

HOTEL BOOKING MANAGEMENT SYSTEM

Introduction

- A Java-based console application for managing hotel bookings.
- Provides functionalities to add, view, update, delete rooms, and book rooms.
- Includes search functionality and generates reports for booked and available rooms.
- Data is persisted in JSON File.

Features

1. Room Management (Add, View, Update, Delete rooms)
2. Booking System (Book rooms for specific dates)
3. Search Functionality (Filter by room type, price, availability)
4. Reports (Booked and available rooms for a given date range)
5. Data Persistence (Save and load rooms from a file)

Create and Setup Project

- Open IntelliJ IDE
- Go to File -> New -> Project
- In Project, name the project, choose the Build System as Maven, Select JDK appropriate version click create
- After that add dependency gson in maven repository official website
- If you want to run the application go to Main.java file and we able to see the run icon

1.Add Room

- Allows users to add new rooms to the system.
- Room details include: room number, type (Single/Double/Suite), price, availability.

2. View & Update Rooms

- View all rooms (available and unavailable).
- Update room details (room number, type, price, availability).

3. Delete Room

- Delete rooms from the system by specifying room number.

4. Book Room

- Book rooms by specifying customer name, room number, start date, and end date.
- Prevents double booking by checking availability for the specified dates.

5. Search Rooms

- Search for rooms by type, price range, and availability.
- Filters rooms based on the given criteria.

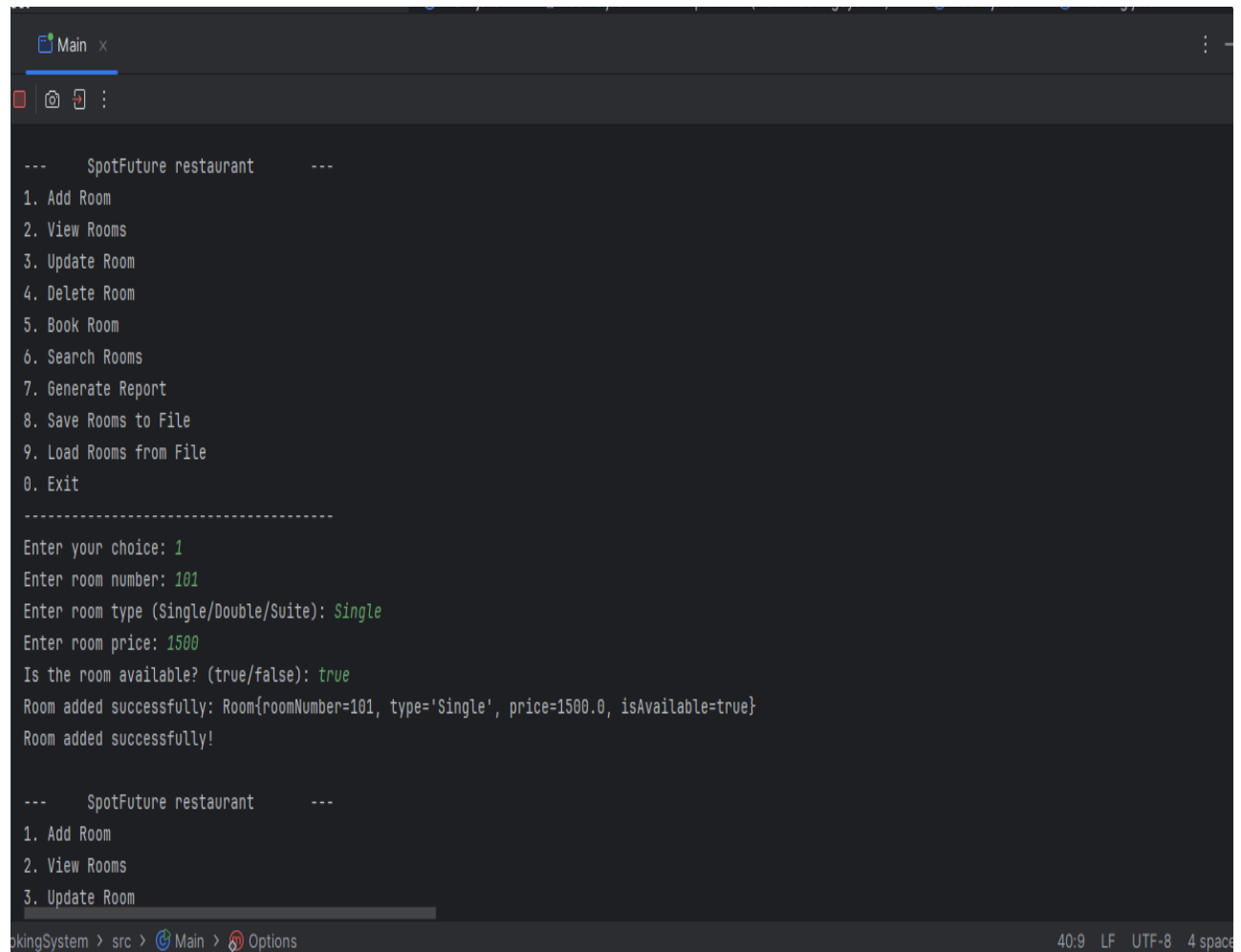
6. Generate Report

- Generate reports for booked and available rooms within a given date range.

