IDs of the allocated students. This data structure ensures that each student ID is unique and prevents duplicate allocations.

List (allocatedBedNumber): The allocatedBedNumber list is used within the Room class to keep track of the allocated bed numbers within a particular room. It allows for the allocation of multiple beds within a room and maintains the order of the allocated bed numbers.

List (bedNumber): The bedNumber list holds the numbers of the allocated beds in the room. It allows for the allocation of multiple beds within a room.

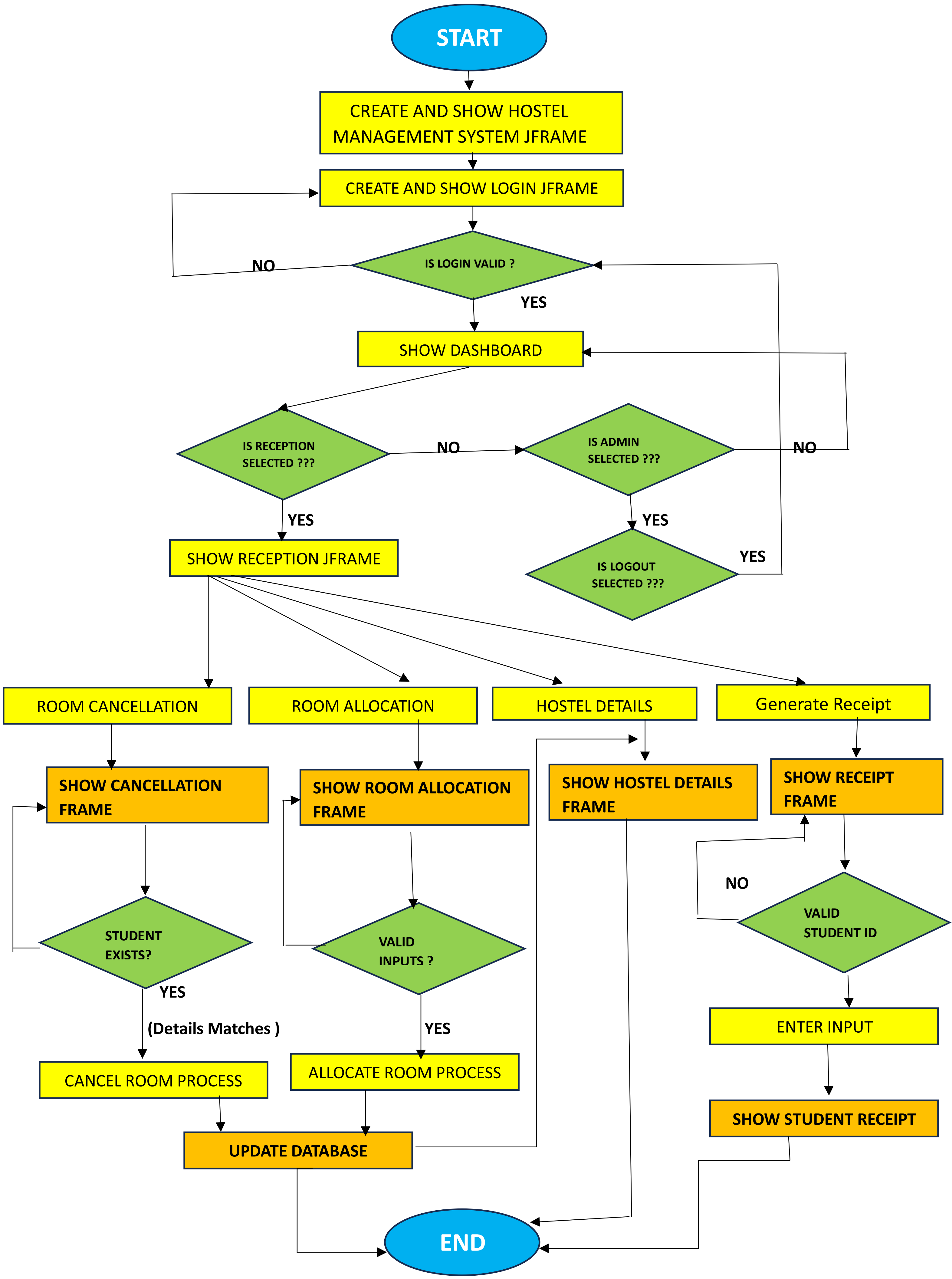
List (students): The students list stores the names of the students occupying the room.

