**Official Project Report: Room Booking System**

**Created By**

* Salsabila Herlin Syahputri Lubis 001202400209
* Diah Safitri Pane 001202400208
* Halima Abdirizak Mohamed 00120240024

**Project Title**

Room Booking System

**1. Introduction**

This report outlines the development and implementation details of the Room Booking System, a desktop application engineered to enhance hotel management operations.

**2. Project Scope and Objectives**

The Room Booking System is designed to serve as a centralized platform for hotel staff to accomplish the following objectives:

* Register and manage customer information.
* Add, update, and track room details and availability.
* Create, manage, and monitor bookings.

**3. System Architecture**

The system is structured using a layered architecture:

* **Presentation Layer**: Includes graphical user interface (GUI) components:
  + Home.java: The primary navigation dashboard.
  + GeneralScreen.java: A tabbed interface for performing CRUD operations on customers and rooms.
  + CustomerForm.java: A form for registering and editing customer details.
  + RoomForm.java: A form for managing room information.
  + BookingForm.java: A form for creating and managing bookings.
* **Business Logic Layer**: Comprises Data Access Objects (DAOs) for database interactions:
  + CustomerDAO.java: Manages customer-related database operations.
  + RoomDAO.java: Handles room-related database operations.
  + BookingDAO.java: Manages booking-related database operations.
* **Data Layer**: Utilizes a MySQL database (room.db), with its schema defined in room\_db.sql, including tables for customers, rooms, and bookings, and stored procedures for key functions.
* **Database Connectivity**: Facilitated by DatabaseConnection.java, which establishes a connection to the MySQL database using JDBC.
* **Model Classes**: Includes Room.java for representing room data, along with assumed model classes like Customer.java and Booking.java for data encapsulation.

**4. System Features**

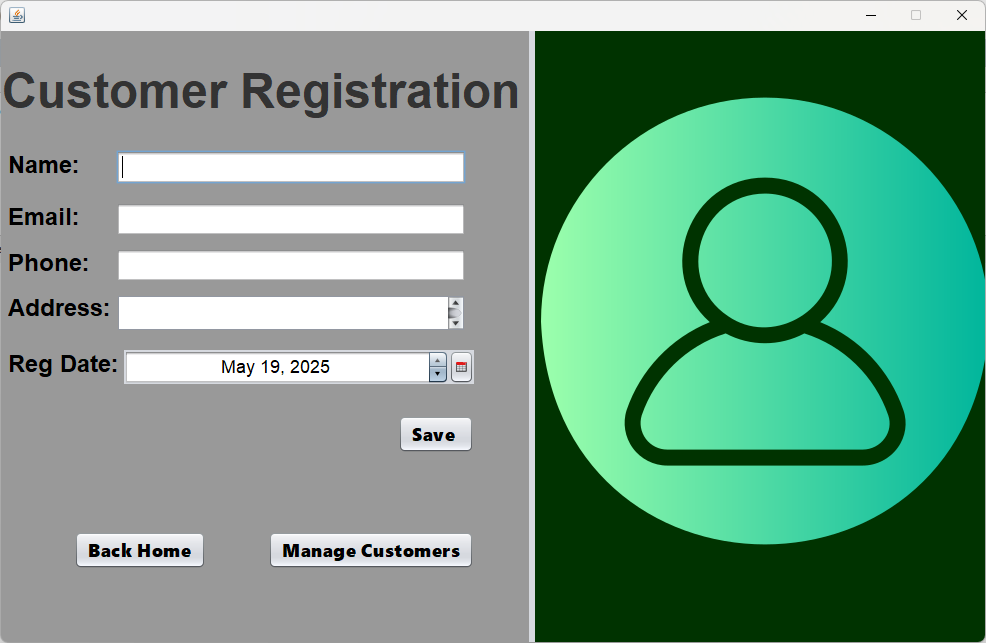
**4.1 Navigation and Dashboard**

* **File**: Home.java
* **Description**: Serves as the application’s entry point, featuring a dashboard with four clickable cards (Customers, Rooms, Bookings, General). Each card directs users to respective forms (CustomerForm.java, RoomForm.java, BookingForm.java, GeneralScreen.java), enhanced with icons for improved interactivity.