7. Save and Load data

- Save rooms data to a file.
- Load rooms data from a file.
- Supports file formats like JSON File

Screenshots



```

Main x
[Icons]

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 1
Enter room number: 101
Enter room type (Single/Double/Suite): Single
Enter room price: 1500
Is the room available? (true/false): true
Room added successfully: Room{roomNumber=101, type='Single', price=1500.0, isAvailable=true}
Room added successfully!

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room

```

bookingSystem > src > Main > Options 40:9 LF UTF-8 4 space

Fig 1.1 Add Room

```
Run Main x
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 2
Room{roomNumber=101, type='Single', price=1500.0, isAvailable=true}
Room{roomNumber=102, type='Double', price=2000.0, isAvailable=true}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report

telBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces
```

Fig 1.2 View Room

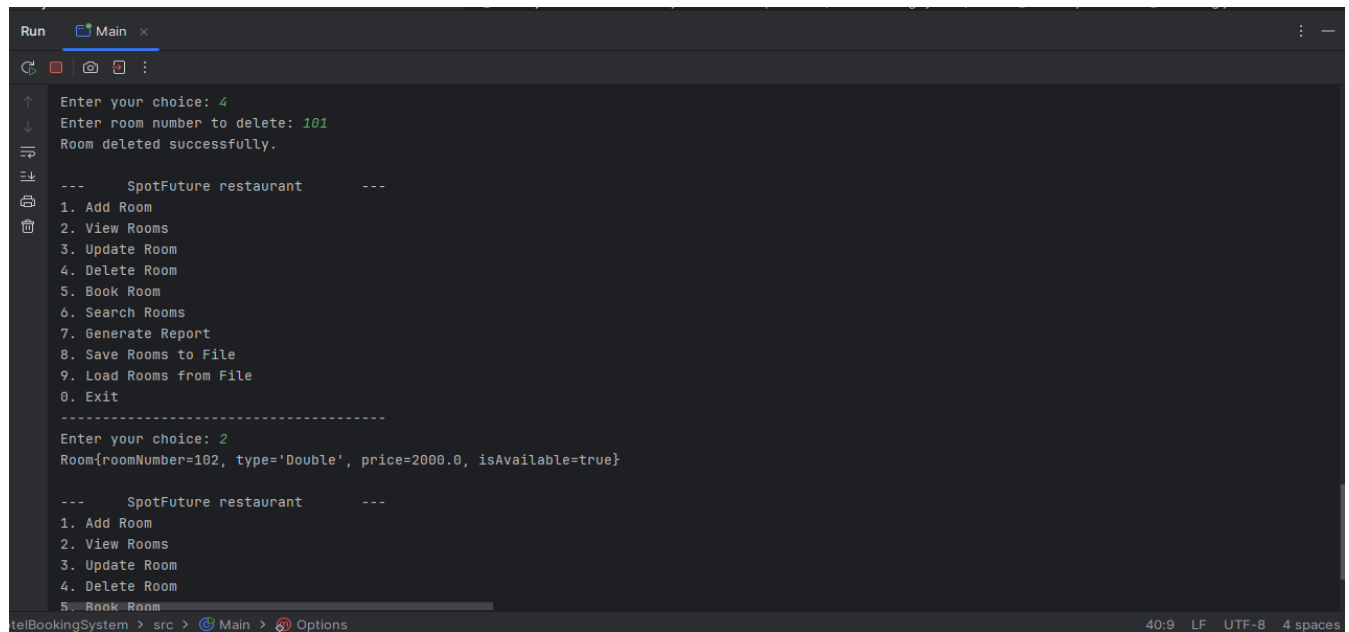
```
Room{roomNumber=102, type='Double', price=2000.0, isAvailable=true}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 3
Enter room number to update: 101
Enter new room type (Single/Double/Suite): Suite
Enter new room price: 1800
Is the room available? (true/false): false
Room updated successfully: Room{roomNumber=101, type='Suite', price=1800.0, isAvailable=false}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room

bookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces
```

