HOSTEL MANAGEMENT SYSTEM.

A MINI-PROJECT REPORT

Submitted by

SARAVANAN B 230701293

SANJAY G 230701286

In partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project “HOSTEL MANAGEMENT SYSTEMS” is the

Bonafide work of “SARAVANAN B AND SANJAY G” who carried out the

project work under my supervision.

SIGNATURE
MRS V JANANEE
ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

SIGNATURE
MR SARAVANA GOKUL
ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on ______

TABLE OF CONTENTS

ABSTRACT

**Abstract: **

A hotel management website serves as a comprehensive digital platform designed to streamline and enhance the overall guest experience, operations, and management of hotels. It typically offers features like online booking, reservation management, room availability, guest check-in/check-out, payment processing, and customer feedback systems. The website may also integrate with property management systems (PMS) to automate administrative tasks, track guest preferences, and handle housekeeping and maintenance requests.

Additionally, it often includes sections for marketing purposes such as special promotions, packages, and virtual tours of the hotel. The aim is to provide a seamless interface for both guests and hotel staff, improving efficiency, optimizing guest satisfaction, and driving revenue through enhanced accessibility and real-time service capabilities. With the rise of mobile and responsive design, hotel management websites are increasingly mobile-friendly, ensuring a consistent user experience across devices. By offering a central hub for all operational and service-related needs, these websites play a crucial role in modern hospitality management.

INTRODUCTION

Key Features of a Hostel Management System Using Java Swing

1. **User Authentication and Role-Based Access**

- **Login Page**: A secure login page where hotel staff (administrators, receptionists, managers, etc.) can sign in with their credentials.
- Role-Based Access Control: Depending on the user's role (admin, manager, or receptionist), the system grants different levels of access to features like room management, billing, reports, and user settings.
- Password Protection: Ensures sensitive data is accessed only by authorized personnel.

---

2. **Manage hostel and Add Hostel**

1. Manage Hostel

The **"Manage Hostel"** module provides hostel administrators with a central interface to manage all the essential aspects of hostel operations, such as:

Key Features:

- **View Hostel Details:**
  - A comprehensive dashboard where the administrator can view all the hostels and their details (name, location, number of rooms, capacity, available services, etc.).
  - Option to filter or search hostels based on various criteria (location, room type, available space, etc.).
  -

- **Manage Rooms:**
  - Administrators can view room availability, occupancy status, and room types (single, shared, deluxe, etc.).
  - Ability to update room details (capacity, cost, room features, etc.) and manage room assignments.

---

3. View Hostel

View Hostel Module in Hostel Management System

The **View Hostel** module is a crucial component of a Hostel Management System, allowing administrators and staff to easily access and manage information about hostels, their rooms, and resident students. This module provides a comprehensive overview of hostel details, occupancy status, and associated services, ensuring effective management and smooth operations.

Key Features of the View Hostel Module

1. **Hostel Overview:**
   - Display a list of all hostels in the system, including essential details such as:
     - Hostel Name
     - Location (city, area)
     - Total Number of Rooms
     - Current Occupancy Rate

        - Contact Information

2. **Room Details:**

    - For each hostel, provide an overview of room availability, including:

        - Room Numbers/IDs
        - Room Types (single, double, shared, etc.)
        - Room Status (available, occupied, under maintenance)
        - Daily/Monthly Rates
        - Amenities available in each room (e.g., Wi-Fi, AC)

---

4. **Search Option**

To implement a **search option** for the **Hostel Management System** project, we'll expand on the concept of searching for hostels, rooms, and student details. The search functionality should be dynamic and allow administrators to find relevant information quickly based on various parameters such as hostel name, room availability, location, or student assignment.

Here's how the **search option** could be implemented for different sections of the hostel management project:

1. Search Hostels

- The user can search for **hostels** based on parameters such as:
    - Hostel name
    - Location
    - Room availability (e.g., "available", "full")
    - Total rooms

2. Search Rooms

- The user can search for **rooms** based on criteria like:
    - Room number
    - Room type (e.g., single, double, shared, etc.)
    - Availability status (available, occupied, under maintenance)
    - Price per room

3. Search Students

- The user can search for **students** based on parameters such as:
    - Student name
    - Student ID
    - Room number
    - Check-in or check-out dates
    - Payment status

5.Back Operation:

To implement a **Back Operation** in the Hostel Management System project (such as navigating from a detail or searching page back to the main dashboard or menu), you can use a **Back Button** to take the user to the previous screen or main window.

