Login Page:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


public class LoginPage {


    public static void main(String[] args) {

        // Create frame (window) for the login page

        JFrame frame = new JFrame("Login Page");

        frame.setSize(400, 300);  // Set window size

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setLocationRelativeTo(null);  // Center the window on the screen


        // Set up the background color of the frame

        frame.getContentPane().setBackground(new Color(255, 204, 204)); // Light Pink
background

        frame.setLayout(new BorderLayout());


        // Create a panel for the login form

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 2, 10, 10)); // Grid layout with 4 rows and 2 columns

        panel.setBackground(new Color(255, 204, 204));


        // Add heading (Login) at the top
        JLabel headingLabel = new JLabel("Login", SwingConstants.CENTER);
        headingLabel.setFont(new Font("Arial", Font.BOLD, 30)); // Bold and large font for
heading
        headingLabel.setForeground(new Color(255, 0, 102));  // Vibrant pink color
        headingLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0)); // Add
padding around the heading

        // Create labels and text fields
        JLabel usernameLabel = new JLabel("USER NAME :");
        usernameLabel.setFont(new Font("Arial", Font.BOLD, 20));
        JTextField usernameField = new JTextField(20);
        usernameField.setBackground(new Color(255, 255, 255)); // White background for the
input field
        usernameField.setForeground(Color.BLACK); // Black text color for username input
field
        usernameField.setFont(new Font("Times New Roman", Font.BOLD, 15)); // Very bold
font for text fields

        // Set the label for Username to Black and Bold
        usernameLabel.setForeground(Color.BLACK);  // Set label color to black

        JLabel passwordLabel = new JLabel("PASSWORD :");
        JPasswordField passwordField = new JPasswordField(20);
        passwordField.setBackground(new Color(255, 255, 255)); // White background for the
password field
        passwordField.setForeground(Color.BLACK); // Black text color for password input field
        passwordField.setFont(new Font("Arial", Font.BOLD, 16)); // Very bold font for password
field
```

```
        // Set the label for Password to Black and Bold
        passwordLabel.setForeground(Color.BLACK);  // Set label color to black
        passwordLabel.setFont(new Font("Times New Roman", Font.BOLD, 20)); // Make the
label bold

        // Add labels and text fields to panel
        panel.add(usernameLabel);
        panel.add(usernameField);
        panel.add(passwordLabel);
        panel.add(passwordField);

        // Create Submit and Cancel buttons
        JButton submitButton = new JButton("Submit");
        submitButton.setBackground(new Color(255, 102, 102)); // Light red button
        submitButton.setForeground(Color.WHITE);
        submitButton.setFont(new Font("Arial", Font.BOLD, 14)); // Bold font for buttons

        JButton cancelButton = new JButton("Cancel");
        cancelButton.setBackground(new Color(255, 204, 0)); // Vibrant yellow button
        cancelButton.setForeground(Color.WHITE);
        cancelButton.setFont(new Font("Arial", Font.BOLD, 14)); // Bold font for buttons

        // Add buttons to panel
        panel.add(submitButton);
        panel.add(cancelButton);
        });


        // Add components to the frame
        frame.add(headingLabel, BorderLayout.NORTH);
        frame.add(panel, BorderLayout.CENTER);

        // Button actions
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                // Check login credentials
                if (username.equals("admin") && password.equals("password")) {
                    // Hide the login frame
                    frame.dispose();  // Close the login page

                    // Open the HostelManagementSystem1 frame
                    HostelManagementSystem1 hostelSystem = new HostelManagementSystem1();
// Create an instance
                    hostelSystem.setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(frame, "Invalid Username or Password", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        cancelButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Clear the fields on cancel
                usernameField.setText("");
                passwordField.setText("");
            }
        });

        // Make the window visible
        frame.setVisible(true);
    }
}
```

Hostel Management System:

```
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.*;
import java.util.ArrayList;

public class HostelManagementSystem1 extends JFrame {
    private DefaultTableModel tableModel;
    private JTextField searchField;

    private JTable hostelTable;

    private JButton backButton;


    private String backgroundImagePath = "path/to/your/background/image.jpg"; // Update
this path


    public HostelManagementSystem1() {

        setTitle("Hostel Management System");

        setSize(600, 500); // Adjusted size for additional fields

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLocationRelativeTo(null);