Fig 1.3 Update Room



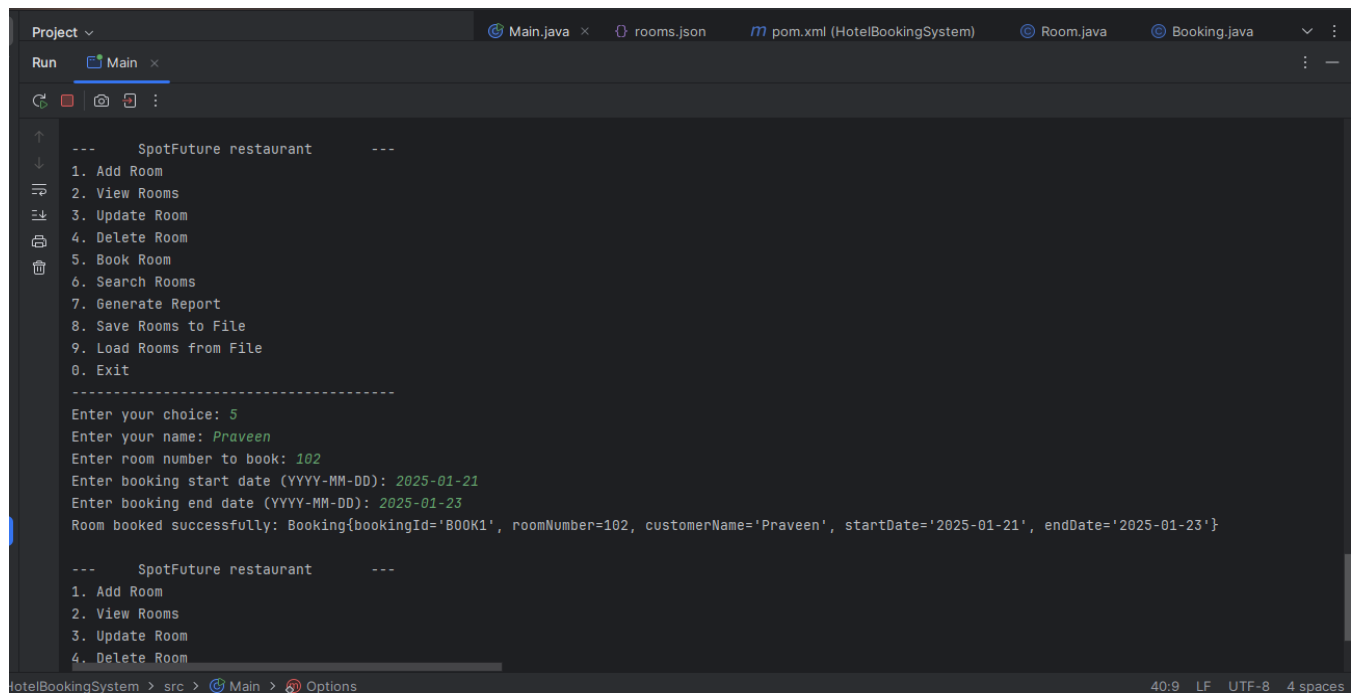
```
Run Main x
Enter your choice: 4
Enter room number to delete: 101
Room deleted successfully.

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 2
Room{roomNumber=102, type='Double', price=2000.0, isAvailable=true}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
```

HotelBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces

Fig 1.4 Delete Room

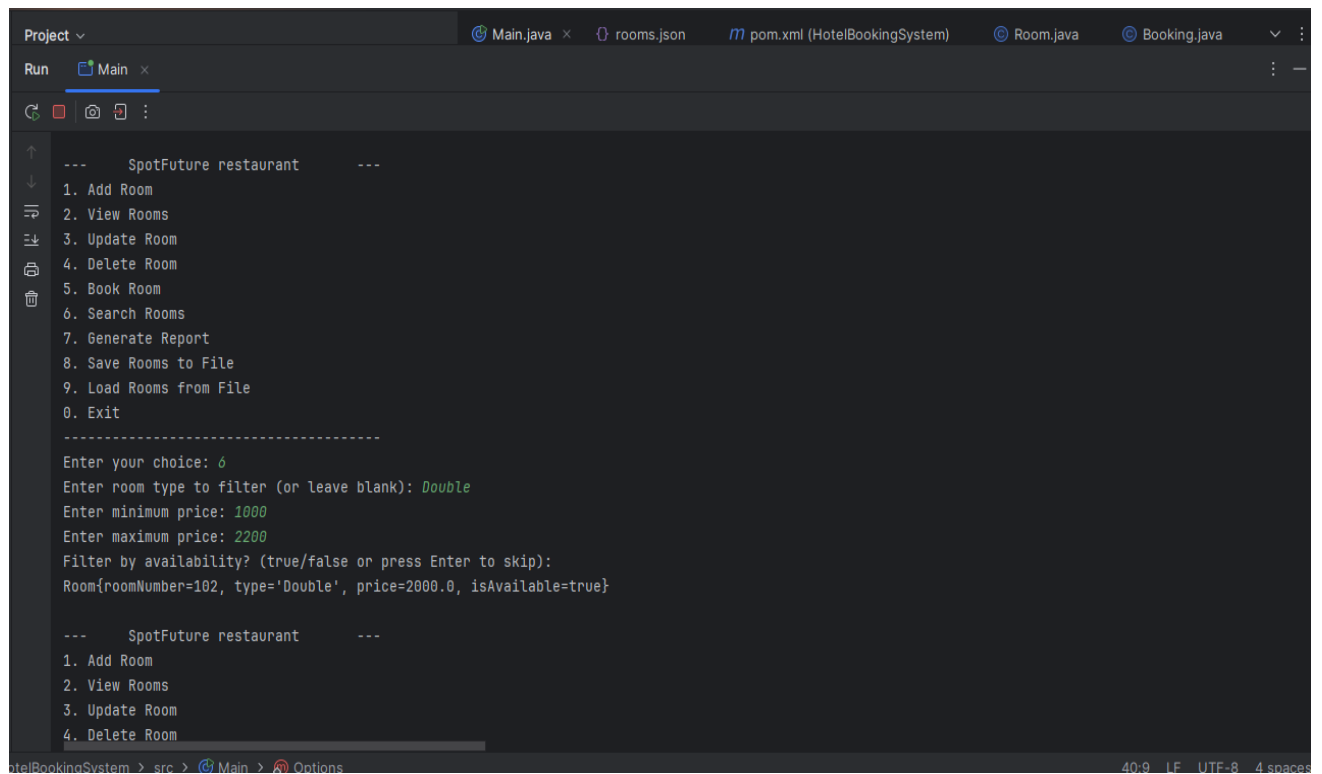


```
Project Main.java x rooms.json pom.xml (HotelBookingSystem) Room.java Booking.java
Run Main x
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 5
Enter your name: Praveen
Enter room number to book: 102
Enter booking start date (YYYY-MM-DD): 2025-01-21
Enter booking end date (YYYY-MM-DD): 2025-01-23
Room booked successfully: Booking{bookingId='B00K1', roomNumber=102, customerName='Praveen', startDate='2025-01-21', endDate='2025-01-23'}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
```

HotelBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces

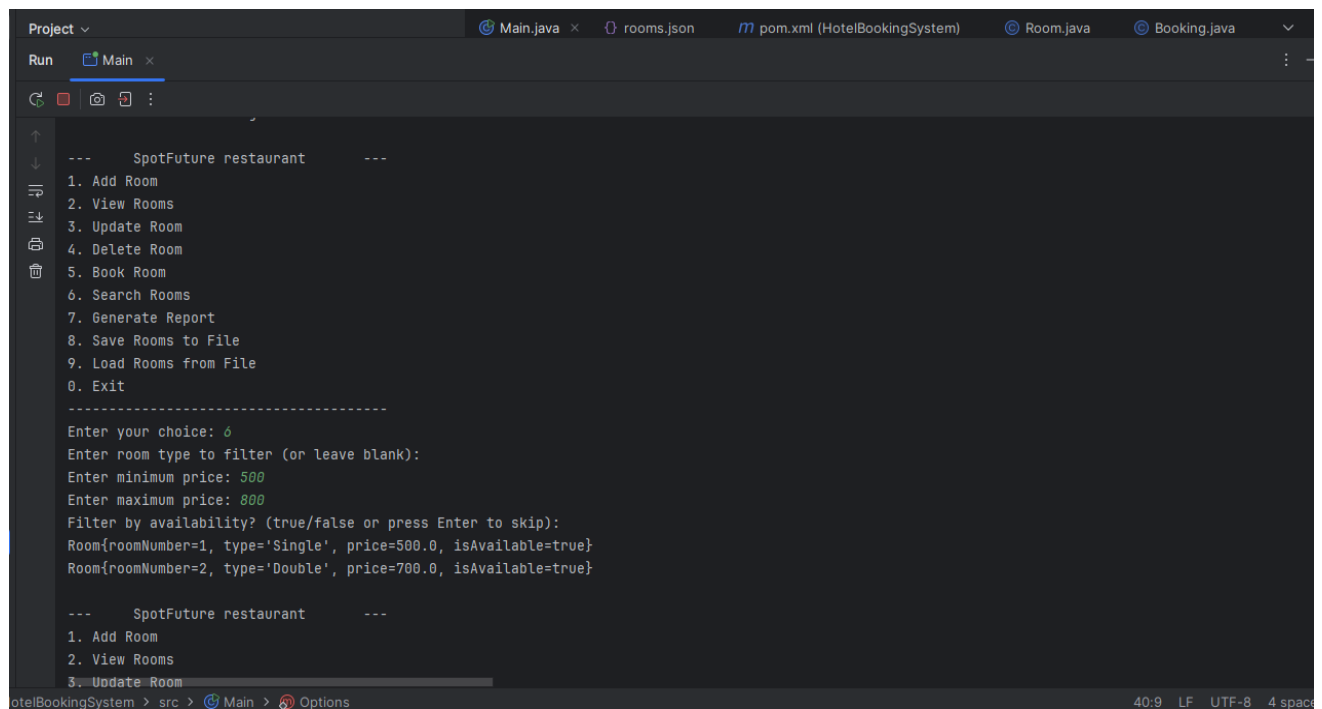
Fig 1.5 Book Room



```
Project ▾ Main.java × rooms.json pom.xml (HotelBookingSystem) Room.java Booking.java
Run Main ×
Run Main
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 6
Enter room type to filter (or leave blank): Double
Enter minimum price: 1000
Enter maximum price: 2200
Filter by availability? (true/false or press Enter to skip):
Room{roomNumber=102, type='Double', price=2000.0, isAvailable=true}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
otelBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces
```

