Restaurant Management System

Name:Siddarth S

Objective

The objective of the Restaurant Management System (RMS) is to efficiently manage the operations of a restaurant, including tables, reservations, orders, and menu items. The system aims to streamline the management of restaurant resources, enhance customer service, and ensure smooth daily operations. This involves handling reservations, managing table assignments, processing orders, and maintaining an updated menu, ultimately supporting the restaurant's goal of providing a seamless dining experience for guests.

RestaurantTable

- **Represents:** A dining table within the restaurant.
- **Attributes:**
  - tableId: A unique identifier for the table.
  - tableNumber: A numeric identifier for the table used for reference.
  - capacity: The number of people the table can accommodate.
  - locationDescription: A description of the table's location within the restaurant (e.g., window seat, near the bar).
  - status: The current status of the table (e.g., available, reserved, occupied).
- **Role:** Tracks details about each table in the restaurant, including its location, capacity, and current status. This information is crucial for managing reservations and ensuring optimal table utilization.

Reservation

- **Represents:** A reservation made by a customer to secure a table for a specific time and date.
- **Attributes:**
  - reservationId: A unique identifier for the reservation.

    - userId: The identifier of the user who made the reservation.
    - reservationDate: The date and time when the reservation is scheduled.
    - numberOfPeople: The number of people for whom the reservation is made.
    - tableId: The identifier of the table reserved.
- **Role:** Manages customer reservations, including the date, time, number of people, and associated table. This ensures organized table assignments and efficient management of dining schedules.

Order

- **Represents:** A customer's order placed for food and/or drinks.
- **Attributes:**
    - orderId: A unique identifier for the order.
    - reservationId: The identifier of the reservation associated with the order (if any).
    - orderDate: The date and time when the order was placed.
    - orderDetails: Details of the items ordered (e.g., food items, quantities).
- **Role:** Tracks customer orders, including the items ordered, associated reservation, and order timing. This is essential for processing and fulfilling customer requests accurately.

MenuItem

- **Represents:** An item available for purchase from the restaurant's menu.
- **Attributes:**
    - itemId: A unique identifier for the menu item.
    - itemName: The name of the menu item.
    - price: The price of the menu item.
    - description: A description of the menu item.
- **Role:** Maintains information about the items available on the restaurant’s menu, including their names, descriptions, and prices. This helps in managing and updating the restaurant's offerings.