        // Panel for hostel management system UI with custom background

        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));

        titleLabel.setForeground(new Color(0, 51, 102)); // Darker blue for title


        // Adding title to the top

        panel.add(titleLabel, BorderLayout.NORTH);


        // Search Panel
        JPanel searchPanel = new JPanel();
        searchPanel.setOpaque(false); // Make the search panel transparent
        searchField = new JTextField(20); // Search field with a specified number of columns
        JButton searchButton = new JButton("Search");

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

        // Adding back button to search panel
        backButton = new JButton("Back");
        backButton.setEnabled(false); // Initially disable back button
        searchPanel.add(backButton); // Add the back button to the panel

        panel.add(searchPanel, BorderLayout.NORTH);

        // Table model to manage hostel data
        String[] columnNames = {"Hostel Name", "Room Number", "Available Beds", "Location",
"Contact Number", "Facilities", "Room Type", "Room Rent"};
        Object[][] data = {
            {"A Block", "101", "3", "North Wing", "123-456-7890", "WiFi, Laundry", "Single",
"5000"},
            {"B Block", "202", "5", "South Wing", "123-456-7891", "WiFi, Gym", "Double", "8000"},
            {"C Block", "303", "2", "East Wing", "123-456-7892", "WiFi, Pool", "Deluxe", "12000"}
        };

        // DefaultTableModel for dynamic table updates
```

```
tableModel = new DefaultTableModel(data, columnNames);

hostelTable = new JTable(tableModel);

JScrollPane tableScrollPane = new JScrollPane(hostelTable);

panel.add(tableScrollPane, BorderLayout.CENTER);


// Footer with buttons for adding and managing hostels

JPanel footerPanel = new JPanel();

footerPanel.setLayout(new FlowLayout());

footerPanel.setOpaque(false); // Make the footer panel transparent


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

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

JButton viewButton = new JButton("View Hostel"); // New View Hostel button


// Set button colors and font

styleButton(addButton, new Color(76, 175, 80)); // Vibrant green

styleButton(manageButton, new Color(255, 152, 0)); // Vibrant orange

styleButton(viewButton, new Color(0, 123, 255)); // Blue for view button


footerPanel.add(addButton);

footerPanel.add(manageButton);

footerPanel.add(viewButton); // Add the view button to the footer


panel.add(footerPanel, BorderLayout.SOUTH);


// Add panel to frame
```

```
        add(panel);

        // Add action listeners for buttons
        addButton.addActionListener(e -> showAddHostelDialog());
        manageButton.addActionListener(e -> showAddHostelDialog());
        viewButton.addActionListener(e -> showViewHostelsDialog()); // Action for View Hostel
button

        // Add action listener for the search button
        searchButton.addActionListener(e -> filterHostels());

        // Add action listener for the back button
        backButton.addActionListener(e -> resetSearch());
    }

    // Custom JPanel class to handle background image painting
    private class BackgroundPanel extends JPanel {
        private Image backgroundImage;

        public BackgroundPanel() {
            try {
                backgroundImage = Toolkit.getDefaultToolkit().getImage(backgroundImagePath);
                // Optionally scale the image to fit the panel
                backgroundImage = backgroundImage.getScaledInstance(getWidth(), getHeight(),
Image.SCALE_SMOOTH);
            } catch (Exception e) {
                e.printStackTrace();
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        // Draw the background image
        g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
    }
}

// Method to show a dialog for adding a new hostel
private void showAddHostelDialog() {
    JDialog dialog = new JDialog(this, "Add New Hostel", true);
    dialog.setSize(450, 500);
    dialog.setLocationRelativeTo(this);

    JPanel dialogPanel = new JPanel();
    dialogPanel.setLayout(new GridLayout(9, 2, 10, 10));
    dialogPanel.setBackground(Color.WHITE); // Set background to white for contrast