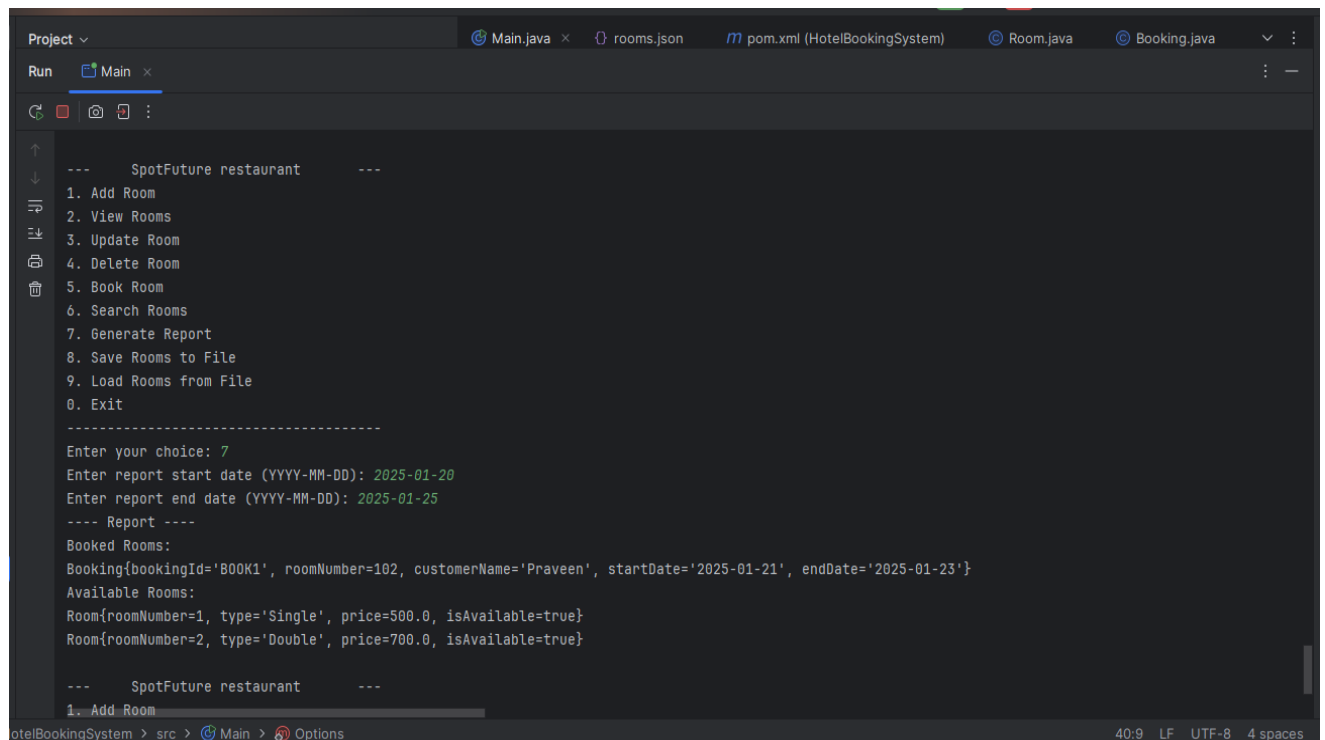
Fig 1.6.1 Search room based on type



```
Project ▾ Main.java × rooms.json pom.xml (HotelBookingSystem) Room.java Booking.java
Run Main ×
Run Main
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 6
Enter room type to filter (or leave blank):
Enter minimum price: 500
Enter maximum price: 800
Filter by availability? (true/false or press Enter to skip):
Room{roomNumber=1, type='Single', price=500.0, isAvailable=true}
Room{roomNumber=2, type='Double', price=700.0, isAvailable=true}

--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
otelBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces
```

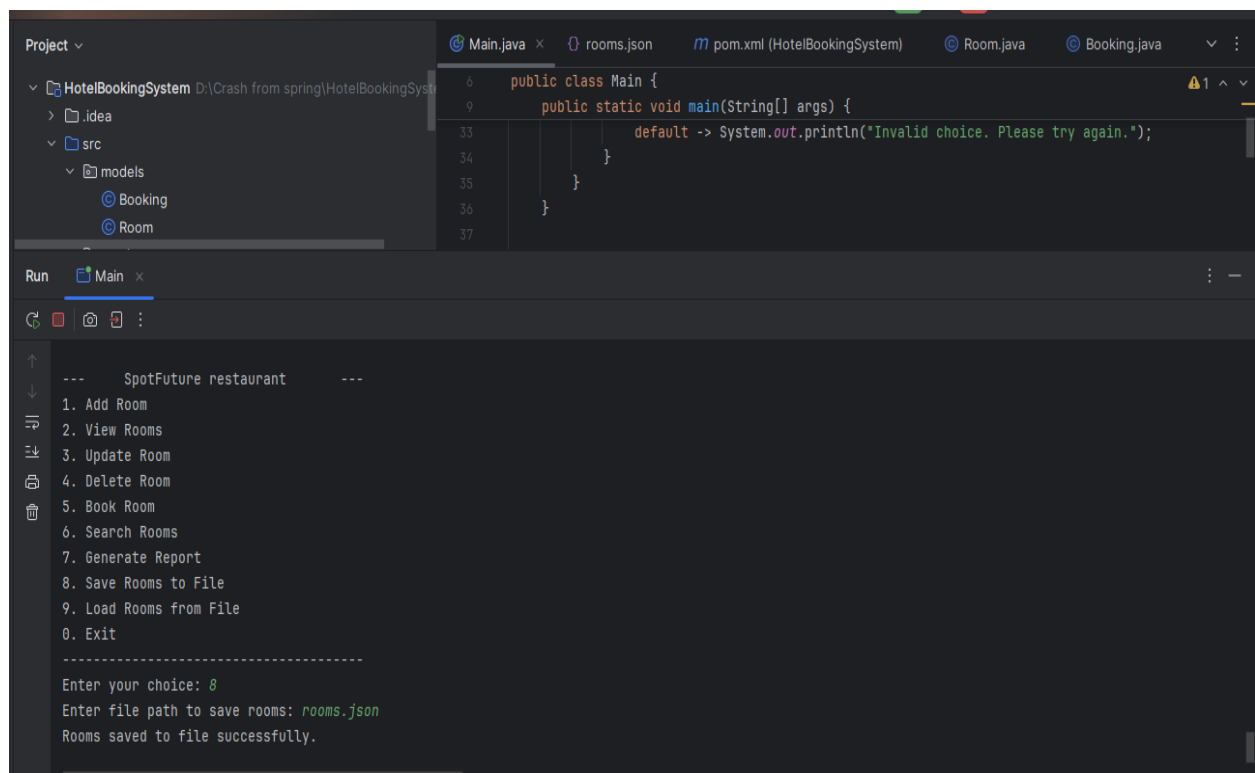
Fig 1.6.2 Search room based on price



```
Project ▾
Main.java × rooms.json pom.xml (HotelBookingSystem) Room.java Booking.java
Run Main ×
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 7
Enter report start date (YYYY-MM-DD): 2025-01-20
Enter report end date (YYYY-MM-DD): 2025-01-25
---- Report ----
Booked Rooms:
Booking{bookingId='B00K1', roomNumber=102, customerName='Praveen', startDate='2025-01-21', endDate='2025-01-23'}
Available Rooms:
Room{roomNumber=1, type='Single', price=500.0, isAvailable=true}
Room{roomNumber=2, type='Double', price=700.0, isAvailable=true}
--- SpotFuture restaurant ---
1. Add Room
```