List (studentMarks): The studentMarks list stores the marks of the students occupying the room. The order of marks corresponds to the order of students in the students list.

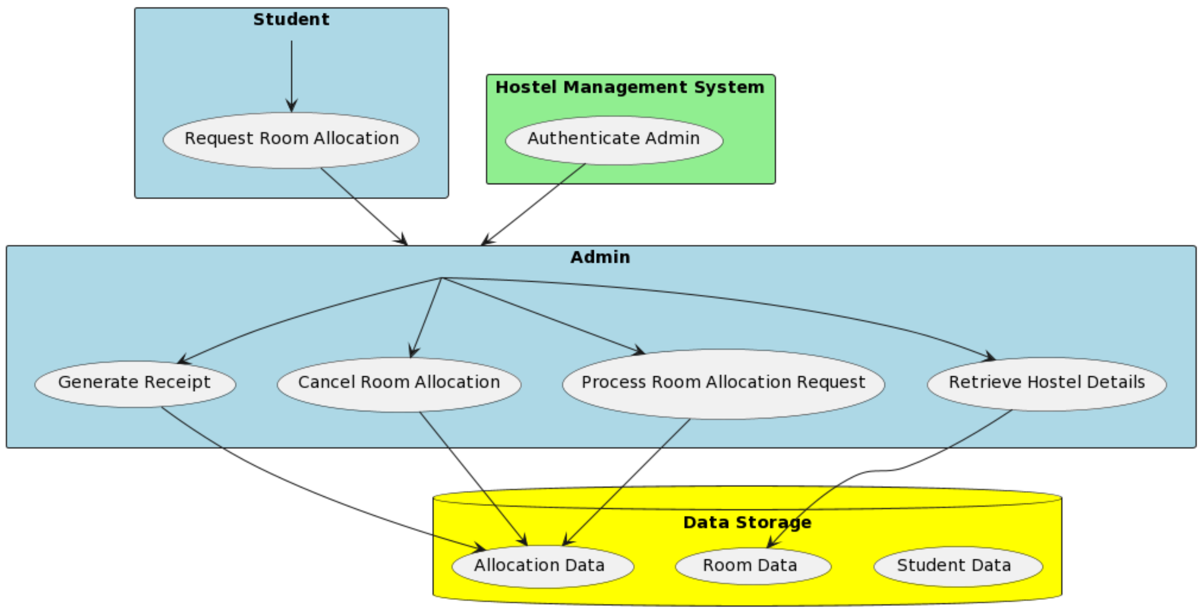
List (studentBranches): The studentBranches list holds the branches of the students occupying the room. The order of branches corresponds to the order of students in the students list.

List (studentYears): The studentYears list stores the academic years of the students occupying the room. The order of years corresponds to the order of students in the students list.