In Java Swing, when you're creating a GUI, navigation typically involves switching between different panels or frames. The **Back Button** can be used to either close the current window and open a previous one or show a different panel within the same frame.

Here’s a breakdown of how you could implement the **Back Operation** in your **Hostel Management System**:

1. Back Operation in a New Window

If the **Back Button** needs to navigate between different windows (e.g., from a **Search Hostels** window back to the **Main Dashboard** window), we would use the dispose () method to close the current window and then create a new window (i.e., the main dashboard or previous screen).

Example with a Main Dashboard and Search Window

Let's assume the flow of the system has two windows:

1. **Main Dashboard (Home)**: Where the user can navigate to different sections.

2. **Search Hostels Window**: The user performs the search here.

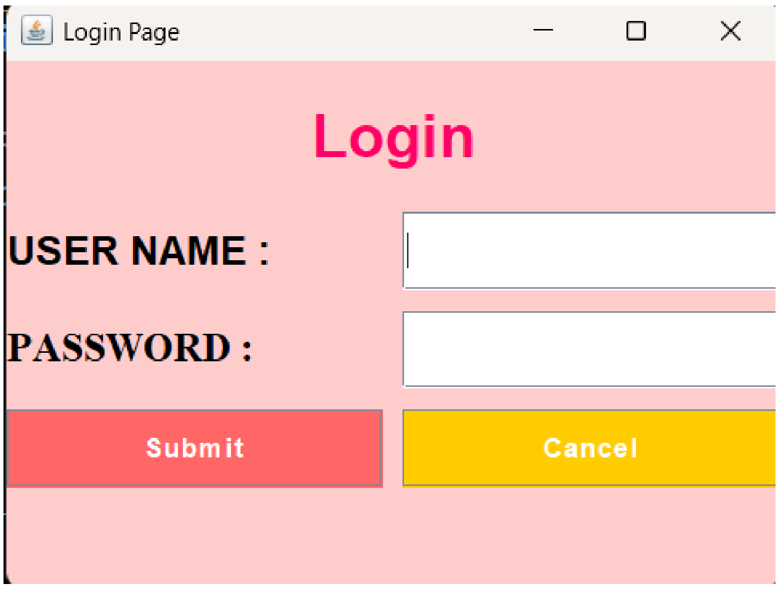
   We'll implement the back button so that clicking "Back" in the **Search Hostels** window will close it and show the **Main Dashboard** again.

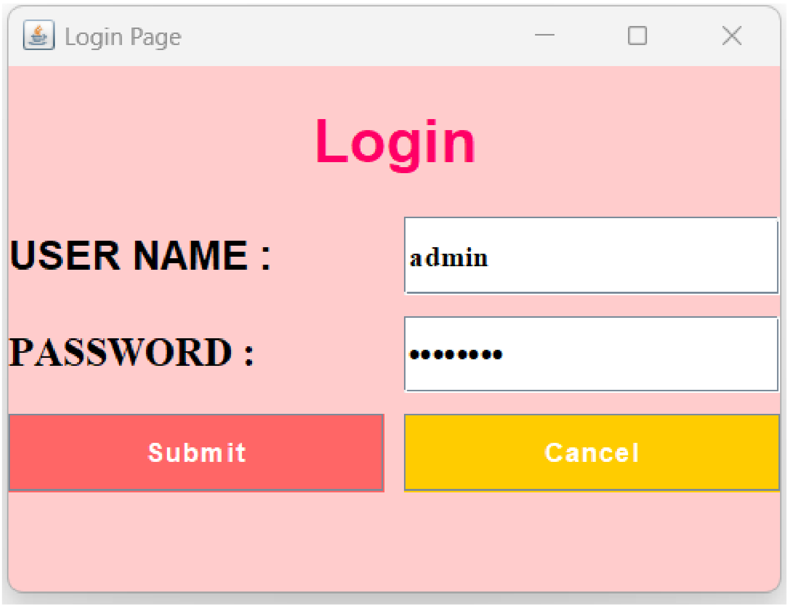
SCOPE OF THE PROJECT

The **Hostel Management System** focuses on efficiently managing hostel operations with an emphasis on adding and managing hostels by name and related details. It enables administrators to add, update, and delete hostels, view information on occupancy and room availability, and manage room assignments. The system allows student management by assigning students to specific rooms and tracking their check-in and check-out dates. A robust search functionality lets users quickly find hostels by name, location, or availability. Additionally, the system tracks billing by generating bills for student stays and monitoring payment statuses. With an intuitive dashboard interface developed using Java Swing, users can navigate various management tasks easily. Reporting features provide insights into hostel occupancy and revenue, while role-based access controls ensure secure management by administrators and staff. Utilizing a MySQL or SQLite database for data management, this system aims to streamline hostel operations and enhance administrative efficiency.

