**MINI PROJECT REPORT**

**ON**

**HOTEL MANAGEMENT SYSTEM USING**

**USER INTERFACE(UI) IN JAVA**

Submitted by

**CR ROYCE RAPHAEL**

**NC.AI.U4AID24051**

For

23AID111- Object Oriented Programming

II Semester

B.Tech. AID

**School of Computing**

**Amrita Vishwa Vidyapeetham, Nagercoil**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **Chapter Name** | **Page No.** |
|  | Introduction | 3 & 4 |
|  | Problem Statement | 5 |
|  | Objectives | 6 |
|  | UML diagrams of the project   1. Class Diagram 2. Object Diagram 3. Use-Case Diagram 4. Activity Diagram | 7,8,9&10 |
|  | Modules of project | 11&12 |
|  | Code | 13,14&15 |
|  | Output Screenshots | 16,17,18&19 |
|  | Application of the project | 20,21&22 |
|  | Limitations of the project | 23&24 |
|  | Bibliography | 25 |
|  | GitHub link of the project | 26 |

**Introduction**

The **Hotel Management System** developed using Java Swing is a basic desktop application aimed at managing hotel guest details efficiently. The system enables hotel staff to sign up, log in, and handle day-to-day operations related to guest registration and checkout. Built with user-friendliness and simplicity in mind, this application acts as a foundational software solution for small hotels or educational demonstrations.

The project makes use of core Java concepts such as object-oriented programming, GUI (Graphical User Interface) development through Swing, and basic data handling using in-memory lists. The main goal is to manage guest information such as their names and assigned room numbers, and to facilitate adding, viewing, and removing guests from the system.

This project consists of two primary components:

1. **Sign-Up Page** – A form-based interface where a new user can register by providing their name, email, and password.
2. **Hotel Management Dashboard** – The main panel that allows the user to add new guests, view current guest records, and perform checkouts.

This application is ideal for learning and demonstrating how to integrate GUI design with back-end logic in Java, as well as for showcasing how small-scale systems can be built without the use of databases.

The Hotel Management System is equipped with the following features:

1. Sign-up system

* When the application launches, it first prompts the user with a Sign-Up form.
* The form requires input for the user’s name, email, and password.
* If all fields are correctly filled, the system notifies the user of a successful sign-up and proceeds to open the hotel dashboard.

1. Guest Registration

* The user can input a guest’s name and assign a room number.
* The system checks if both fields are filled and if the room number is a valid integer.
* Once verified, the guest is added to the system, and the information is displayed on-screen for confirmation.

1. View Guest List

* With a single click, the system displays all currently registered guests.
* Each guest entry includes their name and room number.

1. Guest Checkout

* The user can input a guest name to check them out of the hotel.
* The system performs a case-insensitive search and removes the guest if found.
* Confirmation is displayed whether the checkout is successful or if the guest could not be found.

1. GUI Design

* Uses Swing components like JTextField, JLabel, JButton, JTextArea, and JScrollPane.
* Layouts are managed using FlowLayout for simplicity and clarity.

This feature-rich interface provides a seamless and smooth experience for hotel staff with minimal training required.

The Hotel Management System uses Java Swing for its front-end GUI and Java Collections for back-end data storage. The application runs on the JVM and does not depend on any external libraries or databases, making it highly portable and easy to run on any machine with Java installed.

**Class Structure:**

* **Guest Class**: A simple data model class representing each guest with fields for name and room number.
* **SignUpPage Class**: Handles user registration and transitions to the main dashboard.
* **HotelManagementSystem Class**: The main application window where the core functionalities are implemented.

This structure promotes modularity and separates concerns between different parts of the application. Each functionality is encapsulated in a clearly defined class, making the system easy to understand and extend.

**Conclusion:**

This Hotel Management System is an excellent starting point for developers new to Java GUI programming. It demonstrates how real-world problems like hotel guest tracking can be addressed through simple and well-structured code. While the application is currently limited to in-memory data, it lays a strong foundation for further enhancements such as integrating a database, adding authentication, generating reports, and implementing search filters. It reflects both practical programming and user-centered design principles, making it suitable for academic and training purposes.

**Problem Statement**

In the hospitality industry, managing guest information efficiently and accurately is crucial for delivering quality service and ensuring smooth operations. Traditional methods of maintaining guest records using paper-based logs or spreadsheets are not only time-consuming but also prone to errors, mismanagement, and data loss. These outdated practices can lead to difficulties in tracking room assignments, monitoring check-ins and checkouts, and generating guest reports, especially in small to mid-sized hotels that may not have access to sophisticated hotel management software.

The core problem lies in the **lack of a lightweight, easy-to-use, and cost-effective system** that enables hotel staff to manage guest details without the need for advanced technical knowledge or heavy infrastructure. While large hotels often use commercial software systems, smaller establishments often operate without any digital support, relying on manual entries that hamper efficiency and limit scalability.

To address this issue, there is a need for a **simple desktop-based Hotel Management System** that allows hotel employees to:

* Register themselves into the system securely.
* Add new guests and assign them room numbers.
* View a list of all current guests in the hotel.
* Process checkouts by removing guests from the system.
* Display updates and notifications via a user-friendly interface.