    // Labels and text fields for hostel information
    JLabel nameLabel = new JLabel("Hostel Name:");
    JLabel roomLabel = new JLabel("Room Number:");
    JLabel bedsLabel = new JLabel("Available Beds:");
    JLabel locationLabel = new JLabel("Location:");
JLabel contactLabel = new JLabel("Contact Number:");

JLabel facilitiesLabel = new JLabel("Facilities:");

JLabel roomTypeLabel = new JLabel("Room Type:");

JLabel roomRentLabel = new JLabel("Room Rent:");


JTextField nameField = new JTextField();

JTextField roomField = new JTextField();

JTextField bedsField = new JTextField();

JTextField locationField = new JTextField();

JTextField contactField = new JTextField();

JTextField facilitiesField = new JTextField();


// Room type selection

String[] roomTypes = {"Single", "Double", "Deluxe"};

JComboBox<String> roomTypeComboBox = new JComboBox<>(roomTypes);

JTextField roomRentField = new JTextField();

roomRentField.setEditable(false); // Make rent field read-only


// Add an action listener to calculate room rent based on selected room type

roomTypeComboBox.addActionListener(new ActionListener() {

  @Override

  public void actionPerformed(ActionEvent e) {

    String selectedType = (String) roomTypeComboBox.getSelectedItem();

    int rent = 0;


    switch (selectedType) {
            case "Single":
               rent = 5000; // Rent for single room
               break;
            case "Double":
               rent = 8000; // Rent for double room
               break;
            case "Deluxe":
               rent = 12000; // Rent for deluxe room
               break;
         }
         roomRentField.setText(String.valueOf(rent)); // Set calculated rent
      }
   });

   // Style text fields
   styleTextField(nameField);
   styleTextField(roomField);
   styleTextField(bedsField);
   styleTextField(locationField);
   styleTextField(contactField);
   styleTextField(facilitiesField);
   styleTextField(roomRentField); // Style for read-only field

   // Buttons for Submit and Cancel
   JButton submitButton = new JButton("Submit");
   JButton cancelButton = new JButton("Cancel");
```

```
// Style buttons
styleButton(submitButton, new Color(76, 175, 80)); // Vibrant green
styleButton(cancelButton, Color.RED); // Vibrant red for cancel

// Add components to the dialog panel
dialogPanel.add(nameLabel);
dialogPanel.add(nameField);
dialogPanel.add(roomLabel);
dialogPanel.add(roomField);
dialogPanel.add(bedsLabel);
dialogPanel.add(bedsField);
dialogPanel.add(locationLabel);
dialogPanel.add(locationField);
dialogPanel.add(contactLabel);
dialogPanel.add(contactField);
dialogPanel.add(facilitiesLabel);
dialogPanel.add(facilitiesField);
dialogPanel.add(roomTypeLabel);
dialogPanel.add(roomTypeComboBox);
dialogPanel.add(roomRentLabel);
dialogPanel.add(roomRentField);
dialogPanel.add(submitButton);
dialogPanel.add(cancelButton);

// Add the panel to the dialog
```

```
        dialog.add(dialogPanel);


        // Submit button action
        submitButton.addActionListener(e -> {
          String name = nameField.getText();
          String roomNumber = roomField.getText();
          String availableBeds = bedsField.getText();
          String location = locationField.getText();
          String contactNumber = contactField.getText();
          String facilities = facilitiesField.getText();
          String roomType = (String) roomTypeComboBox.getSelectedItem();
          String roomRent = roomRentField.getText();


          // Validate input fields
          if (name.isEmpty() || roomNumber.isEmpty() || availableBeds.isEmpty() ||
location.isEmpty() ||
                contactNumber.isEmpty() || facilities.isEmpty() || roomType.isEmpty() ||
roomRent.isEmpty()) {
            JOptionPane.showMessageDialog(dialog, "All fields must be filled!", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
          }


          // Save hostel information to database (you can call the database method here)
          try {
            saveHostelToDatabase(name, roomNumber, availableBeds, location,
contactNumber, facilities, roomType, roomRent);
```

