* INTRODUCTION OF THE PROJECT
* ***Overview***

*The Hotel Management System is a desktop application aimed at digitalizing hotel operations. It allows for efficient booking of rooms, staff allocation, and tracking of check-ins and check-outs. The system supports both admin and receptionist roles, offering them separate access and features. The application replaces manual paper-based workflows, improving speed and accuracy. It is built using Java Swing for GUI and MySQL for data storage. Design patterns are incorporated to make the system modular and extensible.*

* + - * *Objectives of the Project*
* ***Objective***
* *To provide a user-friendly interface for receptionists and admins that simplifies hotel operations.*
* *To manage room availability, customer details, and staff scheduling in an organized way.*
* *To implement well-known software design patterns like Builder, Adapter, and Factory for better code organization.*
* *To ensure data is stored securely and access is controlled based on user roles.*
* *To allow for future scalability, such as adding payment systems or online booking modules.*
* *To improve the efficiency, accuracy, and performance of hotel-related operations.*

# *Functional and Non-functional requirements*

### ***Functional Requirements***

1. ***User Authentication*** *The system must allow Admin and Receptionist users to log in using a valid username and password. Each user role should have specific access privileges.*
2. ***Room Management*** *Admins should be able to add, update, and remove rooms. Room status (available/booked) should be automatically updated based on bookings and checkouts.*
3. ***Employee Management*** *Admins should be able to add and manage employee details using a structured form. Employee records must be editable and deletable.*
4. ***Customer Booking*** *Receptionists should be able to book rooms for customers. The system should capture customer details, booking duration, and selected room type.*
5. ***Dashboard Navigation*** *Users should have access to an intuitive dashboard that allows quick navigation to major modules like Booking, Rooms, and Employees.*

### ***Non-Functional Requirements***

1. ***Usability*** *The system must provide a user-friendly and intuitive interface using Java Swing components. It should require minimal training for receptionists and admins.*
2. ***Performance*** *The system must handle operations like bookings and check-outs efficiently with quick screen transitions and database updates.*
3. ***Reliability*** *The application should handle unexpected inputs or failures gracefully without crashing. Error messages and validations must guide the user appropriately.*
4. ***Maintainability*** *The system’s design should follow Object-Oriented Programming and design pattern principles so that future changes and extensions are easy to implement.*
5. ***Security*** *User credentials should be verified during login, and role-based access should ensure that each user can only access permitted features.*

# Project Features

#### 1. **Role-Based Login System**

*The system provides separate login panels for Admin and Receptionist roles. Each user has access to specific features according to their role, ensuring data security and proper access control.*

#### 2. **Room Booking and Availability**

*Receptionists can view available rooms and book them for customers. The room status updates in real time, and the system prevents double-booking or invalid check-ins.*

#### 3. **Employee Management**

*Admins can add, edit, and delete employee records using a structured form. The Builder Pattern is applied here for organized and flexible object creation.*

#### 4. **Customer Check-Out Process**

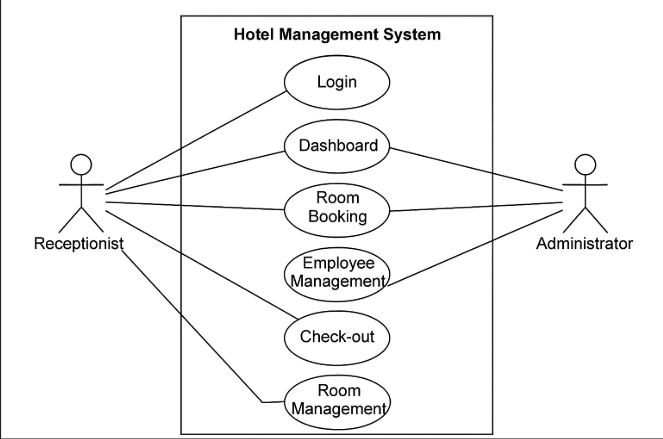
*The system supports customer check-out with proper status updates and receipt generation. The Adapter Pattern is used to connect the check-out process with room status and customer records seamlessly.*

#### 5. **Graphical User Interface (GUI)**

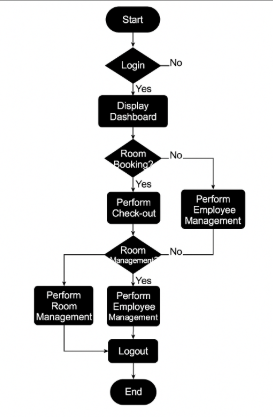
*The project uses Java Swing to build an intuitive, visually modern, and responsive interface. Clear navigation, icons, and user-friendly design make the system easy to use*.