The goal is to provide a **Java-based GUI application** that utilizes core programming concepts and an intuitive graphical interface to simulate basic hotel management operations. The application should maintain an internal record of guest data during runtime and offer basic functionalities to perform day-to-day hotel tasks.

This project aims to demonstrate how even a beginner-level software application can provide valuable automation for hotel guest tracking, laying the groundwork for future upgrades such as database integration, real-time data synchronization, and role-based access control. It is designed to be educational, practical, and extendable, serving both as a learning tool and a foundational solution for small-scale hospitality businesses.

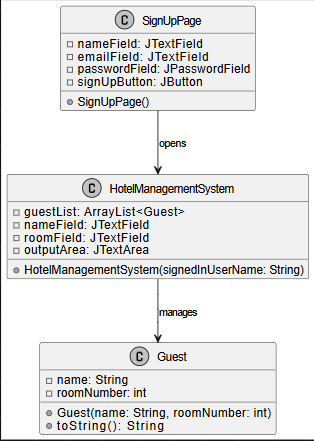
**Objectives**

The main objective of the Hotel Management System project is to design and develop a basic yet functional software application that streamlines the process of managing guest information in a hotel environment. The specific objectives of the project are as follows:

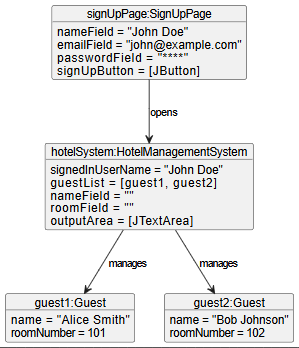
1. **To Develop a User-Friendly GUI Application**  
   Create an intuitive and easy-to-navigate interface using Java Swing that allows users with minimal technical knowledge to perform hotel operations efficiently.
2. **To Enable Secure User Sign-Up**  
   Implement a simple registration system to allow hotel staff to sign up with their name, email, and password before accessing the main application.
3. **To Manage Guest Information Dynamically**  
   Allow users to add new guests with names and room numbers, store the data temporarily in memory, and display the guest list dynamically as it is updated.
4. **To Facilitate Guest Check-In and Check-Out**  
   Provide functionality to add new guest entries (check-in) and remove guests based on their name (check-out), making the system mimic real-world hotel operations.
5. **To Maintain a Real-Time Display of Guest Data**  
   Use a text area to show the current list of guests in real-time, giving the user constant feedback on the state of the hotel’s guest registry.
6. **To Demonstrate Core Java Concepts**  
   Utilize object-oriented programming principles, event-driven design, and Swing components to demonstrate effective Java programming techniques.
7. **To Serve as a Base for Future Enhancements**  
   Build the application in a modular and extensible way so that it can be further improved by integrating features like database support, login systems, search functionality, or billing modules.
8. **To Provide an Educational Tool for Beginners**  
   Design the system with simplicity and clarity in mind, making it suitable for educational purposes and for teaching the basics of software development using Java.