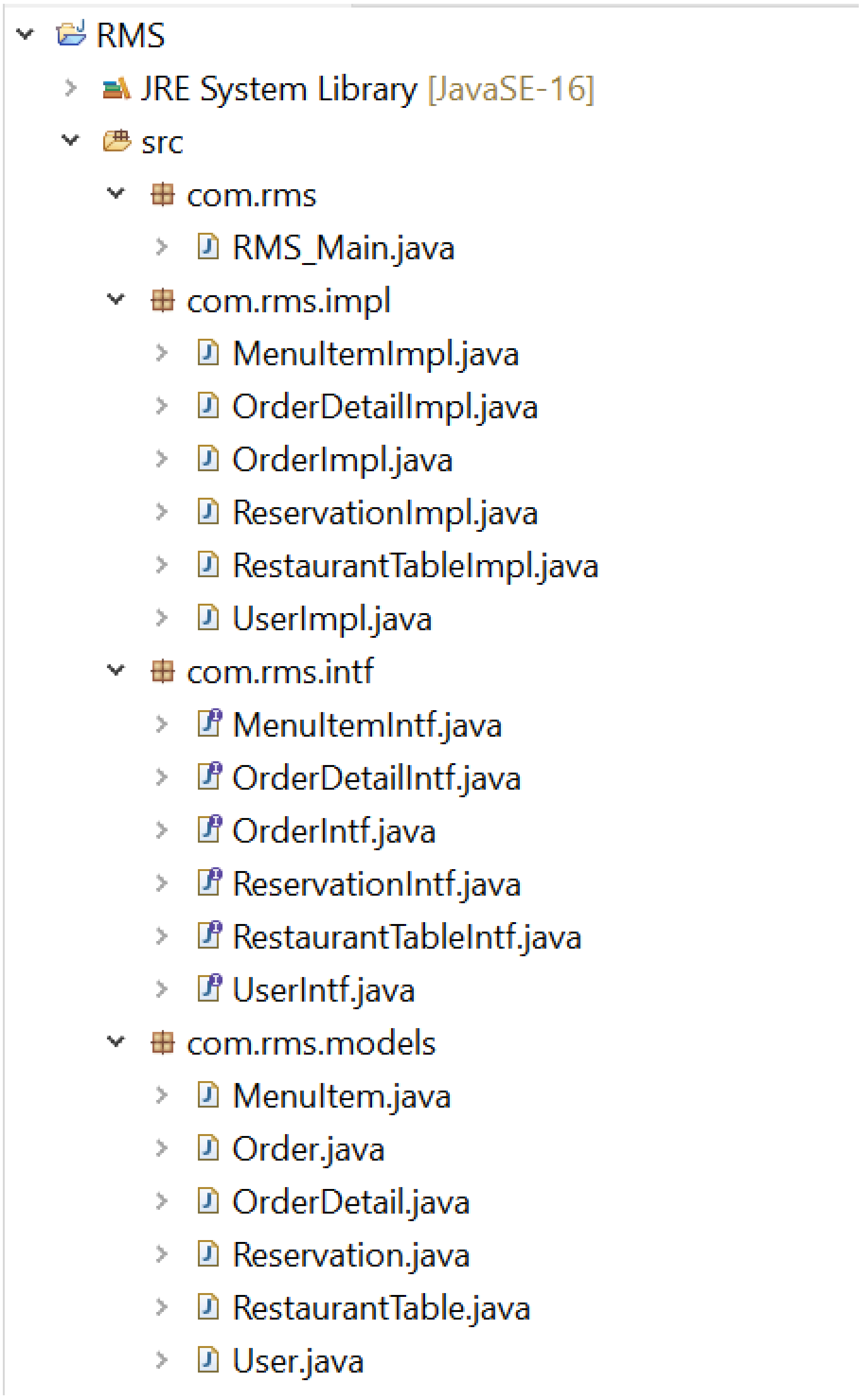
OrderDetail

- **Represents:** The details of a specific item within an order.
- **Attributes:**
  - orderDetailId: A unique identifier for the order detail.
  - orderId: The identifier of the order to which this detail belongs.
  - itemId: The identifier of the menu item.
  - quantity: The quantity of the item ordered.
- **Role:** Provides detailed information about individual items within an order, helping to track what items were ordered and in what quantities.

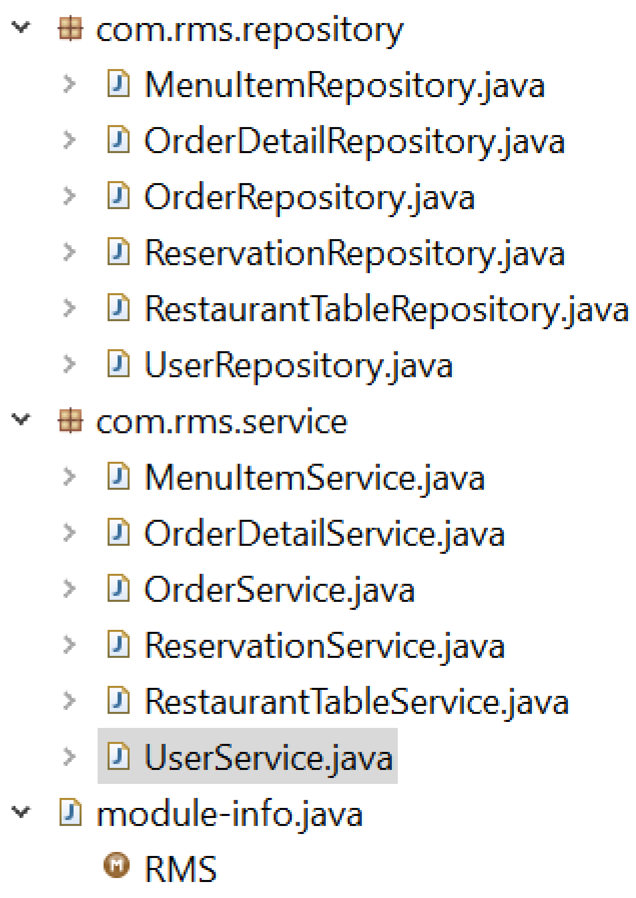
User

- **Represents:** A customer or staff member interacting with the restaurant system.
- **Attributes:**
  - userId: A unique identifier for the user.
  - name: The name of the user.
  - role: The role of the user (e.g., customer, staff member).
- **Role:** Tracks user information and roles, facilitating interactions within the system, such as making reservations, placing orders, and managing restaurant operations.