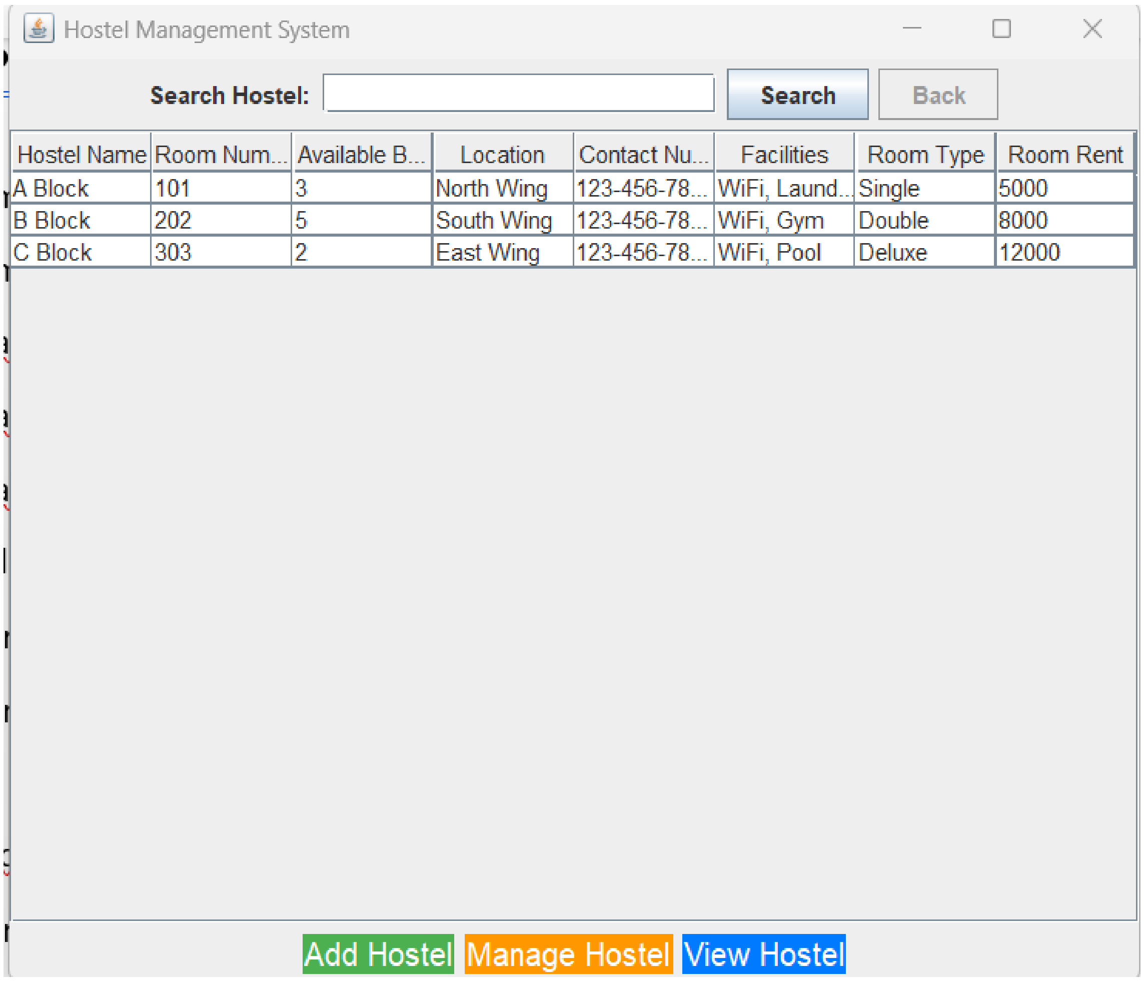
MySQL) and use Java Swing for a user-friendly interface. It will be a standalone desktop application, designed for small to medium-sized hotels, with potential for future enhancements.