HotelBookingSystem > src > Main > Options 40:9 LF UTF-8 4 spaces

Fig 1.7 Generate Report



```
Project ▾
HotelBookingSystem D:\Crash from spring\HotelBookingSystem
> .idea
> src
> models
> Booking
> Room
Main.java × rooms.json pom.xml (HotelBookingSystem) Room.java Booking.java
Run Main ×
--- SpotFuture restaurant ---
1. Add Room
2. View Rooms
3. Update Room
4. Delete Room
5. Book Room
6. Search Rooms
7. Generate Report
8. Save Rooms to File
9. Load Rooms from File
0. Exit
-----
Enter your choice: 8
Enter file path to save rooms: rooms.json
Rooms saved to file successfully.
```

```
6 public class Main {
9 public static void main(String[] args) {
33     default -> System.out.println("Invalid choice. Please try again.");
34 }
35 }
36 }
37 }
```

Fig 1.8 Save file

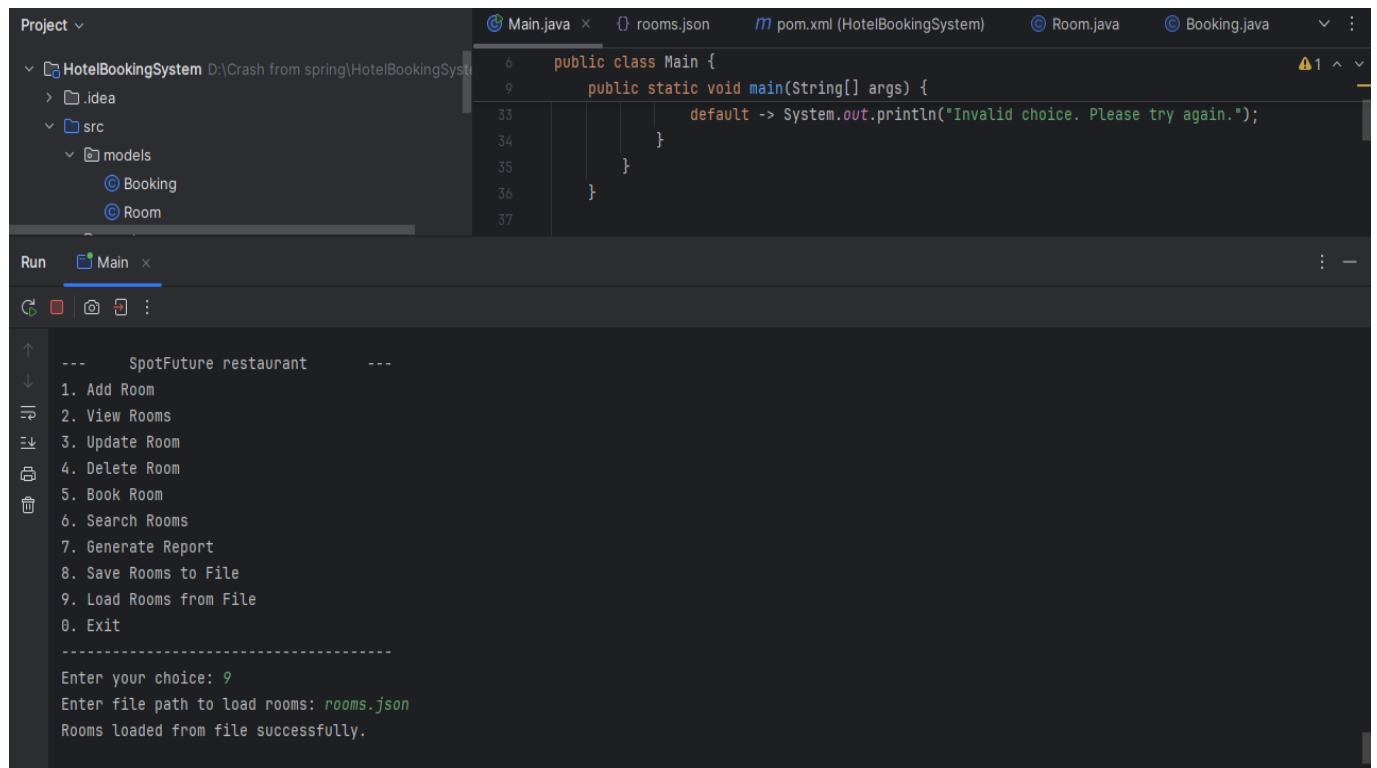


Fig 1.9 Load file

Coding

Room.java

```
public class Room {  
    private int roomNumber;  
    private String type;  
    private double price;  
    private boolean isAvailable;  
  
    //generate constructor  
    //generate getter and setter  
    //generate toString() method  
}
```