UML Diagram

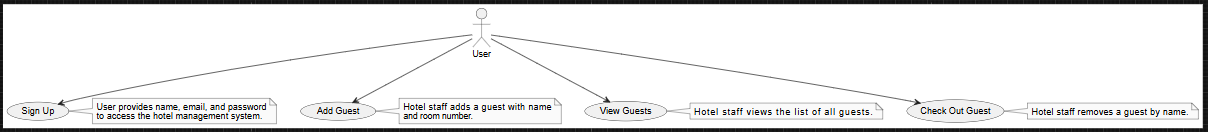
1. Class Diagram



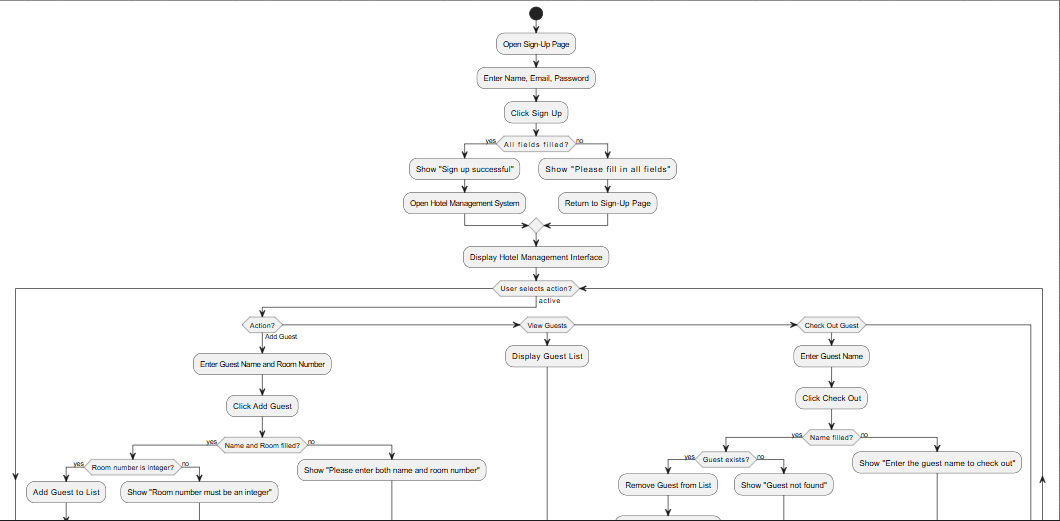
1. Object Diagram

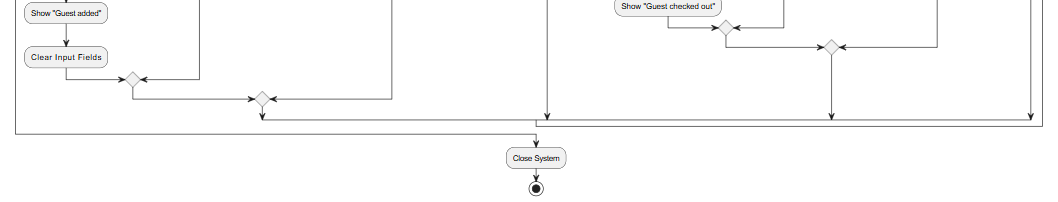


1. Use – Case Diagram



1. Activity Diagram





**Modules**

The Hotel Management System is divided into several functional modules, each responsible for handling specific tasks within the application. These modules work together to deliver a complete and seamless hotel management experience.

* 1. **User Registration Model(Sign-up Module)**

**Purpose:**  
To register new users (e.g., hotel staff) before granting them access to the main system.

**Key Features:**

* Input fields for Name, Email, and Password
* Basic input validation to ensure all fields are filled
* Confirmation message upon successful registration
* Transition to the main hotel dashboard
  1. **Guest Management Module**

**Purpose:**  
To handle all operations related to managing guest information.

**Key Features:**

* Input fields to enter guest name and room number
* Ability to add a new guest to the internal guest list
* Data validation (e.g., ensuring the room number is an integer)
* Display of success or error messages
  1. **View Guest List Module**

**Purpose:**  
To provide a real-time display of all guests currently registered in the system.

**Key Features:**

* Button to trigger the display of the full guest list
* Output area showing names and room numbers of all guests
* Clear and readable formatting of guest data
  1. **Check – out Module**

**Purpose:**  
To allow hotel staff to remove a guest from the system once they check out.

**Key Features:**

* Input field to enter the name of the guest to check out
* Case-insensitive search for guest names
* Automatic removal of guest from the list
* Notification of success or error if guest is not found
  1. **User Interface(UI) Module**

**Purpose:**  
To define the overall look and layout of the application.

**Key Features:**

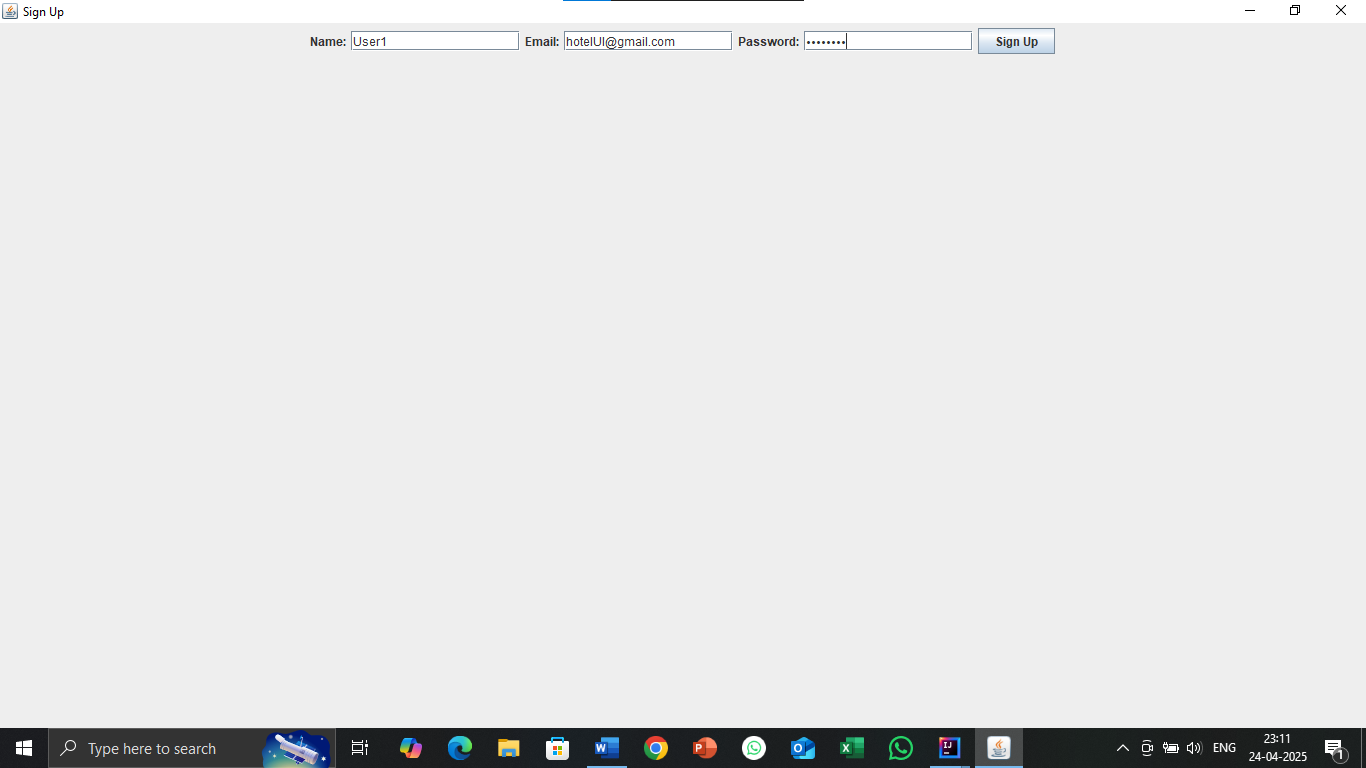
* Use of Java Swing components (JFrame, JLabel, JTextField, JButton, JTextArea, JScrollPane)
* Organized layout using FlowLayout for simplicity
* Real-time updates in the interface without needing to restart the application