```
            JOptionPane.showMessageDialog(dialog, "Hostel added successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
            dialog.dispose(); // Close dialog after submission
          } catch (SQLException ex) {
            JOptionPane.showMessageDialog(dialog, "Error saving data: " + ex.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
          }
        });

        // Action for Cancel button
        cancelButton.addActionListener(e -> dialog.dispose());

        // Show the dialog
        dialog.setVisible(true);
    }

    // Method to save hostel to the database
    private void saveHostelToDatabase(String name, String roomNumber, String
availableBeds, String location,
                                 String contactNumber, String facilities, String roomType, String
roomRent) throws SQLException {
        Connection conn = connectToDatabase();
        String query = "INSERT INTO hostels (hostel_name, room_number, available_beds,
location, contact_number, facilities, room_type, room_rent) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, name);
        pstmt.setString(2, roomNumber);
        pstmt.setString(3, availableBeds);
        pstmt.setString(4, location);
        pstmt.setString(5, contactNumber);
        pstmt.setString(6, facilities);
        pstmt.setString(7, roomType);
        pstmt.setString(8, roomRent);
        pstmt.executeUpdate();
        conn.close();
    }

    // Method to connect to the database
    private Connection connectToDatabase() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/hostel_management";
        String user = "root";
        String password = "Saravanan123#";
        return DriverManager.getConnection(url, user, password);
    }
    private void showViewHostelsDialog() {
        JDialog dialog = new JDialog(this, "View Stored Hostels", true);
        dialog.setSize(800, 400);
        dialog.setLocationRelativeTo(this);

        // Create a new table model
        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");


// Fetch data from the database and populate the table

try (Connection conn = connectToDatabase();

    Statement stmt = conn.createStatement();

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


   // Clear previous data from the model

   model.setRowCount(0);  // Clears the existing rows in the table


   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) {
            JOptionPane.showMessageDialog(dialog, "Error fetching data: " + e.getMessage(),
"Database Error", JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();  // Print stack trace for debugging
        }

        // JTable to display hostel data
        JTable table = new JTable(model);
        JScrollPane scrollPane = new JScrollPane(table);
        dialog.add(scrollPane);

        // Show the dialog
        dialog.setVisible(true);
    }



    // Method to display the stored hostels in a dialog
    /*private void showViewHostelsDialog() {
        JDialog dialog = new JDialog(this, "View Stored Hostels", true);
        dialog.setSize(800, 400);
        dialog.setLocationRelativeTo(this);

        // Create a new table model
        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");


// Fetch data from the database and populate the table

try (Connection conn = connectToDatabase();

    Statement stmt = conn.createStatement();

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


   while (rs.next()) {

      model.addRow(new Object[]{

            rs.getString("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 to display hostel data
        JTable table = new JTable(model);
        JScrollPane scrollPane = new JScrollPane(table);
        dialog.add(scrollPane);

        // Show the dialog
        dialog.setVisible(true);
    }*/

    // Method to filter hostels based on search text
    private void filterHostels() {
        String query = searchField.getText();
        TableRowSorter<DefaultTableModel> sorter = new TableRowSorter<>(tableModel);
        hostelTable.setRowSorter(sorter);
        if (query.length() == 0) {
            sorter.setRowFilter(null);
        } else {
            sorter.setRowFilter(RowFilter.regexFilter("(?i)" + query)); // (?i) for case-insensitive
search
        }
    }
```

```
// Method to reset the search filter
private void resetSearch() {
    searchField.setText("");
    filterHostels();
}

// Method to style the buttons
private void styleButton(JButton button, Color color) {
    button.setFont(new Font("Arial", Font.PLAIN, 16));
    button.setBackground(color);
    button.setForeground(Color.WHITE);
    button.setBorder(BorderFactory.createLineBorder(color));
}

// Method to style the text fields
private void styleTextField(JTextField textField) {
    textField.setFont(new Font("Arial", Font.PLAIN, 16));
    textField.setBorder(BorderFactory.createLineBorder(new Color(204, 204, 204)));
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        HostelManagementSystem1 app = new HostelManagementSystem1();
        app.setVisible(true);
    });
}
```

```
}
```