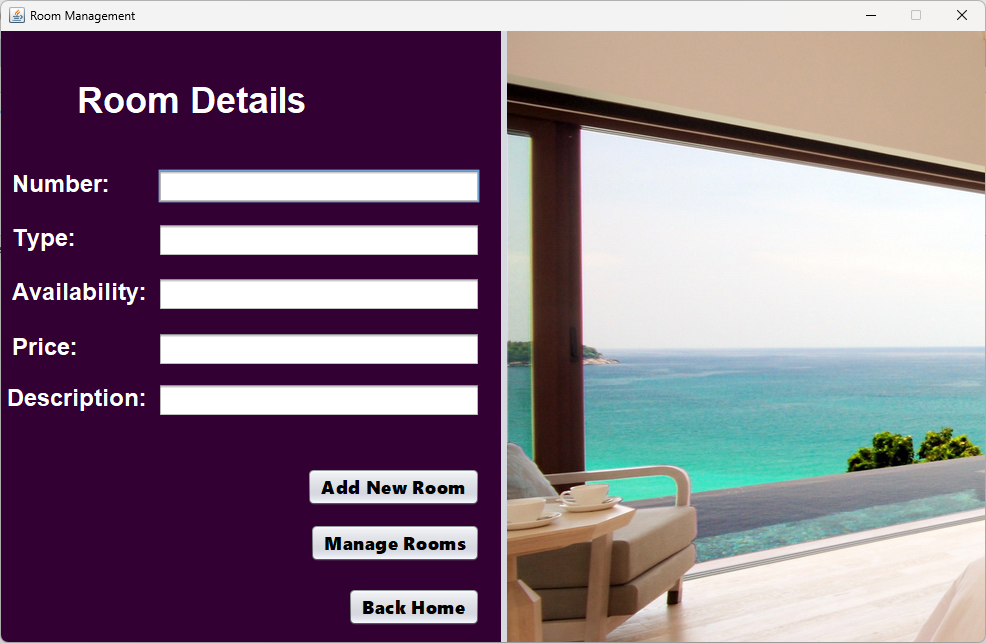
**4.2 Customer Management**

* **Files**: CustomerForm.java, CustomerDAO.java, GeneralScreen.java (Customer Tab)
* **Description**:
  + CustomerForm.java facilitates the registration of new customers, capturing details such as name, phone, email, address, and registration date.
  + GeneralScreen.java (Customer Tab) displays a table of all customers, enabling CRUD operations (edit and delete) via CustomerDAO.java.
  + Customer data is stored in the customers table in the database.



