

# HOTEL RESERVATION SYSTEM

## 1. Introduction

The **Hotel Reservation System** is a menu-driven command line application developed using **Core Java, JDBC, and MySQL**.

It allows users to manage hotel room reservations such as booking rooms, viewing reservations, searching by reservation ID, and cancelling reservations.

The project follows a **layered architecture** using:

- DTO (Data Transfer Object)
- DAO (Data Access Object)
- Service Layer
- Main Class

This ensures **modularity, reusability, and maintainability**.

---

## 2. Objectives

- To understand **Core Java concepts**
  - To implement **JDBC connectivity**
  - To interact with **MySQL database**
  - To follow **DAO & DTO design pattern**
  - To create a **menu-driven console application**
- 

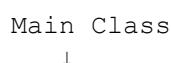
## 3. Technologies Used

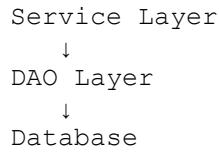
Technology	Description
Java	Core Java (JDK 8+)
JDBC	Database connectivity
MySQL	Relational Database
VS Code	IDE
MySQL Connector	JDBC Driver

---

## 4. Project Architecture

The project follows a **layered architecture**:





## 5. Project Structure

```
HotelReservationSystem
├── src
│   ├── dao
│   │   ├── ReservationDAO.java
│   │   └── ReservationDAOImpl.java
│   ├── dto
│   │   └── Reservation.java
│   ├── service
│   │   └── ReservationService.java
│   ├── util
│   │   └── DBConnection.java
│   └── main
        └── HotelReservationApp.java
```

---

## 6. Module Description

### 6.1 DTO Layer

#### Reservation.java

- Used to transfer data between layers
  - Contains fields like:
    - Reservation ID
    - Guest Name
    - Room Type
    - Check-in Date
    - Check-out Date
- 

### 6.2 DAO Layer

#### ReservationDAO.java

- Interface defining database operations:
  - Add reservation
  - View all reservations
  - Search reservation

- Cancel reservation

### **ReservationDAOImpl.java**

- Implements the DAO interface
  - Contains JDBC code for database interaction
- 

## **6.3 Service Layer**

### **ReservationService.java**

- Acts as a bridge between DAO and Main class
  - Contains business logic
  - Calls DAO methods and handles responses
- 

## **6.4 Utility Layer**

### **DBConnection.java**

- Manages database connection
  - Uses JDBC DriverManager
  - Ensures reusability of connection logic
- 

## **6.5 Main Class**

### **HotelReservationApp.java**

- Entry point of the application
  - Provides menu-driven interface
  - Takes user input using Scanner
  - Calls service layer methods
- 

## **7. Database Design**

### **Database Name**

hotel\_db

### **Table Structure**

```
CREATE TABLE reservation (
```

```
reservation_id INT PRIMARY KEY AUTO_INCREMENT,  
guest_name VARCHAR(50),  
room_type VARCHAR(30),  
check_in DATE,  
check_out DATE  
) ;
```

---

## 8. Functionalities

1. Add new reservation
  2. View all reservations
  3. Search reservation by ID
  4. Cancel reservation
  5. Exit application
- 

## 9. Sample Output

```
--- HOTEL RESERVATION SYSTEM ---  
1. Add Reservation  
2. View All Reservations  
3. Search Reservation  
4. Cancel Reservation  
5. Exit  
Enter choice:
```

---

## 10. Advantages

- Simple and user-friendly
  - Modular architecture
  - Easy to maintain and extend
  - Uses standard design patterns
  - Suitable for academic projects
- 

## 11. Limitations

- Console-based interface
  - No authentication system
  - No payment module
  - Single user at a time
-

## 12. Future Enhancements

- GUI using JavaFX or Swing
  - User login and authentication
  - Room availability checking
  - Payment and billing module
  - Admin and customer roles
- 

## 13. Conclusion

The **Hotel Reservation System** successfully demonstrates the use of **Core Java, JDBC, and MySQL** to build a real-world application.

It follows proper coding standards and design patterns, making it suitable for **college mini-projects and learning purposes**.

---