Project Structure



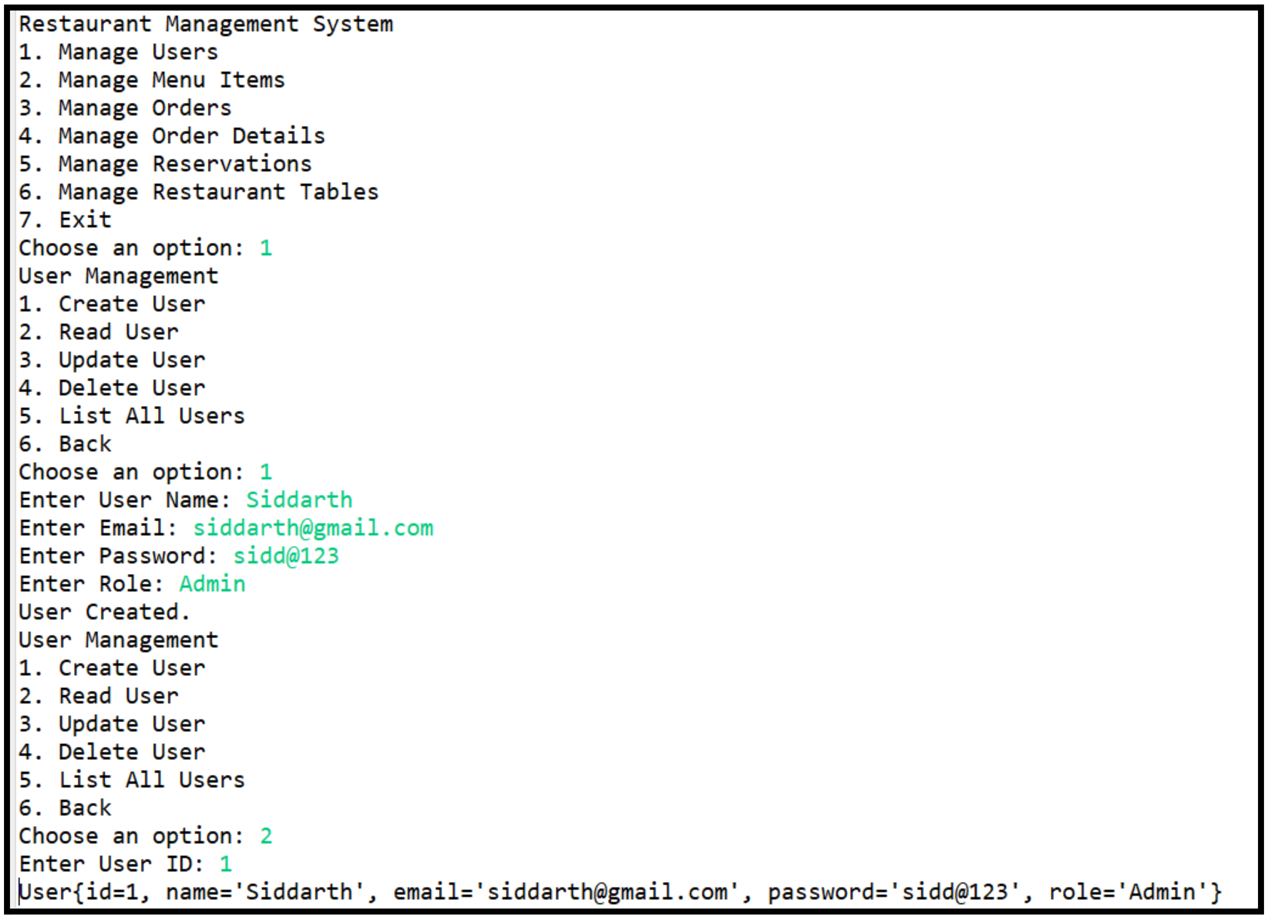
```
RMS
  JRE System Library [JavaSE-16]
  src
    com.rms
      RMS_Main.java
    com.rms.impl
      MenuItemImpl.java
      OrderDetailImpl.java
      OrderImpl.java
      ReservationImpl.java
      RestaurantTableImpl.java
      UserImpl.java
    com.rms.intf
      MenuItemIntf.java
      OrderDetailIntf.java
      OrderIntf.java
      ReservationIntf.java
      RestaurantTableIntf.java
      UserIntf.java
    com.rms.models
      MenuItem.java
      Order.java
      OrderDetail.java
      Reservation.java
      RestaurantTable.java
      User.java
```



```
com.rms.repository
    MenuItemRepository.java
    OrderDetailRepository.java
    OrderRepository.java
    ReservationRepository.java
    RestaurantTableRepository.java
    UserRepository.java
com.rms.service
    MenuItemService.java
    OrderDetailService.java
    OrderService.java
    ReservationService.java
    RestaurantTableService.java
    UserService.java
module-info.java
    RMS
```