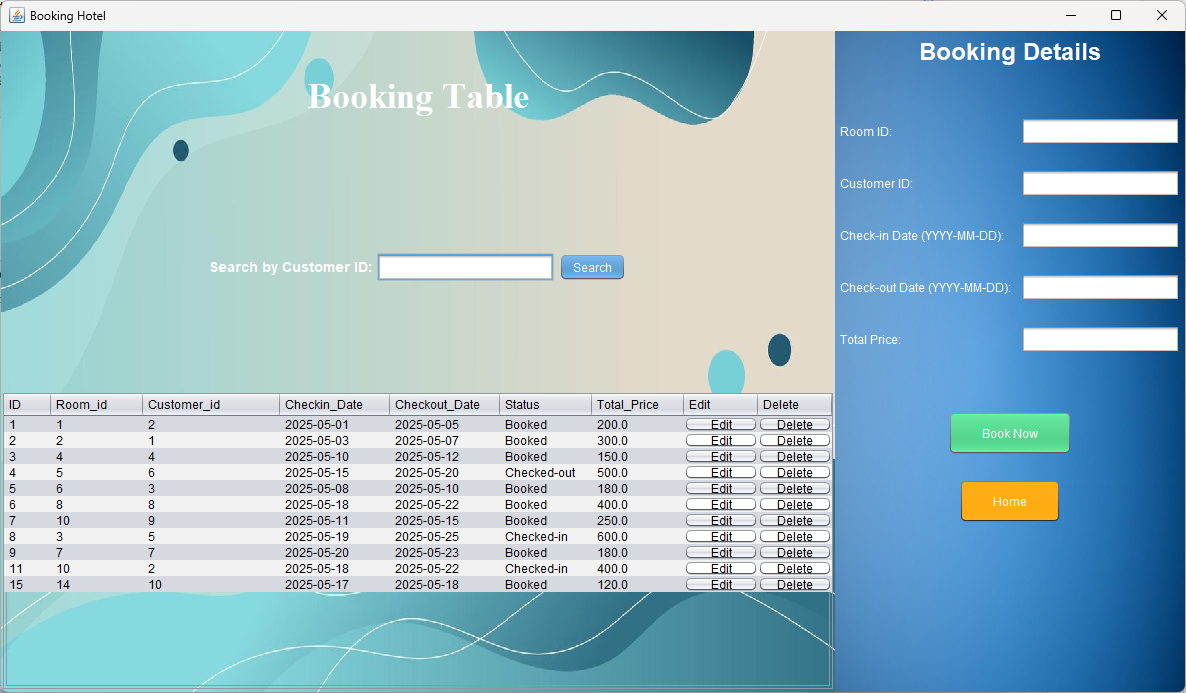
**4.3 Room Management**

* **Files**: RoomForm.java, RoomDAO.java, Room.java, GeneralScreen.java (Room Tab)
* **Description**:
  + RoomForm.java allows the addition and modification of room details, including room number, type, availability, price, and description, with data modeled in Room.java.
  + GeneralScreen.java (Room Tab) lists all rooms, supporting CRUD operations (edit and delete) through RoomDAO.java.
  + Room data is persisted in the rooms table.