# *BEHAVIORAL REPRESENTATION OF PROJECT*

# ***USES CASE***



* ***ACTIVITY DIAGRAM***



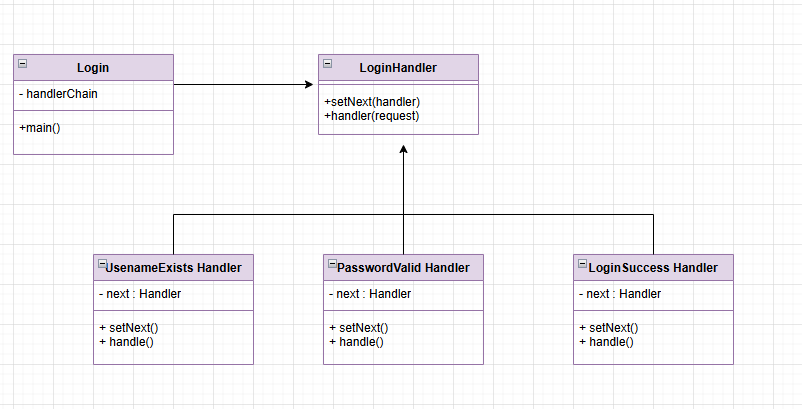
# ***Project Outcome:***

* ***Tools***

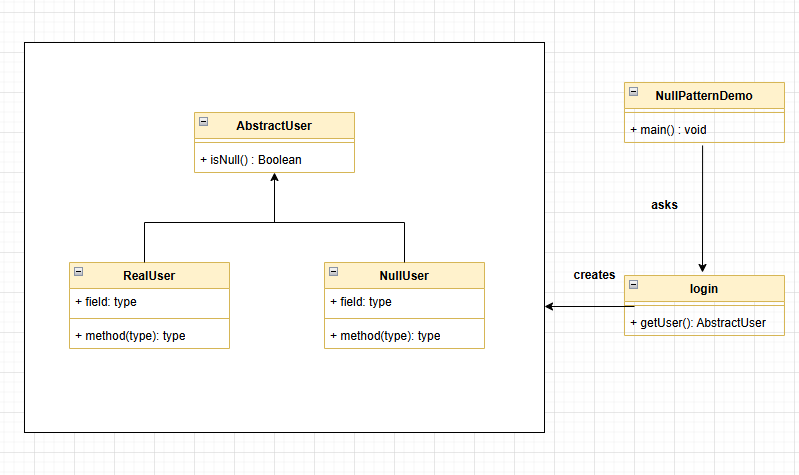
*The following tools and technologies were used throughout the development of the project to ensure functionality, performance, and ease of development:*

* ***Java****: The core programming language used to develop the application, following Object-Oriented Programming principles.*
* ***Java Swing****: A GUI toolkit used to design the visual components of the application such as forms, buttons, panels, and dialogs.*
* ***NetBeans IDE****: Used as the primary Integrated Development Environment (IDE) for writing, compiling, and debugging Java code. It provided visual GUI design support and integrated project management features.*
* ***MySQL****: A relational database used to store and manage hotel data including customers, rooms, employees, and bookings. It provided structured data handling and supported queries via JDBC.*
* ***JDBC (Java Database Connectivity)****: Enabled communication between the Java application and the MySQL database for executing SQL commands.*
* ***Design Pattern used (Name and Class Diagram of each pattern)***

#### **Chain of Responsibility Pattern**

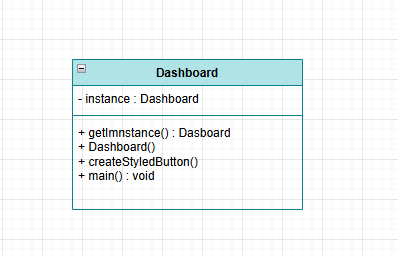
* ***Purpose****: Used in the login system to pass login validation through a chain of handlers (e.g., empty field check → format check → database check).*
* ***Used In****: Login.java*
* ***Benefit****: Increases flexibility and decouples the validation steps.*
* ***Class Diagram***

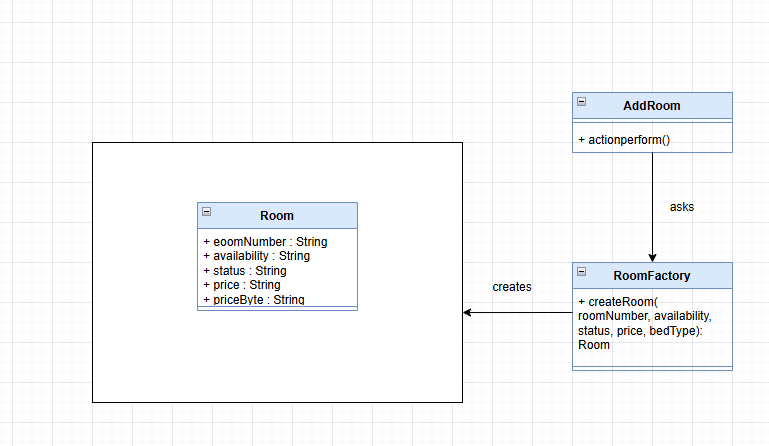
*2.* **Null Object Pattern**

* ***Purpose****: Avoids null checks by providing a default object (e.g., NullUser) when a user is not found.*
* ***Used In****: Login and validation logic.*
* ***Benefit****: Prevents NullPointerException and simplifies error handling.*
* ***Class Diagram****:* ****
  1. ***Singleton Pattern***
* ***Purpose****: Ensure only* ***one Dashboard window*** *is created and reused.*
* ***Used in****: LoginSuccessHandler (calls Dashboard.getInstance()).*
* ***Benefits****:*
  + *Avoids multiple Dashboard windows*
  + *Saves memory*
  + *Centralized control*
  + *Easy to access from anywhere in the app*