JAVA UI PICTURES

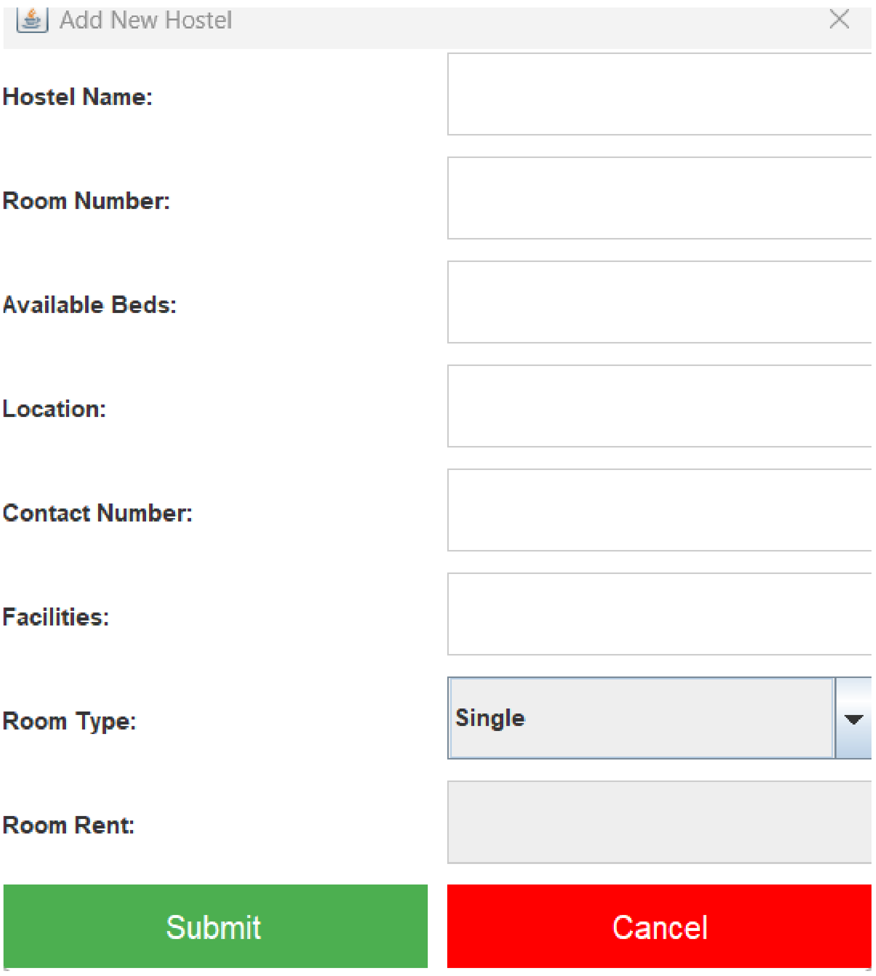




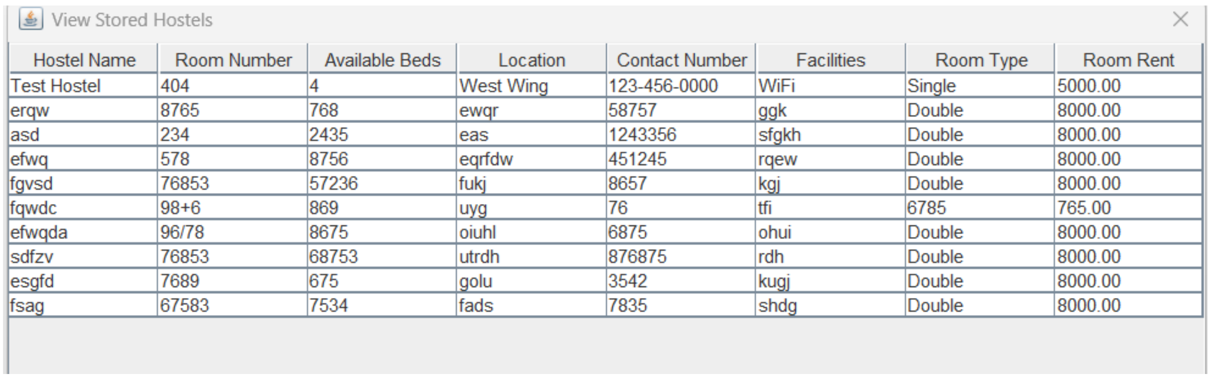
SEARCH OPTION:



Hostel Name	Room Num...	Available B...	Location	Contact Nu...	Facilities	Room Type	Room Rent
A Block	101	3	North Wing	123-456-78...	WiFi, Laund...	Single	5000
B Block	202	5	South Wing	123-456-78...	WiFi, Gym	Double	8000
C Block	303	2	East Wing	123-456-78...	WiFi, Pool	Deluxe	12000