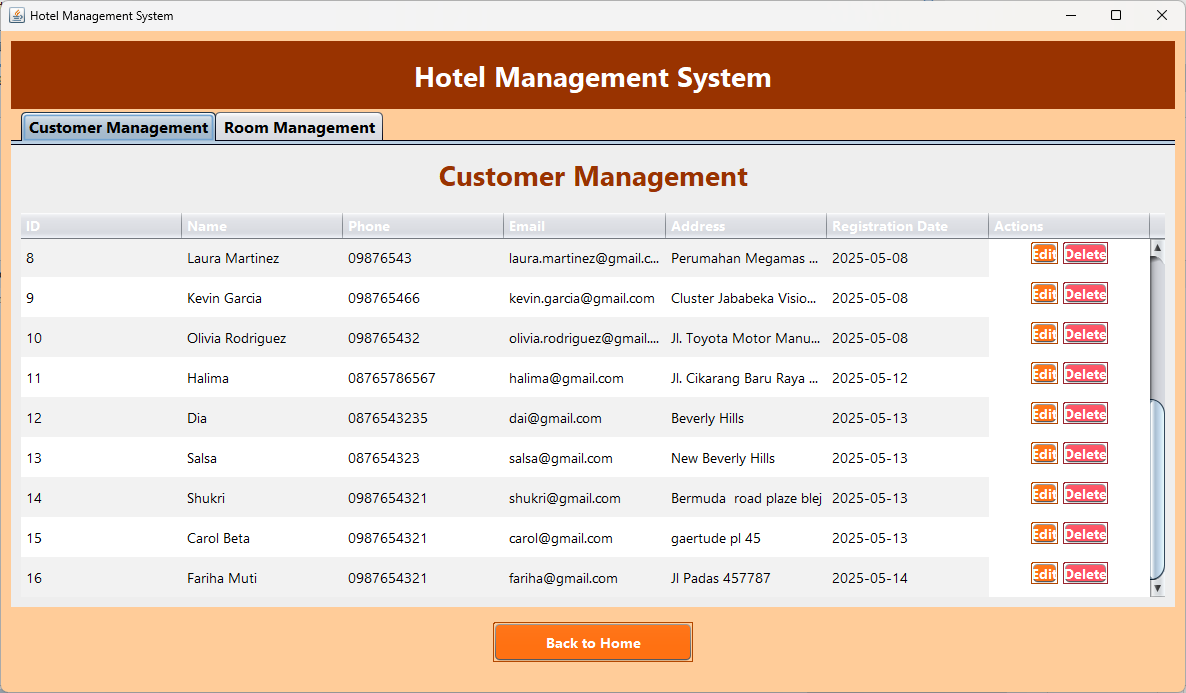
**4.4 Booking Management**

* **Files**: BookingForm.java, BookingDAO.java, Booking.java
* **Description**:
  + **Model (**Booking.java**)**: Represents a booking with attributes such as ID, Room\_ID, Customer\_ID, Checkin\_Date, Checkout\_Date, Status, and Total\_Price. It provides two constructors: one for new bookings (without ID) and another for existing bookings (with ID). Getters and setters ensure data encapsulation, and a toString method provides a string representation of the booking.
  + **DAO (**BookingDAO.java**)**: Handles database operations for bookings, including:
    - Creating new bookings via BookingsScreen, which inserts booking details into the bookings table.
    - Updating existing bookings with updateBooking, allowing modifications to all booking fields.
    - Deleting bookings using deleteBooking, based on the booking ID.
    - Retrieving bookings through methods like getAllBookings, getBookingById, getBookingsByCustomerId, getBookingsByRoomId, getBookingsByStatus, and getActiveBookings, enabling flexible data access.
    - Checking room availability with isRoomAvailable, which verifies if a room is free for a specified date range by querying overlapping bookings.
    - Updating booking status via updateBookingStatus, supporting status changes (e.g., "Booked", "Checked-in", "Completed", "Cancelled").
  + **UI (**BookingForm.java**)**: Provides a comprehensive interface for managing bookings:
    - Displays a table of all bookings with columns for ID, Room\_ID, Customer\_ID, Checkin\_Date, Checkout\_Date, Status, Total\_Price, and Edit/Delete buttons.
    - Supports CRUD operations: users can create new bookings, edit existing ones, and delete bookings directly from the table.
    - Includes a form for entering booking details (Room ID, Customer ID, Check-in Date, Check-out Date, Total Price, and Status during edits), with validation for date formats (YYYY-MM-DD) and numeric fields.
    - Features a search functionality by Customer ID, allowing users to filter bookings using getBookingsByCustomerId.
    - Validates room availability before creating a new booking using isRoomAvailable, ensuring no overlapping bookings.
    - Provides navigation back to the Home screen via a "Home" button.
    - Enhances user experience with a custom table renderer (WhiteTextRenderer) for alternating row colors and button renderers/editors (ButtonRenderer, ButtonEditor) for Edit/Delete actions.



**4.5 CRUD Operations on Customers and Rooms**

* **Files**: GeneralScreen.java, CustomerDAO.java, RoomDAO.java
* **Description**:
  + GeneralScreen.java provides a tabbed interface with tables for customers and rooms, enabling Create, Read, Update, and Delete (CRUD) operations.
  + The Customer Tab allows viewing, editing, and deleting customer records, managed by CustomerDAO.java.
  + The Room Tab supports viewing, editing, and deleting room records, handled by RoomDAO.java.
  + These operations ensure dynamic management of data stored in the customers and rooms tables.





**5. User Interface Design**

* **Home Screen (Home.java)**: Features a header with the system title and four interactive cards with icons, styled in a consistent color scheme (dark blue, magenta, orange tones).
* **General Screen (GeneralScreen.java)**: A tabbed interface with tables for customer and room management, including action buttons (e.g., Edit/Delete) with hover effects for CRUD operations.
* **Forms (CustomerForm.java, RoomForm.java, BookingForm.java)**: Offer input fields for data entry, with validation and a user-friendly layout.
* **Styling**: Employs a cohesive theme with colors (dark orange, light orange, white) and fonts (Segoe UI, Dialog) for a professional appearance.

**6. Database Design**

* **Tables (from room\_db.sql)**:
  + customers: Stores customer details (ID, Name, Phone, Email, Address, Registration\_Date).
  + rooms: Stores room details (ID, Number, Type, Availability, Price, Description).
  + bookings: Stores booking details (ID, Room\_ID, Customer\_ID, Checkin\_Date, Checkout\_Date, Status, Total\_Price).
* **Stored Procedures**:
  + BookRoom: Calculates total price and updates room availability during booking.
  + CheckIn and CheckOut: Manage booking status and room availability (available for potential future integration).
  + SearchAvailableRooms: Retrieves rooms available for specified dates and capacity.
* **Constraints**: Foreign keys ensure referential integrity between bookings, customers, and rooms.

**7. Conclusion**

The Room Booking System delivers a robust and efficient solution for hotel management, integrating a user-friendly Java Swing interface with a MySQL database. It effectively supports customer and room management through CRUD operations in GeneralScreen.java, alongside booking management features.