*Instead of new Dashboard(), use Dashboard.getInstance()*

* ***Class Diagram***

****

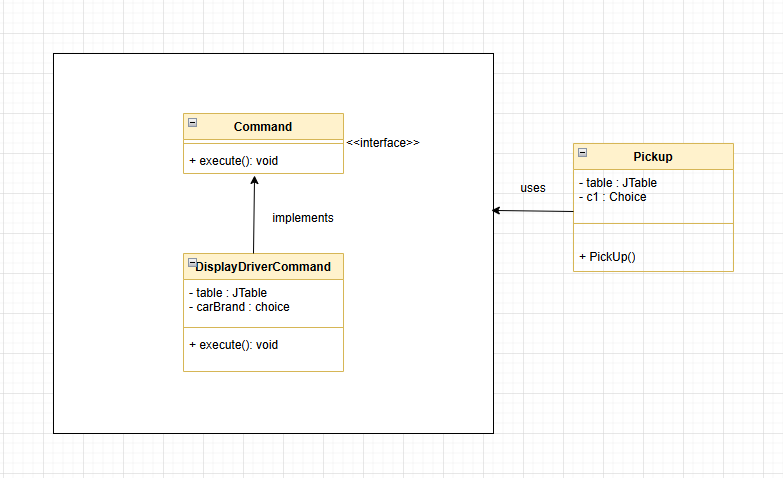
* 1. ***Builder Pattern***
* ***Purpose****: Used for constructing complex Employee objects step-by-step.*
* ***Used In****: EmployeeBuilder.java*
* ***Class Diagram****: (You can include a diagram showing Employee, EmployeeBuilder, and how the object is built.)*
  1. ***Factory Pattern***
* ***Purpose****: Used to create different types of Room objects based on conditions like room type.*
* ***Used In****: RoomFactory.java*
* ***Class Diagram****:* 
  1. ***Adapter Pattern***
* ***Purpose****: Applied in the checkout module to connect CheckOut logic with existing room and billing structures.*
* ***Used In****: CheckOut.java*
* ***Class Diagram****: (Include a diagram showing the Adapter, Target, and Adaptee classes.)*
  1. ***Command pattern***

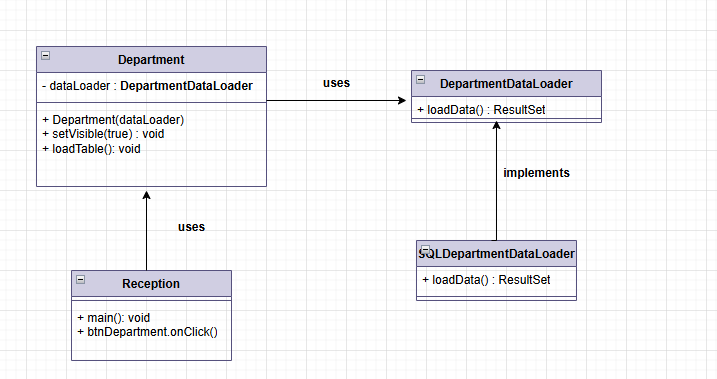
***Purpose:*** *To separate the action of displaying drivers from the GUI button logic.*

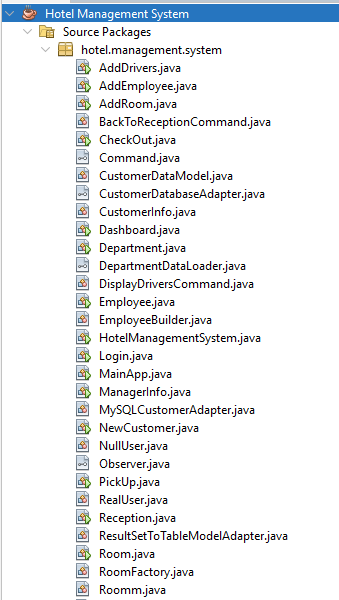
***• Used In:*** *PickUp (UI), DisplayDriversCommand (logic), Command (interface)*

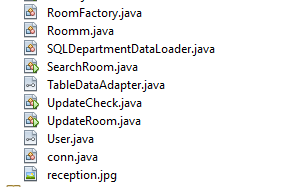
***• Benefit:***

* *Clean separation of GUI and logic*
* *Easier to maintain and test*
* *Can reuse or extend logic easily (e.g., add logging or undo)*