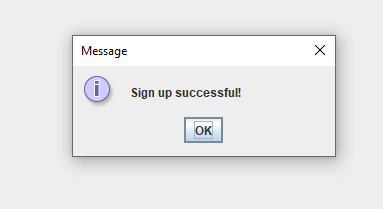
**Code**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.ArrayList;  
  
// --- Guest Class ---  
class Guest {  
 String name;  
 int roomNumber;  
  
 Guest(String name, int roomNumber) {  
 this.name = name;  
 this.roomNumber = roomNumber;  
 }  
  
 public String toString() {  
 return "Name: " + name + ", Room: " + roomNumber;  
 }  
}  
  
// --- Sign-Up Frame ---  
class SignUpPage extends JFrame {  
 private JTextField nameField = new JTextField(15);  
 private JTextField emailField = new JTextField(15);  
 private JPasswordField passwordField = new JPasswordField(15);  
 private JButton signUpButton = new JButton("Sign Up");  
  
 SignUpPage() {  
 setTitle("Sign Up");  
 setSize(350, 250);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(new FlowLayout());  
  
 add(new JLabel("Name:"));  
 add(nameField);  
 add(new JLabel("Email:"));  
 add(emailField);  
 add(new JLabel("Password:"));  
 add(passwordField);  
 add(signUpButton);  
  
 signUpButton.addActionListener(e -> {  
 String name = nameField.getText();  
 String email = emailField.getText();  
 String password = new String(passwordField.getPassword());  
  
 if (!name.isEmpty() && !email.isEmpty() && !password.isEmpty()) {  
 // Sign up success  
 JOptionPane.*showMessageDialog*(this, "Sign up successful!");  
 this.dispose(); // close the sign-up window  
 new HotelManagementSystem(name).setVisible(true); // open main system  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Please fill in all fields.");  
 }  
 });  
 }  
}  
  
// --- Hotel Management Frame ---  
public class HotelManagementSystem extends JFrame {  
 private ArrayList<Guest> guestList = new ArrayList<>();  
 private JTextField nameField = new JTextField(15);  
 private JTextField roomField = new JTextField(5);  
 private JTextArea outputArea = new JTextArea(10, 30);  
  
 public HotelManagementSystem(String signedInUserName) {  
 setTitle("Hotel Management System - Welcome " + signedInUserName);  
 setSize(500, 400);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(new FlowLayout());  
  
 // Input Panel  
 JPanel inputPanel = new JPanel();  
 inputPanel.add(new JLabel("Guest Name:"));  
 inputPanel.add(nameField);  
 inputPanel.add(new JLabel("Room #:"));  
 inputPanel.add(roomField);  
  
 JButton addButton = new JButton("Add Guest");  
 JButton viewButton = new JButton("View Guests");  
 JButton checkoutButton = new JButton("Check Out");  
  
 inputPanel.add(addButton);  
 inputPanel.add(viewButton);  
 inputPanel.add(checkoutButton);  
 add(inputPanel);  
  
 // Output Area  
 outputArea.setEditable(false);  
 add(new JScrollPane(outputArea));  
  
 // Add Guest Action  
 addButton.addActionListener(e -> {  
 String name = nameField.getText();  
 String roomStr = roomField.getText();  
 if (!name.isEmpty() && !roomStr.isEmpty()) {  
 try {  
 int room = Integer.*parseInt*(roomStr);  
 guestList.add(new Guest(name, room));  
 outputArea.setText("Guest added: " + name + " in room " + room);  
 nameField.setText("");  
 roomField.setText("");  
 } catch (NumberFormatException ex) {  
 outputArea.setText("Room number must be an integer.");  
 }  
 } else {  
 outputArea.setText("Please enter both name and room number.");  
 }  
 });  
  
 // View Guests Action  
 viewButton.addActionListener(e -> {  
 outputArea.setText("Guest List:\n");  
 for (Guest g : guestList) {  
 outputArea.append(g + "\n");  
 }  
 });  
  