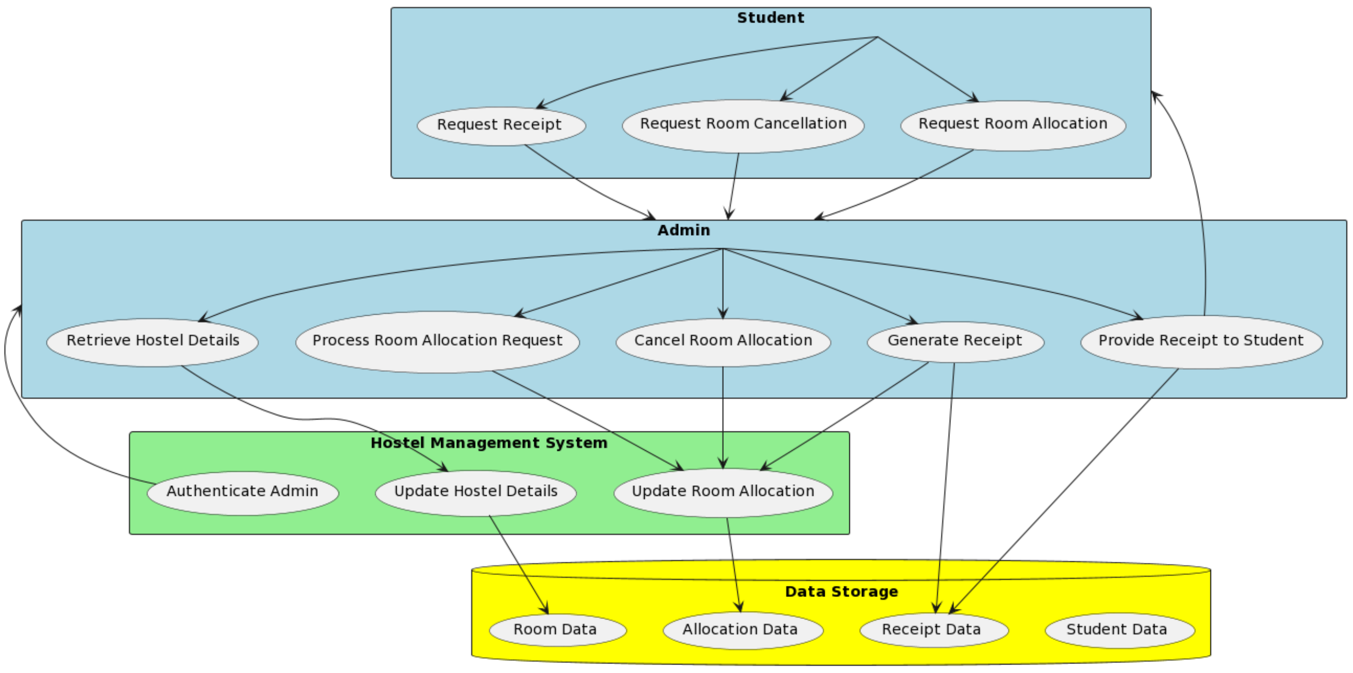
These data structures facilitate efficient storage, retrieval, and manipulation of information within the Hostel Management System. The HashMap allows quick access to allocated rooms, the Set ensures uniqueness of student IDs, and the List maintains the order of allocated bed numbers within each room. By leveraging appropriate data structures, the system can handle room allocation and retrieval operations effectively.


START
CREATE AND SHOW HOSTEL MANAGEMENT SYSTEM JFRAME
CREATE AND SHOW LOGIN JFRAME
IS LOGIN VALID ?
NO
YES
SHOW DASHBOARD
IS RECEPTION SELECTED ???
NO
IS ADMIN SELECTED ???
NO
YES
YES
SHOW RECEPTION JFRAME
IS LOGOUT SELECTED ???
YES
ROOM CANCELLATION
ROOM ALLOCATION
HOSTEL DETAILS
Generate Receipt
SHOW CANCELLATION FRAME
SHOW ROOM ALLOCATION FRAME
SHOW HOSTEL DETAILS FRAME
SHOW RECEIPT FRAME
NO
VALID STUDENT ID
STUDENT EXISTS?
VALID INPUTS ?
YES
(Details Matches)
YES
ENTER INPUT
CANCEL ROOM PROCESS
ALLOCATE ROOM PROCESS
SHOW STUDENT RECEIPT
UPDATE DATABASE
END

Hostel Management System - Level 0 DFD



Hostel Management System - Level 1 DFD


Student
Request Receipt
Request Room Cancellation
Request Room Allocation
Admin
Retrieve Hostel Details
Process Room Allocation Request
Cancel Room Allocation
Generate Receipt
Provide Receipt to Student
Hostel Management System
Authenticate Admin
Update Hostel Details
Update Room Allocation
Data Storage
Room Data
Allocation Data
Receipt Data
Student Data