Add New Hostel

Hostel Name:	
Room Number:	
Available Beds:	
Location:	
Contact Number:	
Facilities:	
Room Type:	Single
Room Rent:	


View Stored Hostels

Hostel Name	Room Number	Available Beds	Location	Contact Number	Facilities	Room Type	Room Rent
Test Hostel	404	4	West Wing	123-456-0000	WiFi	Single	5000.00
erqw	8765	768	ewqr	58757	ggk	Double	8000.00
asd	234	2435	eas	1243356	sfgkh	Double	8000.00
efwq	578	8756	eqrfdw	451245	rqew	Double	8000.00
fgvsd	76853	57236	fukj	8657	kgj	Double	8000.00
fqwdc	98+6	869	uyg	76	tfi	6785	765.00
efwqda	96/78	8675	oiuhl	6875	ohui	Double	8000.00
sdfzv	76853	68753	utrdh	876875	rdh	Double	8000.00
esgfd	7689	675	golu	3542	kugj	Double	8000.00
fsag	67583	7534	fads	7835	shdg	Double	8000.00

PROGRAM:

```
import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableRowSorter;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.*;



public class HostelManagementSystem1 extends JFrame {

    private DefaultTableModel tableModel;

    private JTextField searchField;

    private JTable hostelTable;

    private JButton backButton;



    public HostelManagementSystem1() {

        setTitle("Hostel Management System");
        setSize(600, 500);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLocationRelativeTo(null);

        BackgroundPanel panel = new BackgroundPanel();

        panel.setLayout(new BorderLayout());



        // Title label

        JLabel titleLabel = new JLabel("Hostel Management System",
JLabel.CENTER);

        titleLabel.setFont(new Font("Arial", Font.BOLD, 26));

        panel.add(titleLabel, BorderLayout.NORTH);



        // Search panel

        JPanel searchPanel = new JPanel();

        searchField = new JTextField(20);

        JButton searchButton = new JButton("Search");

        backButton = new JButton("Back");

        searchPanel.add(new JLabel("Search Hostel: "));
        searchPanel.add(searchField);

        searchPanel.add(searchButton);

        searchPanel.add(backButton);

        panel.add(searchPanel, BorderLayout.NORTH);



        // Table setup

        String[] columnNames = {"Hostel Name", "Room Number", "Available
Beds", "Location", "Contact", "Facilities", "Room Type", "Room Rent"};

        Object[][] data = {{"A Block", "101", "3", "North Wing", "123-456-7890",
"WiFi", "Single", "5000"}};

        tableModel = new DefaultTableModel(data, columnNames);

        hostelTable = new JTable(tableModel);

        JScrollPane tableScrollPane = new JScrollPane(hostelTable);

        panel.add(tableScrollPane, BorderLayout.CENTER);



        // Footer buttons

        JPanel footerPanel = new JPanel();

        footerPanel.setLayout(new FlowLayout());
```

```
    JButton addButton = new JButton("Add Hostel");

    JButton manageButton = new JButton("Manage Hostel");

    JButton viewButton = new JButton("View Hostel");

    footerPanel.add(addButton);

    footerPanel.add(manageButton);

    footerPanel.add(viewButton);

    panel.add(footerPanel, BorderLayout.SOUTH);



    add(panel);



    // Action Listeners

    addButton.addActionListener(e -> showAddHostelDialog());

    manageButton.addActionListener(e -> showAddHostelDialog());

    viewButton.addActionListener(e -> showViewHostelsDialog());

    searchButton.addActionListener(e -> filterHostels());

    backButton.addActionListener(e -> resetSearch());

}
```

```
    // Background Panel for custom image

    private class BackgroundPanel extends JPanel {

        private Image backgroundImage;

        public BackgroundPanel() {

            try {

                backgroundImage =
Toolkit.getDefaultToolkit().getImage("path/to/image.jpg");

                backgroundImage = backgroundImage.getScaledInstance(getWidth(),
getHeight(), Image.SCALE_SMOOTH);

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

        protected void paintComponent(Graphics g) {

            super.paintComponent(g);

            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

        }
```

```
}


// Add Hostel Dialog

private void showAddHostelDialog() {

    JDialog dialog = new JDialog(this, "Add New Hostel", true);

    dialog.setSize(450, 500);

    JPanel dialogPanel = new JPanel(new GridLayout(9, 2));

    dialogPanel.add(new JLabel("Hostel Name:"));

    JTextField nameField = new JTextField();

    dialogPanel.add(nameField);

    dialogPanel.add(new JLabel("Room Number:"));

    JTextField roomField = new JTextField();

    dialogPanel.add(roomField);