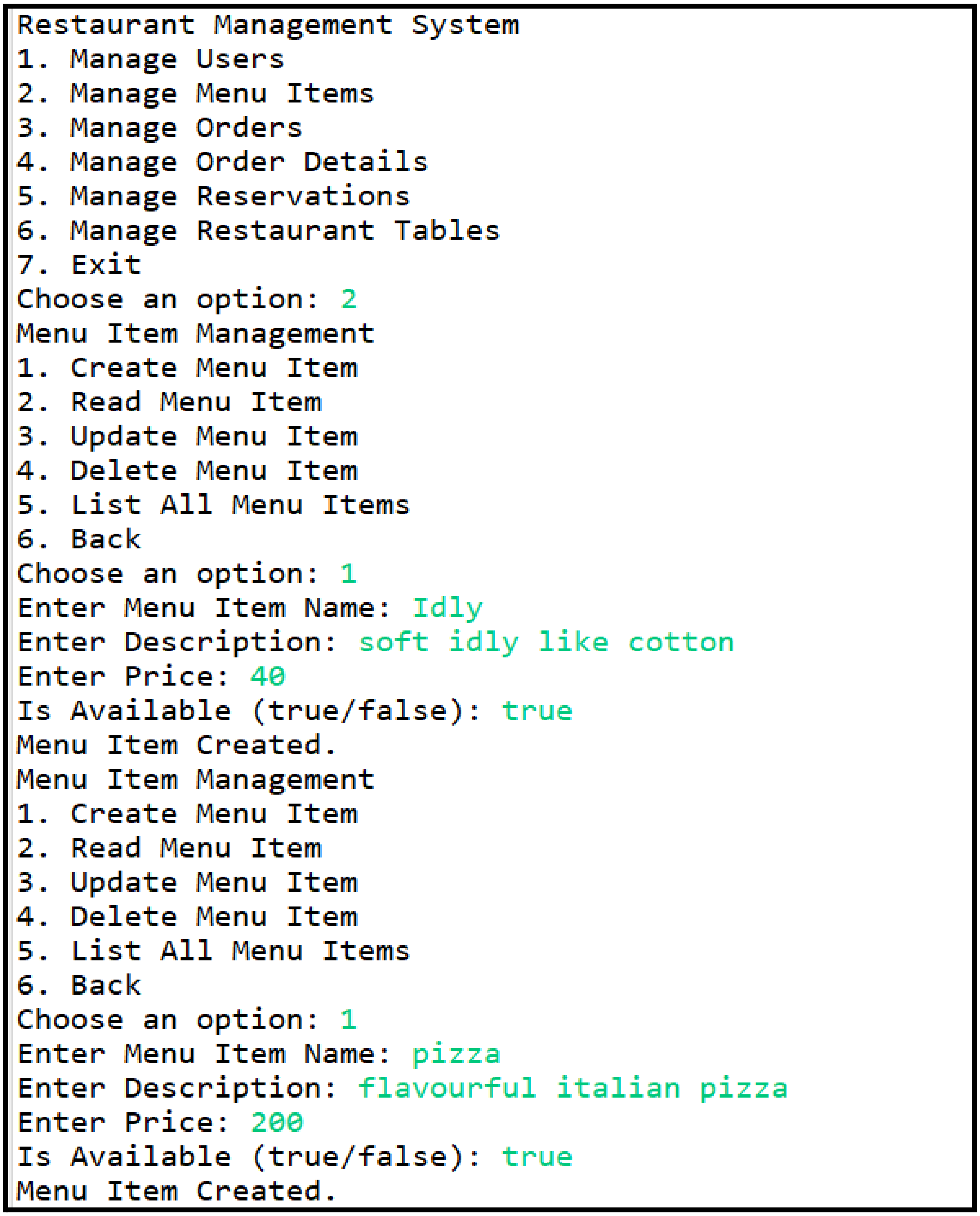
Console ScreenShot:

Managing users:

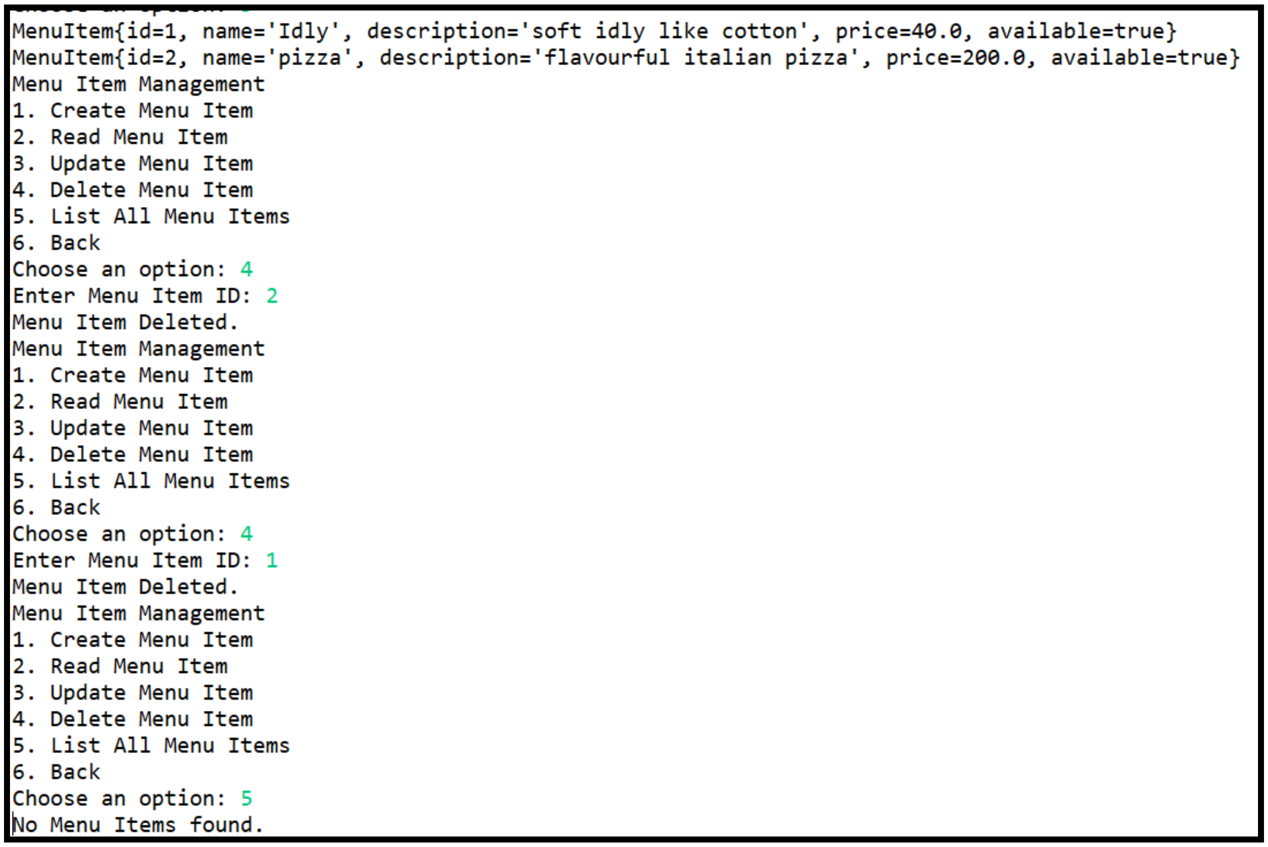


```
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 1
User Management
1. Create User
2. Read User
3. Update User
4. Delete User
5. List All Users
6. Back
Choose an option: 1
Enter User Name: Siddarth
Enter Email: siddarth@gmail.com
Enter Password: sidd@123
Enter Role: Admin
User Created.
User Management
1. Create User
2. Read User
3. Update User
4. Delete User
5. List All Users
6. Back
Choose an option: 2
Enter User ID: 1
User{id=1, name='Siddarth', email='siddarth@gmail.com', password='sidd@123', role='Admin'}
```

Managing menu items:

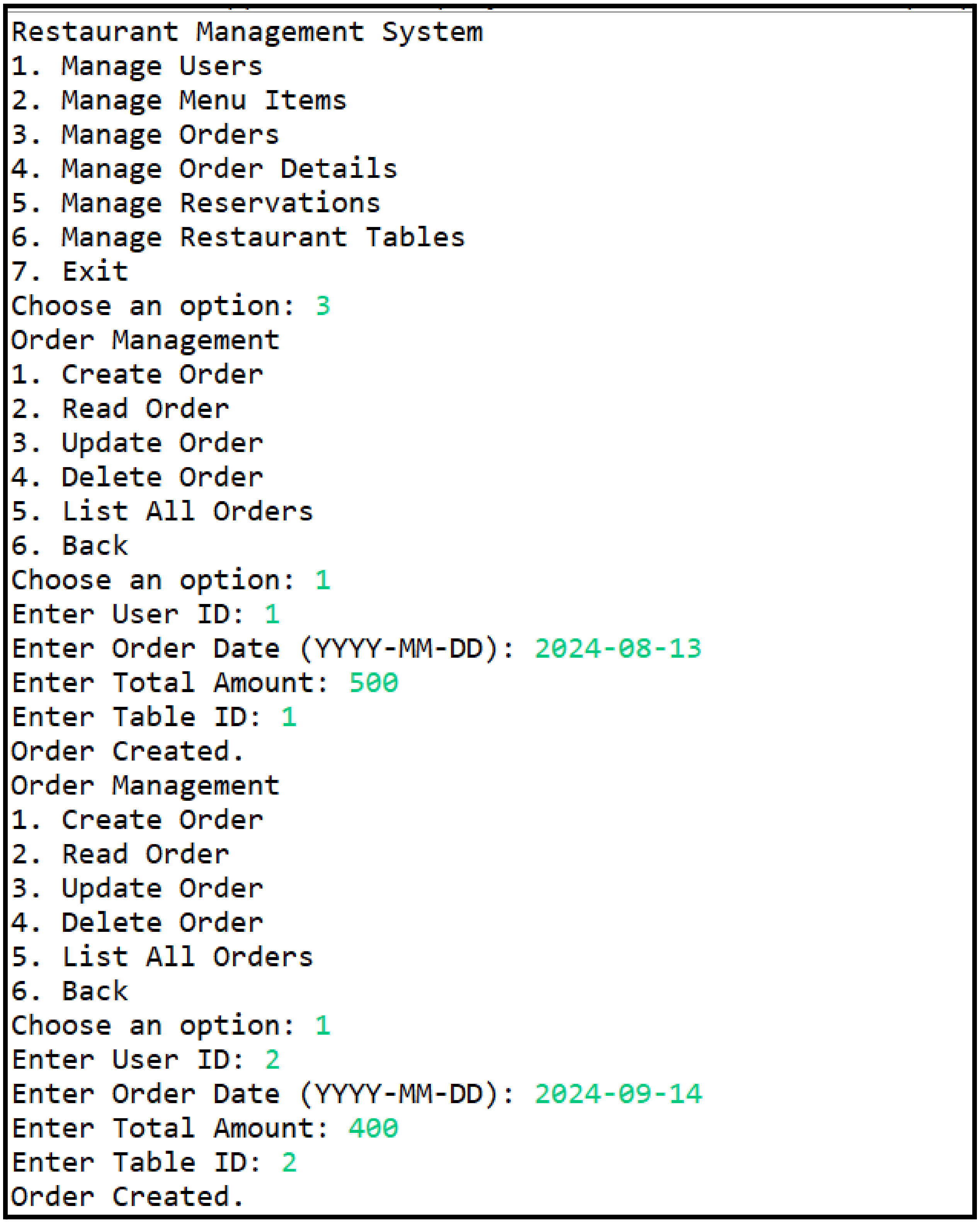


```
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 2
Menu Item Management
1. Create Menu Item
2. Read Menu Item
3. Update Menu Item
4. Delete Menu Item
5. List All Menu Items
6. Back
Choose an option: 1
Enter Menu Item Name: Idly
Enter Description: soft idly like cotton
Enter Price: 40
Is Available (true/false): true
Menu Item Created.
Menu Item Management
1. Create Menu Item
2. Read Menu Item
3. Update Menu Item
4. Delete Menu Item
5. List All Menu Items
6. Back
Choose an option: 1
Enter Menu Item Name: pizza
Enter Description: flavourful italian pizza
Enter Price: 200
Is Available (true/false): true
Menu Item Created.
```