****

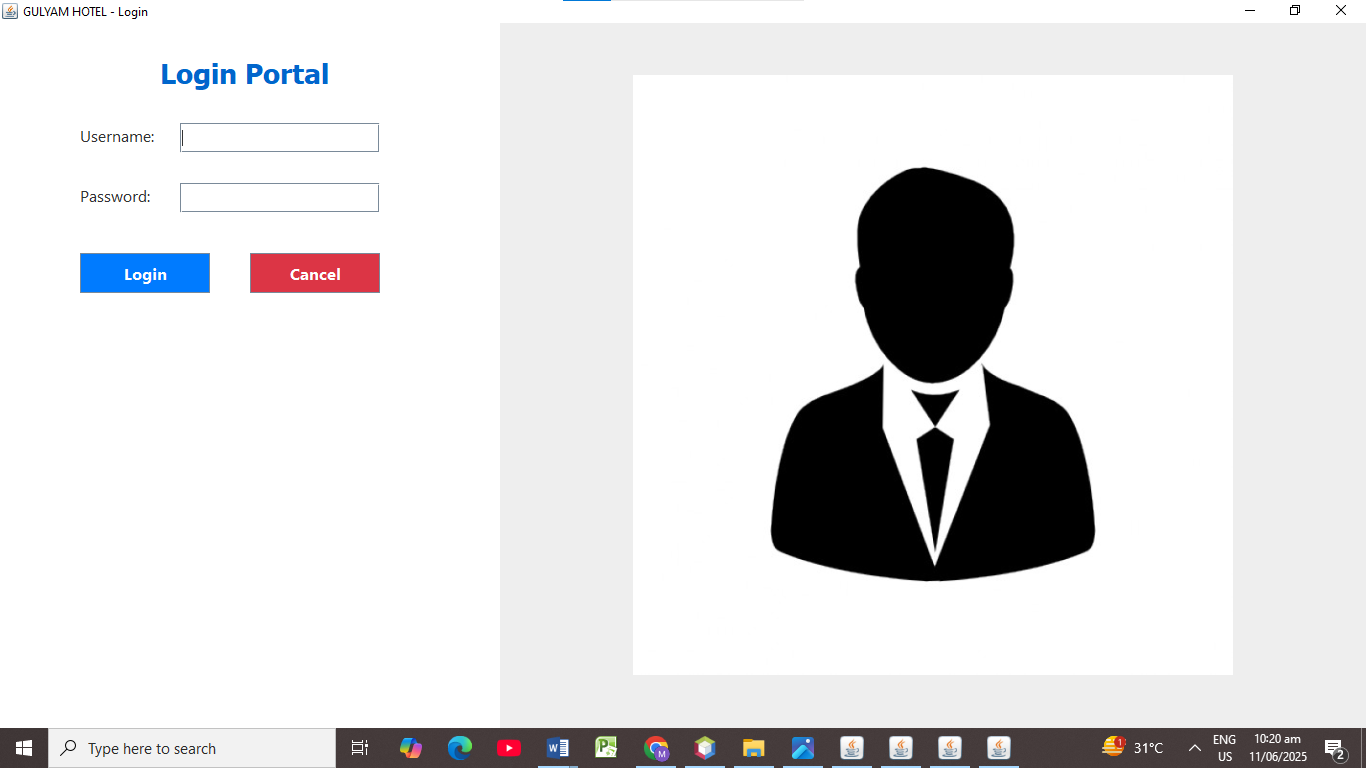
* 1. ***Bridge Pattern***
* **Purpose*:*** *It separates how data is loaded (e.g., from SQL, file, mock) from how data is displayed (in the Department JFrame).* This makes the code more flexible, testable, and easier to maintain.
* **Used in:** *Department class uses the Bridge Pattern. It depends on the DepartmentDataLoader interface, and uses the concrete class* ***SQLDepartmentDataLoader*** *to get data*.
* **Class diagram: **
* ***Development***