Booking.java

```
public class Booking {  
    private String bookingId;  
    private int roomNumber;  
    private String customerName;  
    private String startDate;  
    private String endDate;  
  
    //generate constructor  
    //generate getter and setter  
    //generate toString() method  
}
```

FileHandler.java

```
public class FileHandler {

    public static void saveRoomsToFile(String filePath, List<Room> rooms) throws
    IOException {

        Gson gson = new Gson();

        try (FileWriter writer = new FileWriter(filePath)) {

            writer.write(gson.toJson(rooms));

        }

    }

    public static List<Room> loadRoomsFromFile(String filePath) throws
    IOException {

        Gson gson = new Gson();

        try (FileReader reader = new FileReader(filePath)) {

            Room[] roomsArray = gson.fromJson(reader, Room[].class);

            return new ArrayList<>(Arrays.asList(roomsArray));

        }

    }

}
```

HotelManager.java

```
public class HotelManager {  
    private List<Room> rooms = new ArrayList<>();  
    private List<Booking> bookings = new ArrayList<>();  
  
    //add Room  
    public void addRoom(Room room) {  
        rooms.add(room);  
        System.out.println("Room added successfully: " + room);  
    }  
  
    //view Room  
    public void viewRooms() {  
        if (rooms.isEmpty()) {  
            System.out.println("No rooms available.");  
        } else {  
            rooms.forEach(System.out::println);  
        }  
    }  
  
    // Update room  
    public void updateRoom(int roomNumber, String type, double price, boolean  
isAvailable) {  
        for (Room room : rooms) {
```

```
        if (room.getRoomNumber() == roomNumber) {  
            room.setType(type);  
            room.setPrice(price);  
            room.setAvailable(isAvailable);  
            System.out.println("Room updated successfully: " + room);  
            return;  
        }  
    }  
    System.out.println("Room not found with number: " + roomNumber);  
}
```

// Delete a room

```
public void deleteRoom(int roomNumber) {  
    boolean removed = rooms.removeIf(room -> room.getRoomNumber() ==  
roomNumber);  
    if (removed) {  
        System.out.println("Room deleted successfully.");  
    } else {  
        System.out.println("Room not found with number: " + roomNumber);  
    }  
}
```

// Book a room

```
public void bookRoom(String customerName, int roomNumber, String  
startDate, String endDate) {
```

```
Room room = rooms.stream()
    .filter(r -> r.getRoomNumber() == roomNumber)
    .findFirst()
    .orElse(null);
```

```
if (room == null) {
    System.out.println("Room not found with number: " + roomNumber);
    return;
}
```

```
LocalDate start = LocalDate.parse(startDate,
DateTimeFormatter.ISO_DATE);
LocalDate end = LocalDate.parse(endDate, DateTimeFormatter.ISO_DATE);
```

```
boolean isBooked = bookings.stream()
    .anyMatch(b -> b.getRoomNumber() == roomNumber &&
        !(LocalDate.parse(b.getEndDate()).isBefore(start) ||
LocalDate.parse(b.getStartDate()).isAfter(end)));
```

```
if (isBooked) {
    System.out.println("Room is already booked for the given date range.");
} else {
```

```
    Booking booking = new Booking(generateBookingId(), roomNumber,
customerName, startDate, endDate);
```



```
        bookings.add(booking);
        System.out.println("Room booked successfully: " + booking);
    }
}
```

// Search a room

```
public void searchRooms(String type, double minPrice, double maxPrice,
Boolean isAvailable) {
```

```
    if(minPrice<0 && maxPrice<0){
        System.out.println("invalid price should be positive");
        return;
    }
```

```
    List<Room> filteredRooms = rooms.stream()
        .filter(room -> (type == null || type.isEmpty() ||
room.getType().equalsIgnoreCase(type)) && // Handle null or empty type
            room.getPrice() >= minPrice && // Price should be greater than or
equal to minPrice
            room.getPrice() <= maxPrice && // Price should be less than or
equal to maxPrice
            (isAvailable == null || room.isAvailable() == isAvailable)) // Check
availability if provided
        .collect(Collectors.toList());

    if (filteredRooms.isEmpty()) {
        System.out.println("No rooms found matching the criteria.");
    }
}
```

```
    } else {  
        filteredRooms.forEach(System.out::println);  
    }  
}
```

```
// Generate report
```

```
public void generateReport(String startDate, String endDate) {  
    LocalDate start = LocalDate.parse(startDate,  
DateFormatter.ISO_DATE);  
    LocalDate end = LocalDate.parse(endDate, DateFormatter.ISO_DATE);  
  
    System.out.println("---- Report ----");  
    System.out.println("Booked Rooms:");  
    bookings.stream()  
        .filter(b -> !(LocalDate.parse(b.getEndDate()).isBefore(start) ||  
LocalDate.parse(b.getStartDate()).isAfter(end)))  
        .forEach(System.out::println);  
  
    System.out.println("Available Rooms:");  
    List<Integer> bookedRoomNumbers = bookings.stream()  
        .filter(b -> !(LocalDate.parse(b.getEndDate()).isBefore(start) ||  
LocalDate.parse(b.getStartDate()).isAfter(end)))  
        .map(Booking::getRoomNumber)  
        .collect(Collectors.toList());
```

```
rooms.stream()
    .filter(r -> !bookedRoomNumbers.contains(r.getRoomNumber()))
    .forEach(System.out::println);
}
```