    dialogPanel.add(new JLabel("Available Beds:"));

    JTextField bedsField = new JTextField();

    dialogPanel.add(bedsField);

    dialogPanel.add(new JLabel("Location:"));
```

```
        JTextField locationField = new JTextField();

        dialogPanel.add(locationField);

        dialogPanel.add(new JLabel("Contact:"));

        JTextField contactField = new JTextField();

        dialogPanel.add(contactField);

        dialogPanel.add(new JLabel("Facilities:"));

        JTextField facilitiesField = new JTextField();

        dialogPanel.add(facilitiesField);

        JComboBox<String> roomTypeComboBox = new JComboBox<>(new
String[]{"Single", "Double", "Deluxe"});

        dialogPanel.add(new JLabel("Room Type:"));

        dialogPanel.add(roomTypeComboBox);

        dialogPanel.add(new JLabel("Room Rent:"));

        JTextField roomRentField = new JTextField();

        roomRentField.setEditable(false);

        dialogPanel.add(roomRentField);

        roomTypeComboBox.addActionListener(e -> {

            String rent = switch ((String) roomTypeComboBox.getSelectedItem()) {
```

```
            case "Single" -> "5000";

            case "Double" -> "8000";

            default -> "12000";

        };

        roomRentField.setText(rent);

    });

    dialogPanel.add(new JButton("Submit"));

    dialogPanel.add(new JButton("Cancel"));

    dialog.add(dialogPanel);

    dialog.setVisible(true);

}



// View Hostels Dialog

private void showViewHostelsDialog() {

    JDialog dialog = new JDialog(this, "View Stored Hostels", true);

    dialog.setSize(800, 400);

    DefaultTableModel model = new DefaultTableModel();
```

```
        }

        model.addColumn("Hostel Name");

        model.addColumn("Room Number");

        model.addColumn("Available Beds");

        model.addColumn("Location");

        model.addColumn("Contact Number");

        model.addColumn("Facilities");

        model.addColumn("Room Type");

        model.addColumn("Room Rent");

        try (Connection conn = connectToDatabase(); Statement stmt =
conn.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT * FROM hostels")) {

            while (rs.next()) {

                model.addRow(new Object[]{rs.getString("hostel_name"),
rs.getString("room_number"),

                        rs.getString("available_beds"), rs.getString("location"),
rs.getString("contact_number"),

                        rs.getString("facilities"), rs.getString("room_type"),
rs.getString("room_rent")});

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        JTable table = new JTable(model);
        dialog.add(new JScrollPane(table));
        dialog.setVisible(true);
    }

    // Database connection
    private Connection connectToDatabase() throws SQLException {
        return
DriverManager.getConnection("jdbc:mysql://localhost:3306/hostel_manageme
nt", "root", "password");
    }


    // Filter Hostels
    private void filterHostels() {
        TableRowSorter<DefaultTableModel> sorter = new
TableRowSorter<>(tableModel);
```

```
    hostelTable.setRowSorter(sorter);

    String query = searchField.getText();

    if (query.length() == 0) sorter.setRowFilter(null);

    else sorter.setRowFilter(RowFilter.regexFilter("(?i)" + query));

}



// Reset Search

private void resetSearch() {

    searchField.setText("");

    filterHostels();

}



public static void main(String[] args) {

    SwingUtilities.invokeLater(() -> {

        HostelManagementSystem1 app = new HostelManagementSystem1();

        app.setVisible(true);

    });
```

```
  }

}
```

CONCLUSION:

The **Hostel Management System** developed with Java Swing provides an efficient solution for managing hostel operations. It allows administrators to easily add, manage, and view hostel details, including room assignments, available beds, and contact information. Integrated with a MySQL database, the system enables seamless data storage and retrieval. Features like search and filtering enhance usability, while dynamic room rent calculations provide flexibility.

This system streamlines tasks such as room allocation and student registration, improving efficiency and reducing manual errors. With a user-friendly interface and modular design, the system can be easily expanded to include additional features in the future. Overall, it simplifies hostel management, improves organization, and enhances the user experience for both staff and students.

REFERENCE:

https://www.google.com/