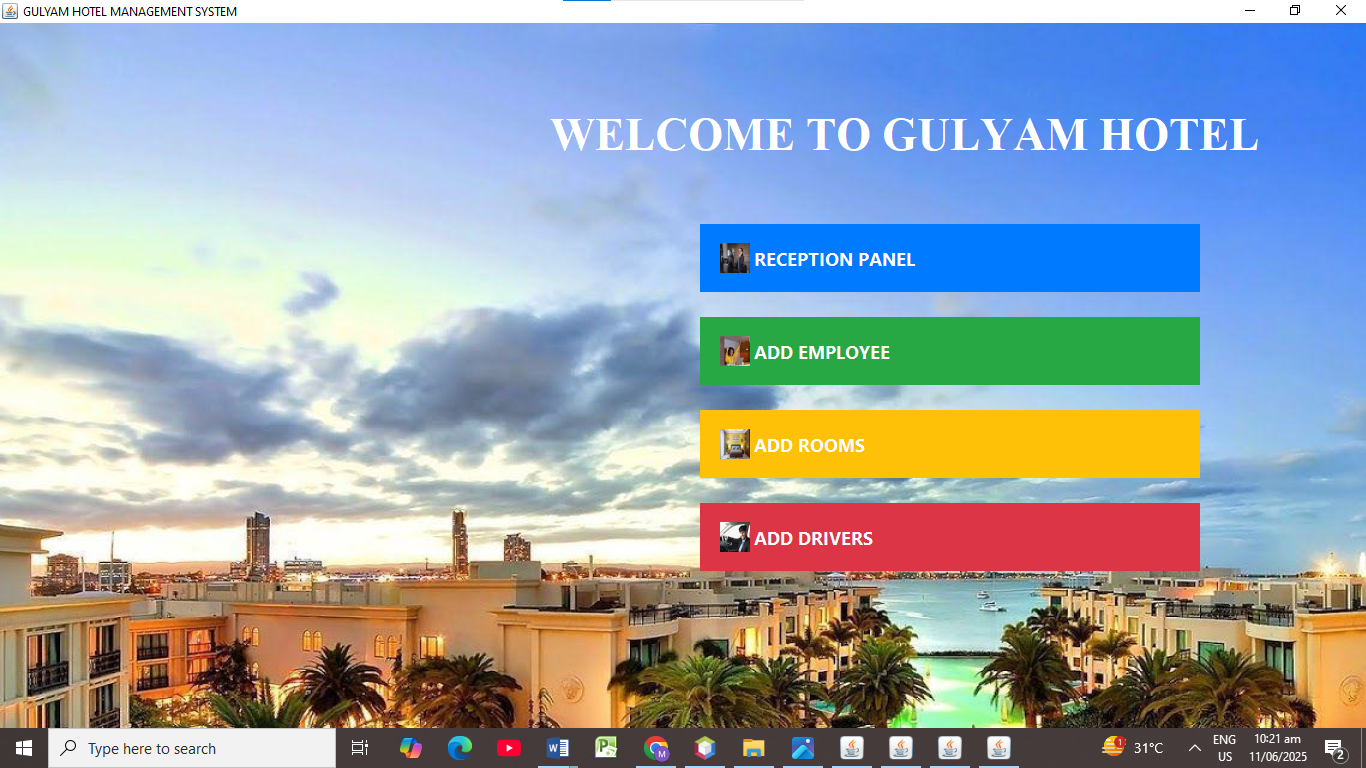
******

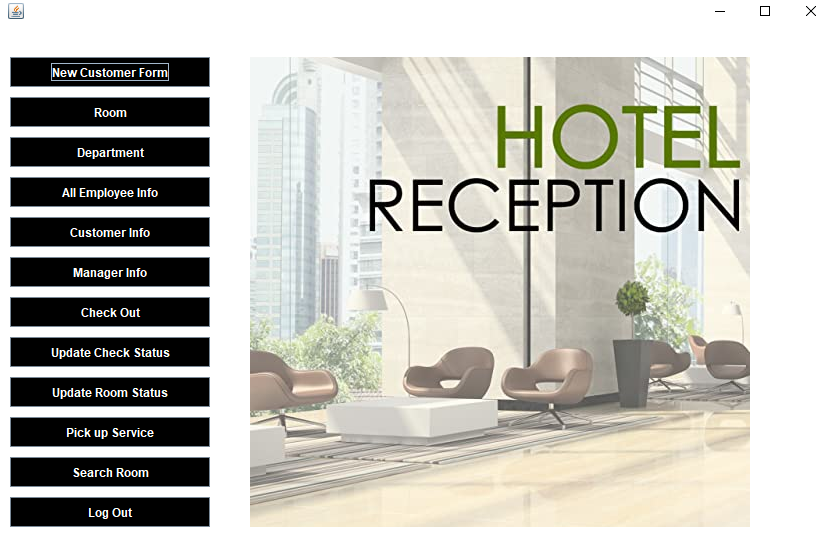
******

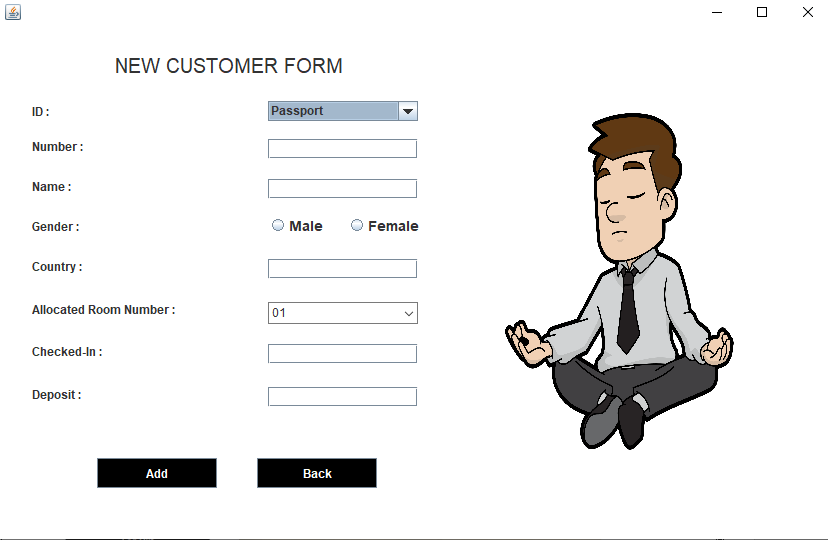
* ***Screen shots of your project***

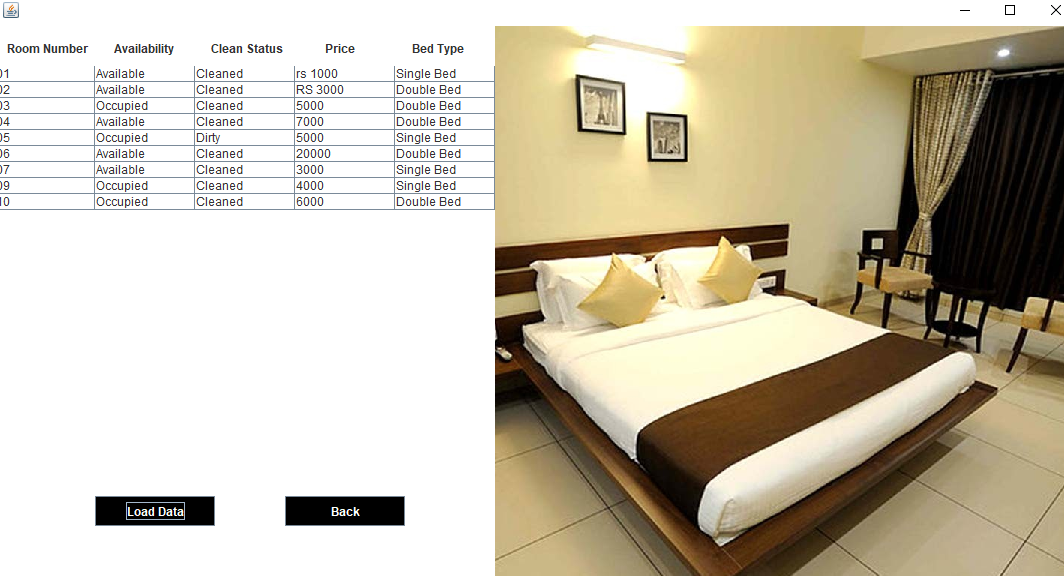


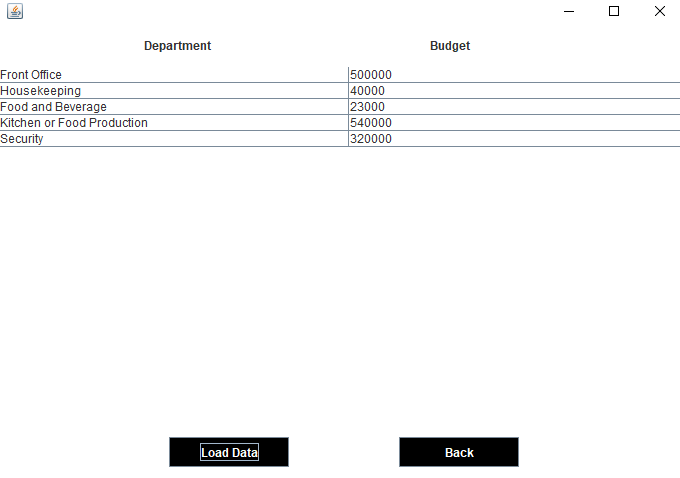


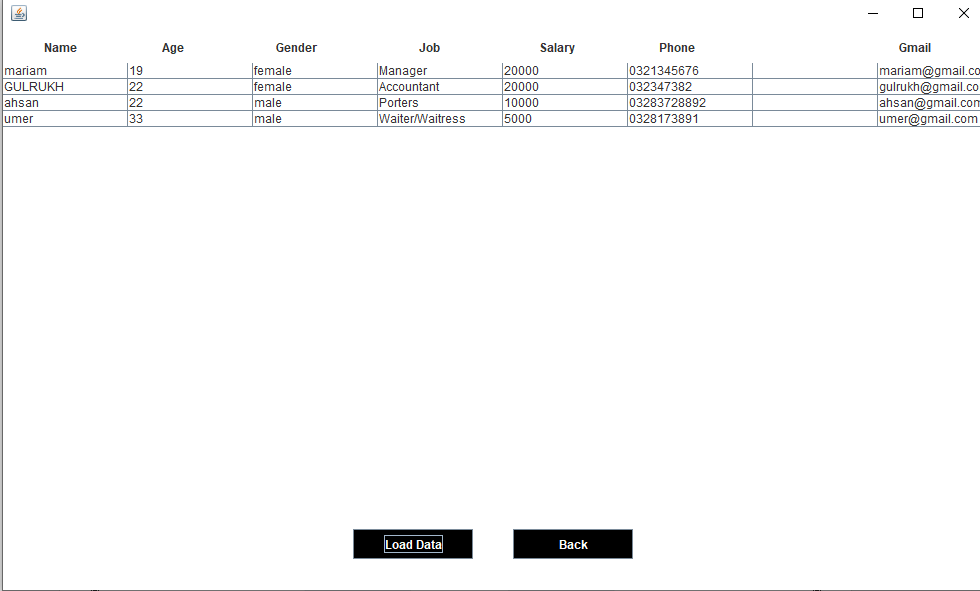


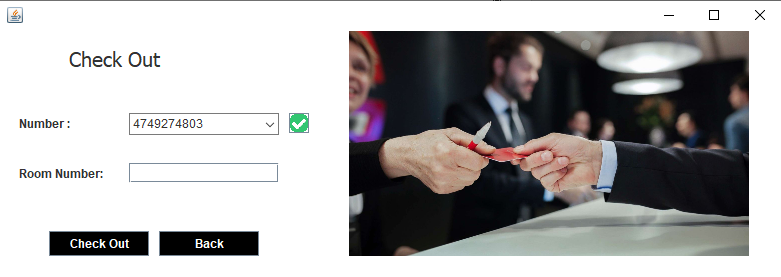


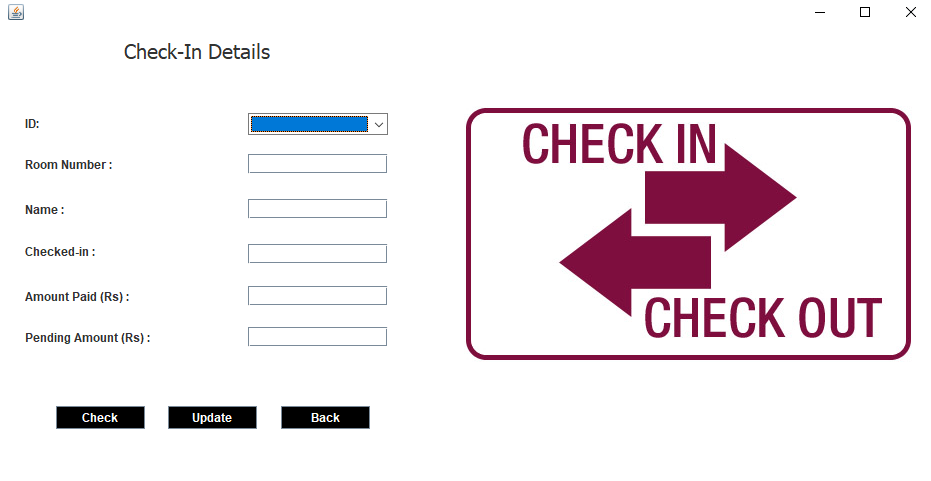