// Generate ID

```
private String generateBookingId() {
    return "BOOK" + (bookings.size() + 1);
}
```

// Save rooms

```
public void saveRooms(String filePath) {
    try {
        FileHandler.saveRoomsToFile(filePath, rooms);
        System.out.println("Rooms saved to file successfully.");
    } catch (Exception e) {
        System.out.println("Error saving rooms to file: " + e.getMessage());
    }
}
```

// Loads rooms

```
public void loadRooms(String filePath) {
    try {
        rooms = FileHandler.loadRoomsFromFile(filePath);
        System.out.println("Rooms loaded from file successfully.");
    }
}
```

```
    } catch (Exception e) {  
        System.out.println("Error loading rooms from file: " + e.getMessage());  
    }  
}  
}
```

Main.java

```
public class Main {  
    private static HotelManager hotelManager = new HotelManager();  
  
    public static void main(String[] args) {  
        Scanner s1 = new Scanner(System.in);  
        boolean value = true;  
  
        while(value) {  
            Options();  
            System.out.print("Enter your choice: ");  
            int choice = s1.nextInt();  
            s1.nextLine();  
  
            switch (choice) {  
                case 1 -> addRoom(s1);  
                case 2 -> viewRooms();  
                case 3 -> updateRoom(s1);  
                case 4 -> deleteRoom(s1);
```

```

        case 5 -> bookRoom(s1);
        case 6 -> searchRooms(s1);
        case 7 -> generateReport(s1);
        case 8 -> saveRooms(s1);
        case 9 -> loadRooms(s1);
        case 0 -> {
            System.out.println("Exiting the system. Goodbye!");
            value = false;
        }
        default -> System.out.println("Invalid choice. Please try again.");
    }
}
}

```

```

private static void Options() {
    System.out.println("\n--- SpotFuture restaurant ---");
    System.out.println("1. Add Room");
    System.out.println("2. View Rooms");
    System.out.println("3. Update Room");
    System.out.println("4. Delete Room");
    System.out.println("5. Book Room");
    System.out.println("6. Search Rooms");
    System.out.println("7. Generate Report");
    System.out.println("8. Save Rooms to File");
    System.out.println("9. Load Rooms from File");
}

```

```
System.out.println("0. Exit");  
System.out.println("-----");  
}
```

```
private static void viewRooms() {  
    hotelManager.viewRooms();  
}
```

```
private static void updateRoom(Scanner s1) {  
    System.out.print("Enter room number to update: ");  
    int roomNumber = s1.nextInt();  
    s1.nextLine();
```

```
    System.out.print("Enter new room type (Single/Double/Suite): ");  
    String type = s1.nextLine();
```

```
    System.out.print("Enter new room price: ");  
    double price = s1.nextDouble();
```

```
    if(price<0){  
        System.out.println("invalid price should be positive");  
        return;  
    }
```

```
System.out.print("Is the room available? (true/false): ");
boolean isAvailable = s1.nextBoolean();

hotelManager.updateRoom(roomNumber, type, price, isAvailable);
}

private static void deleteRoom(Scanner s1) {
    System.out.print("Enter room number to delete: ");
    int roomNumber = s1.nextInt();
    hotelManager.deleteRoom(roomNumber);
}

private static void bookRoom(Scanner s1) {
    System.out.print("Enter your name: ");
    String customerName = s1.nextLine();

    System.out.print("Enter room number to book: ");
    int roomNumber = s1.nextInt();
    s1.nextLine();

    System.out.print("Enter booking start date (YYYY-MM-DD): ");
    String startDate = s1.nextLine();

    System.out.print("Enter booking end date (YYYY-MM-DD): ");
```

```

String endDate = s1.nextLine();

hotelManager.bookRoom(customerName, roomNumber, startDate, endDate);
}

private static void searchRooms(Scanner s1) {
    System.out.print("Enter room type to filter (or leave blank): ");
    String type = s1.nextLine().trim();
    if (type.isEmpty()) type = null;

    System.out.print("Enter minimum price: ");
    double minPrice = s1.nextDouble();

    System.out.print("Enter maximum price: ");
    double maxPrice = s1.nextDouble();

    if(maxPrice<0 || minPrice<0){
        System.out.println("invalid minPrice or maxPrice should be positive");
        return;
    }

    System.out.print("Filter by availability? (true/false or press Enter to skip): ");
    s1.nextLine();
    String availabilityInput = s1.nextLine().trim();
    Boolean isAvailable = null;
    if (!availabilityInput.isEmpty()) {

```



```
        isAvailable = Boolean.parseBoolean(availabilityInput);
    }

    hotelManager.searchRooms(type, minPrice, maxPrice, isAvailable);
}
```

Future Enhancement

- Convert console Application into web Application
- Improve Calendar-based date picker usability
- Payment Integration
- Notification and Reminders
- Add room features(such as wifi, AC, ..etc)
- Mobile App Integration