 // Check Out Action  
 checkoutButton.addActionListener(e -> {  
 String name = nameField.getText();  
 if (!name.isEmpty()) {  
 boolean removed = guestList.removeIf(g -> g.name.equalsIgnoreCase(name));  
 if (removed) {  
 outputArea.setText("Guest checked out: " + name);  
 } else {  
 outputArea.setText("Guest not found: " + name);  
 }  
 } else {  
 outputArea.setText("Enter the guest name to check out.");  
 }  
 });  
 }  
  
 // Entry Point  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 SignUpPage signUp = new SignUpPage();  
 signUp.setVisible(true);  
 });  
 }  
}

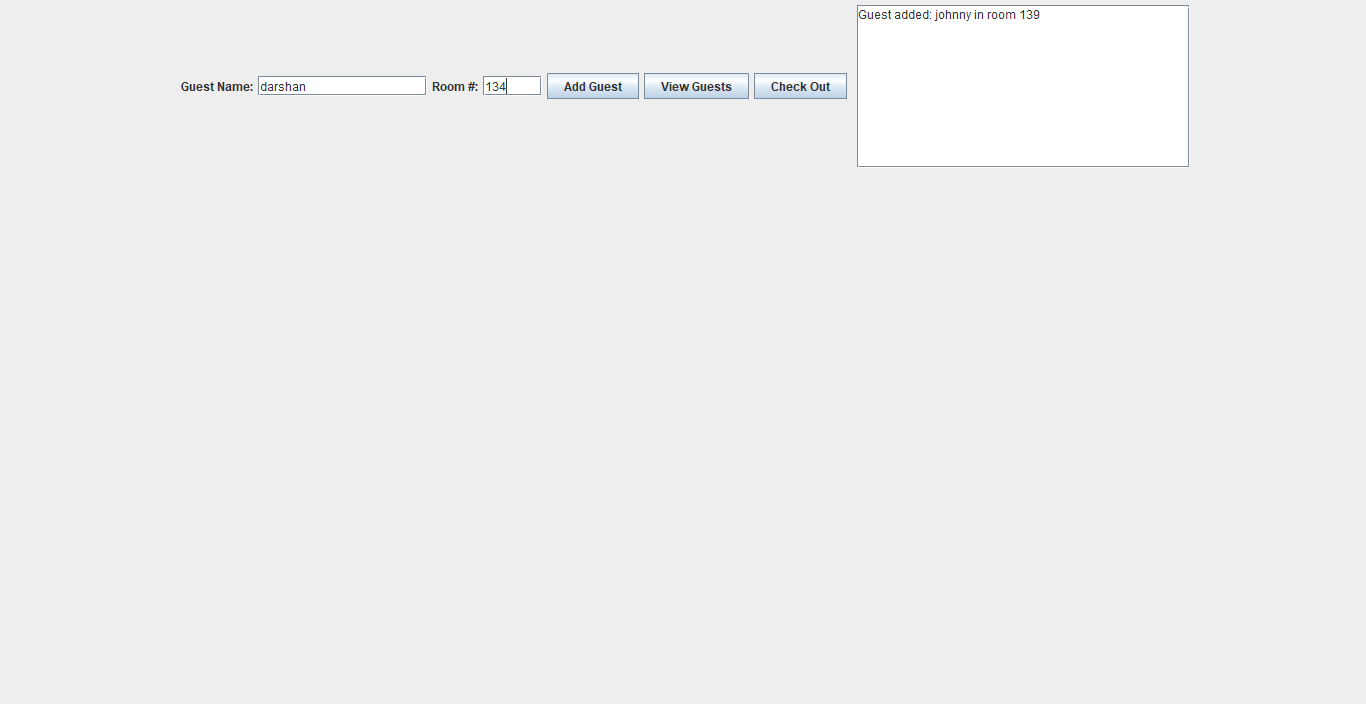
**Output Screenshots**

**Sign-up**

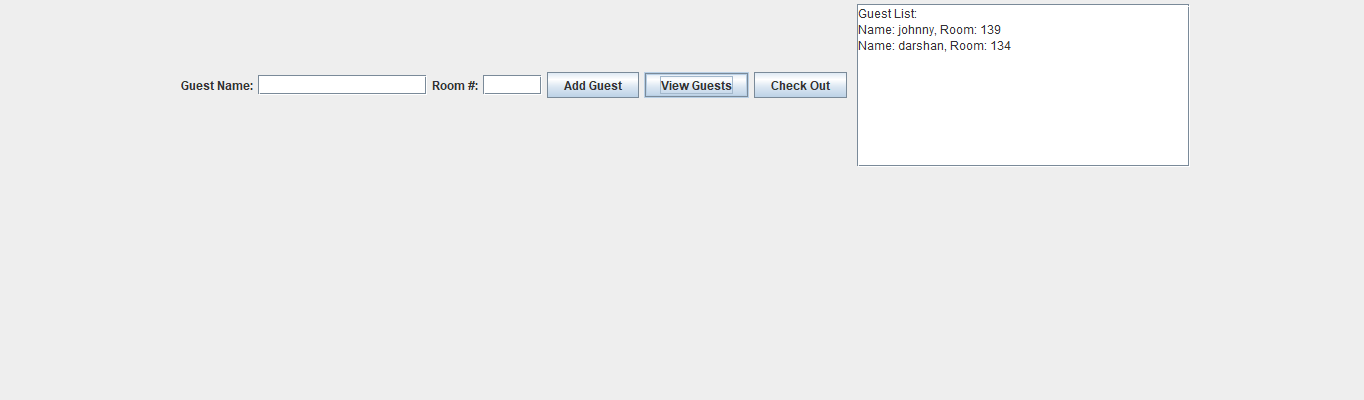