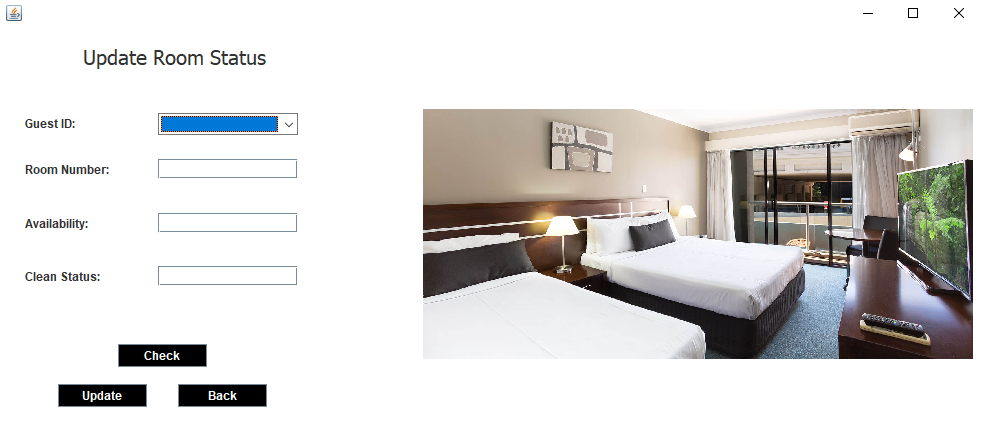


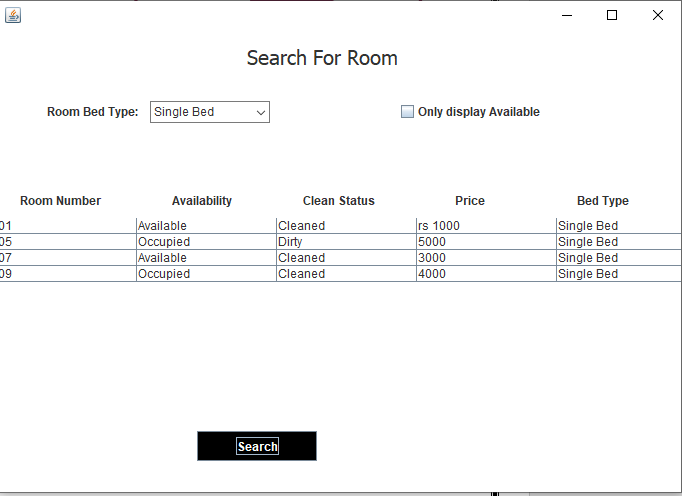


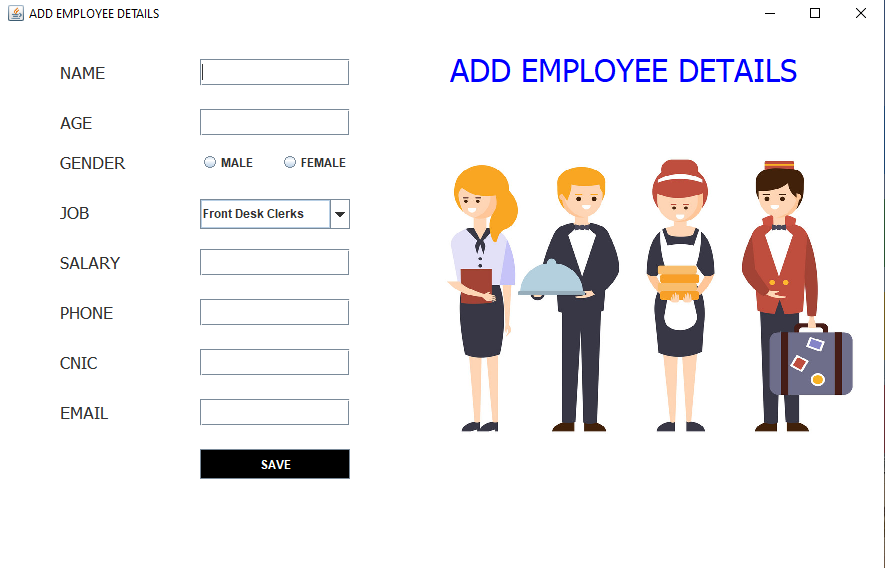


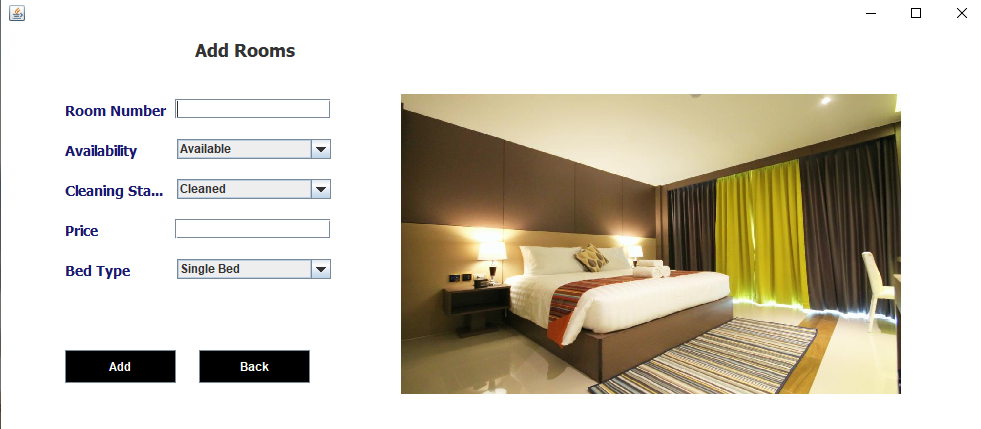
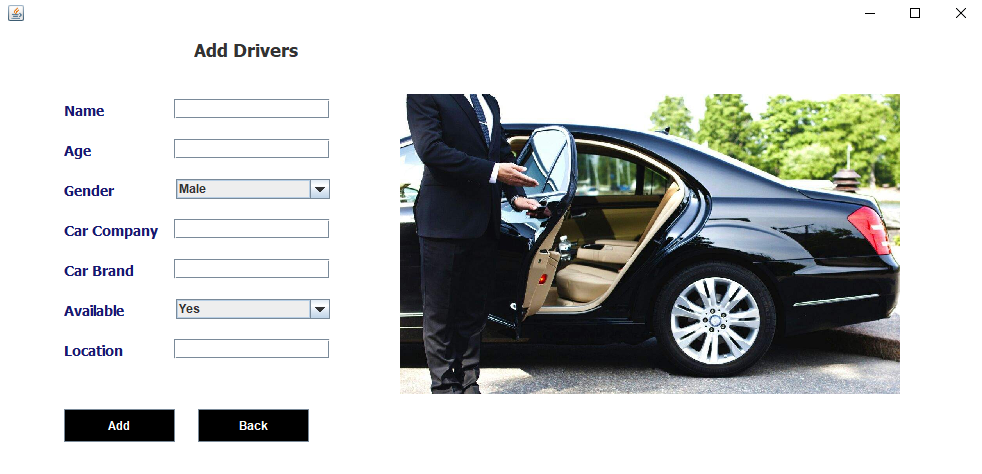










* ***Conclusion***

*The “Hotel Management System” effectively meets its objective of simplifying and automating basic hotel operations. From booking rooms to managing employees and customers, the system delivers a smooth and user-friendly experience. It replaces manual processes with automated workflows, saving time and reducing errors. By incorporating modern design patterns and GUI elements, the application maintains both functionality and flexibility. The project serves as a strong example of desktop-based hotel automation built with industry-relevant tools and practices*

* ***Future Expansion***
* *Move from a local MySQL setup to cloud-based storage for better accessibility and scalability.*
* *Include automated reporting features for bookings, revenue, and employee performance.*
* *Enhance the UI using more advanced layouts and responsive design principles.*
* *Add features such as SMS/email notifications for booking confirmations.*
* *Implement audit logging for actions performed by admins and receptionists.*