****

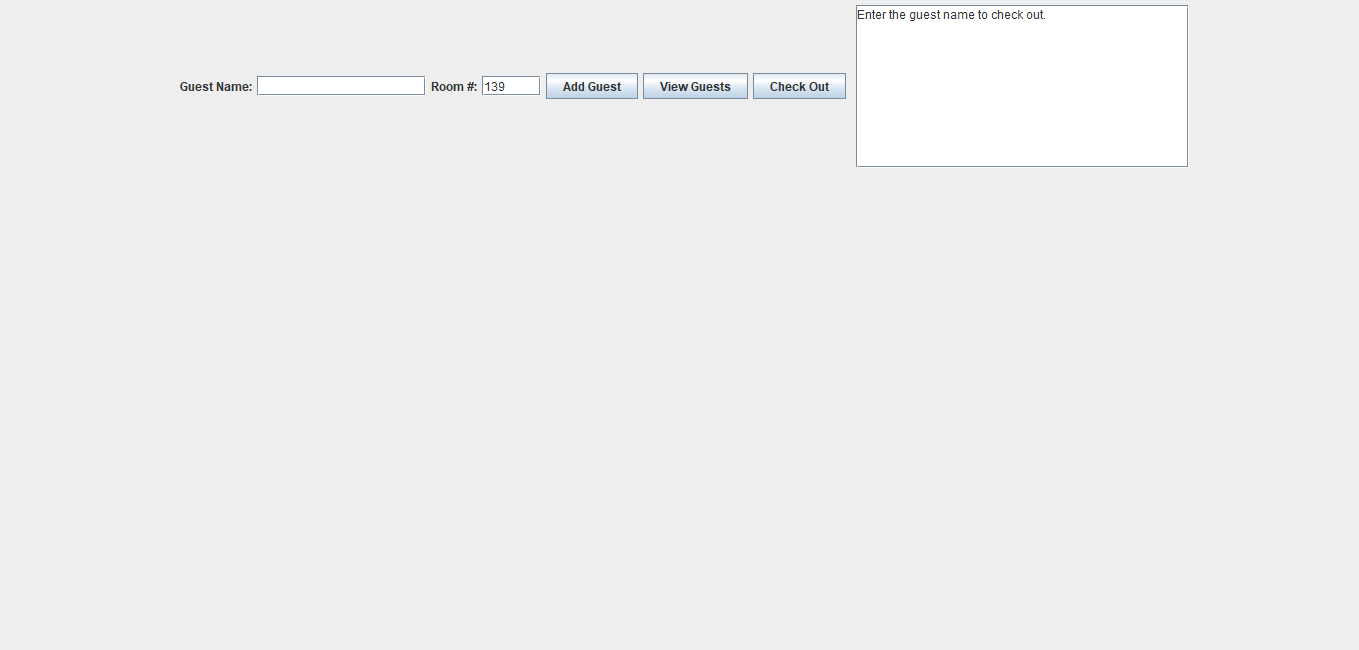
****

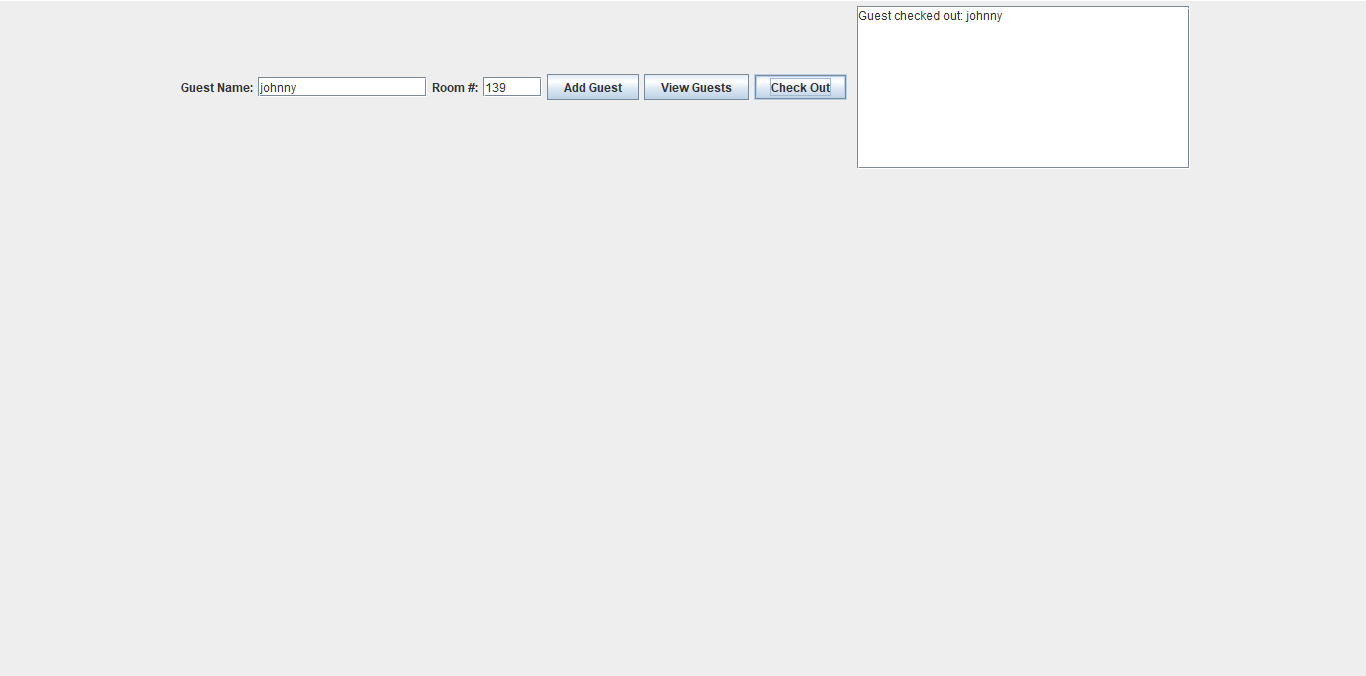
**Adding guest**

****

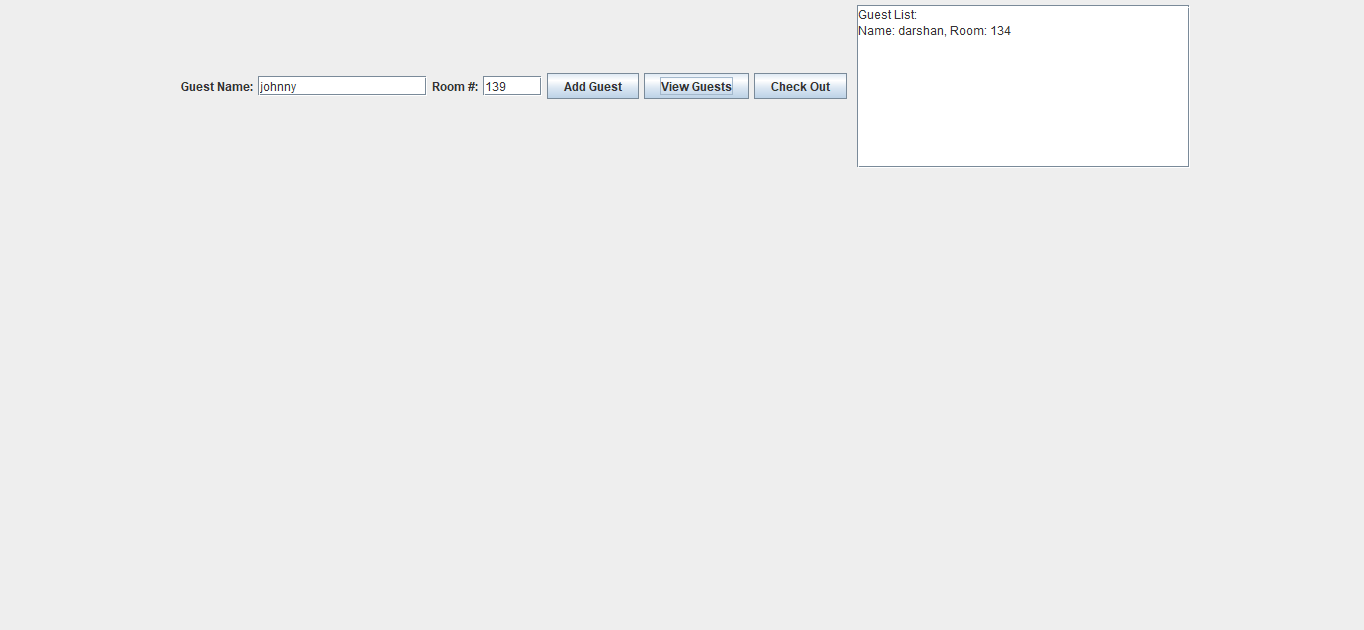
**Viewing Guests**

****

**Checkout  
  
**

****

**Viewing guest after checking out**

****

**Application of the project**

### ****1. Small Hotel or Guesthouse Management****

* Keep track of guest check-ins and check-outs.
* Assign room numbers to guests.
* View current guests with their room information.
* Useful for small-scale hotels, motels, or lodges with minimal staff.

### ****2. Educational Use (Learning Tool)****

* Demonstrates Java Swing for building GUIs.
* Implements event-driven programming using ActionListener.
* Introduces concepts like class objects (Guest), dynamic data structures (ArrayList), and form validation.
* Great as a **college mini-project** or Java GUI practice.

### ****3. Prototyping for Larger Systems****

* Acts as a **base model** for developing a full-featured hotel reservation system.
* Could be extended with:
  + Database connectivity (JDBC + MySQL/SQLite)
  + User authentication & role management
  + Calendar integration and booking features
  + Invoice and billing system

### ****4. Admin Panel for Hostel or Dormitories****

* Track student or resident check-ins/check-outs.
* Could be adapted for managing hostel rooms, especially for temporary visitors or events.

### ****5. Simulation Projects****

* Can simulate how a real hotel works for training hospitality staff.
* Used in business simulations or academic environments for testing management scenarios.

### ****6. Event Accommodation Management****

* Useful for **event organizers** who provide temporary accommodation for attendees (e.g., conferences, workshops, weddings).
* Can track guest allocation to different rooms or areas.

### ****7. Employee/Staff Quarters Management****

* For companies that provide **on-site housing** for employees (like in oil rigs, construction sites, or remote areas).
* This system could be used to manage who stays where and for how long.

### ****8. Shelter/Relief Camp Management****

* In emergency or disaster relief scenarios, NGOs or government agencies can use it to **track people staying in temporary shelters**.
* Helps with assigning beds, organizing logistics, and planning food distribution.

### ****9. Boarding School / College Dorm System****

* Tracks student dorm assignments, check-ins, and check-outs during semester changes or breaks.
* Can be used by hostel wardens or dorm managers.

### ****10. Military or Police Barracks Management****

* Tracks personnel assignment to bunks or rooms.
* Useful in military bases or large compounds with regular personnel rotation.

### ****11. Pet Hotel / Pet Boarding Management****

* Modify "Guest" to be pets (with owner info).
* Track which pet is in which room or kennel.

### ****12. Role-Playing Game (RPG) NPC Hotel System****

* If you're making a Java-based game, this system could simulate an inn or lodging system where players can “rest” or interact with characters.

### ****13. AirBnB Style Property Management****

* Used by landlords managing multiple rental units or vacation homes.
* Tracks current tenants and availability.

### ****14. Museum or Historical Site Overnight Programs****

* For museums or cultural centers that offer overnight stays or camp-ins, like "Night at the Museum" style events.

### ****15. Cruise Ship Cabin Management****

* Adapted to manage cabins and guests on cruise ships.

**Limitations of the project**

**1. No Data Persistence**

* **Issue**: All guest data is stored in memory (Araay List) and **lost when the app closes**.
* **Impact**: There's no record of guests once the program exits.
* **Fix**: Add file or database support (e.g., save to a .txt, .csv, or a database like MySQL/SQLite).

**2. No User Authentication or Account Management**

* **Issue**: The sign-up page collects info but doesn’t actually store or verify users.
* **Impact**: There's no **login system**, no differentiation between users, and no user database.
* **Fix**: Add login/sign-up logic with a user database.

**3. No Room Availability Tracking**

* **Issue**: The system doesn’t check if a room is already taken.
* **Impact**: Multiple guests can be assigned the same room.
* **Fix**: Add a check to prevent double-booking rooms.

**4. No Input Validation or Error Handling**

* **Issue**: Only basic checks exist (e.g., non-empty fields, integer check for room).
* **Impact**: Doesn’t prevent special characters, invalid emails, or overly long input.
* **Fix**: Use regex validation and stronger error handling for inputs.

**5. Poor Scalability**

* **Issue**: The GUI and data handling are not designed for **large data** sets.
* **Impact**: Performance might degrade with many guests, and the UI becomes cluttered.
* **Fix**: Implement pagination or switch to a more robust UI framework (JavaFX or web-based).

**6. No Role-Based Access (Admin/User)**

* **Issue**: Everyone using the app has the same permissions.
* **Impact**: No separation between front desk staff, managers, or customers.
* **Fix**: Add roles and permission levels to restrict/allow actions.

**7. No Date Management (Check-In/Out Times)**

* **Issue**: There's no handling of check-in or check-out **dates/times**.
* **Impact**: You can't track duration of stay, calculate billing, or plan room availability.
* **Fix**: Add LocalDate fields and date pickers for check-in/out.

**8. GUI is Functional but Basic**

* **Issue**: Very simple Swing UI; no styling, animations, or responsiveness.
* **Impact**: May not be user-friendly or professional-looking.
* **Fix**: Improve layout, switch to JavaFX, or move to web/mobile interface.

**9. No Search or Filter Functionality**

* **Issue**: You can’t search for a specific guest by name or room.
* **Impact**: Hard to manage when guest list grows.
* **Fix**: Add search bar or filters in the UI.

**10. Not Networked (Single-User Only)**

* **Issue**: Only one user can use it at a time, on one machine.
* **Impact**: Not usable by teams or across multiple reception desks.
* **Fix**: Add networking capabilities or convert it into a web app.

**Bibliography**

[**https://www.geeksforgeeks.org/java-swing/**](https://www.geeksforgeeks.org/java-swing/)

[**https://plantuml.com/**](https://plantuml.com/)

[**https://argouml.tigris.org/**](https://argouml.tigris.org/)

[**https://docs.oracle.com/javase/tutorial/java/concepts/**](https://docs.oracle.com/javase/tutorial/java/concepts/)

[**https://docs.oracle.com/javase/tutorial/**](https://docs.oracle.com/javase/tutorial/)

**Github link of the project**

<https://github.com/roxygrid77/HotelManagementSystem.git>