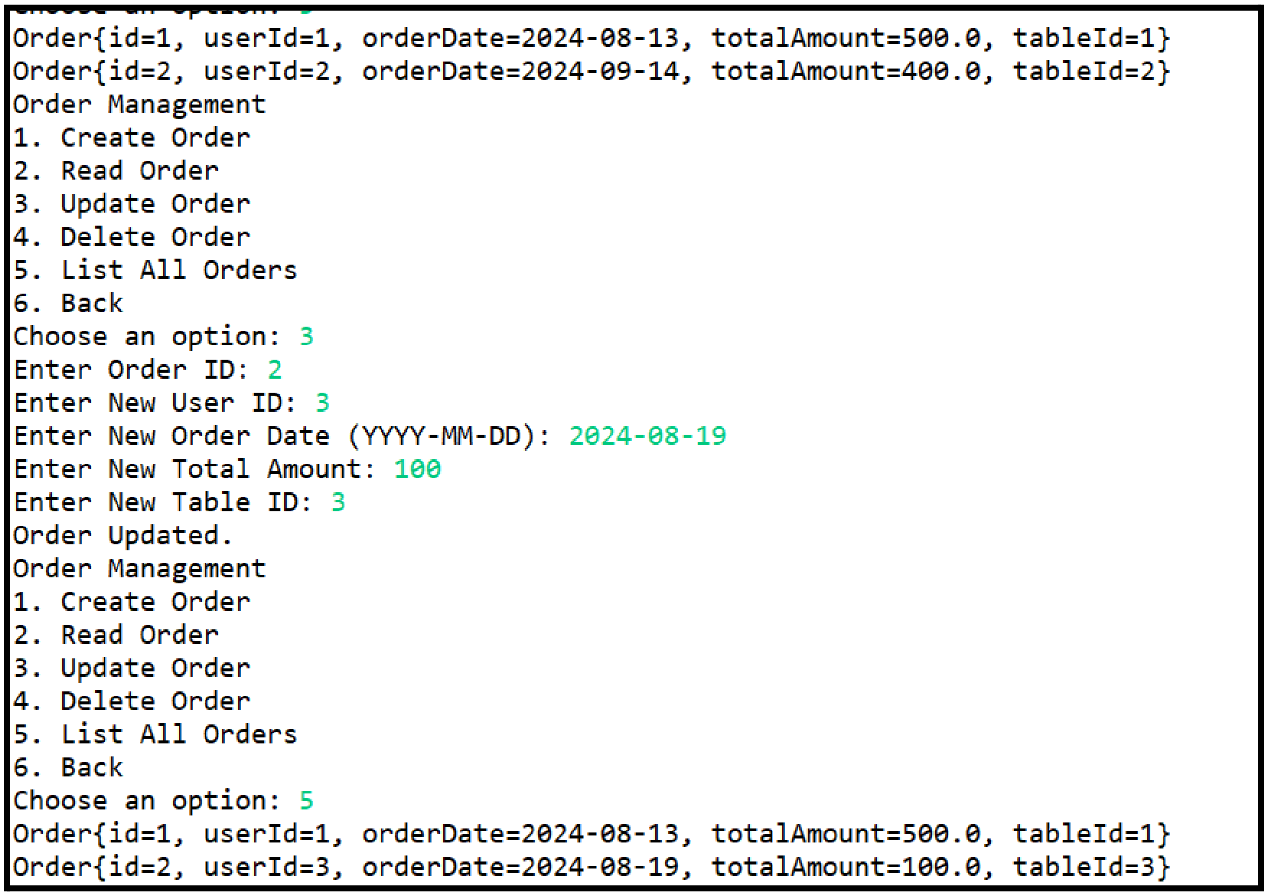


```
MenuItem{id=1, name='Idly', description='soft idly like cotton', price=40.0, available=true}
MenuItem{id=2, name='pizza', description='flavourful italian pizza', price=200.0, available=true}
Menu Item Management
1. Create Menu Item
2. Read Menu Item
3. Update Menu Item
4. Delete Menu Item
5. List All Menu Items
6. Back
Choose an option: 4
Enter Menu Item ID: 2
Menu Item Deleted.
Menu Item Management
1. Create Menu Item
2. Read Menu Item
3. Update Menu Item
4. Delete Menu Item
5. List All Menu Items
6. Back
Choose an option: 4
Enter Menu Item ID: 1
Menu Item Deleted.
Menu Item Management
1. Create Menu Item
2. Read Menu Item
3. Update Menu Item
4. Delete Menu Item
5. List All Menu Items
6. Back
Choose an option: 5
No Menu Items found.
```

Managing orders:

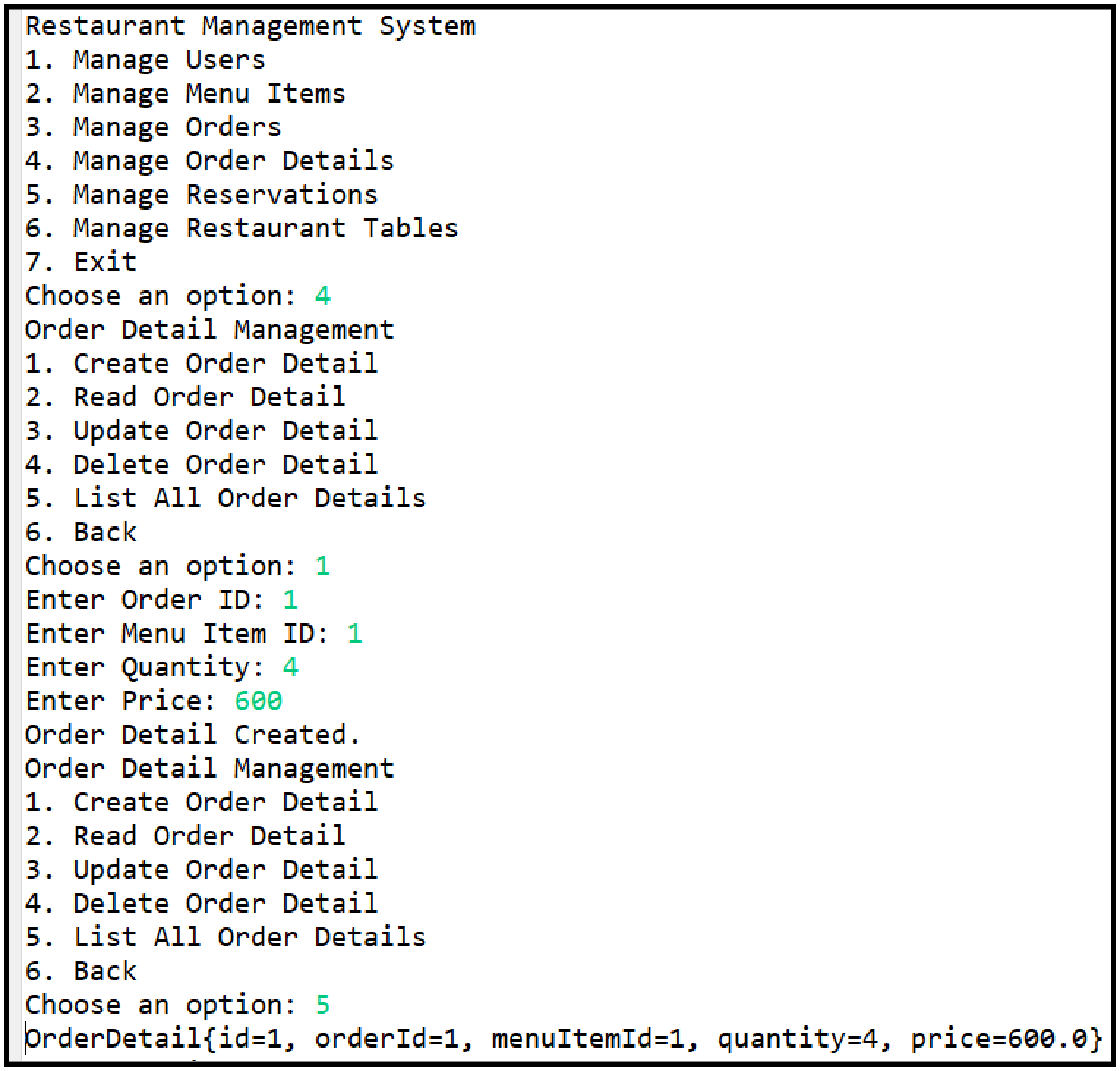


```
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 3
Order Management
1. Create Order
2. Read Order
3. Update Order
4. Delete Order
5. List All Orders
6. Back
Choose an option: 1
Enter User ID: 1
Enter Order Date (YYYY-MM-DD): 2024-08-13
Enter Total Amount: 500
Enter Table ID: 1
Order Created.
Order Management
1. Create Order
2. Read Order
3. Update Order
4. Delete Order
5. List All Orders
6. Back
Choose an option: 1
Enter User ID: 2
Enter Order Date (YYYY-MM-DD): 2024-09-14
Enter Total Amount: 400
Enter Table ID: 2
Order Created.
```



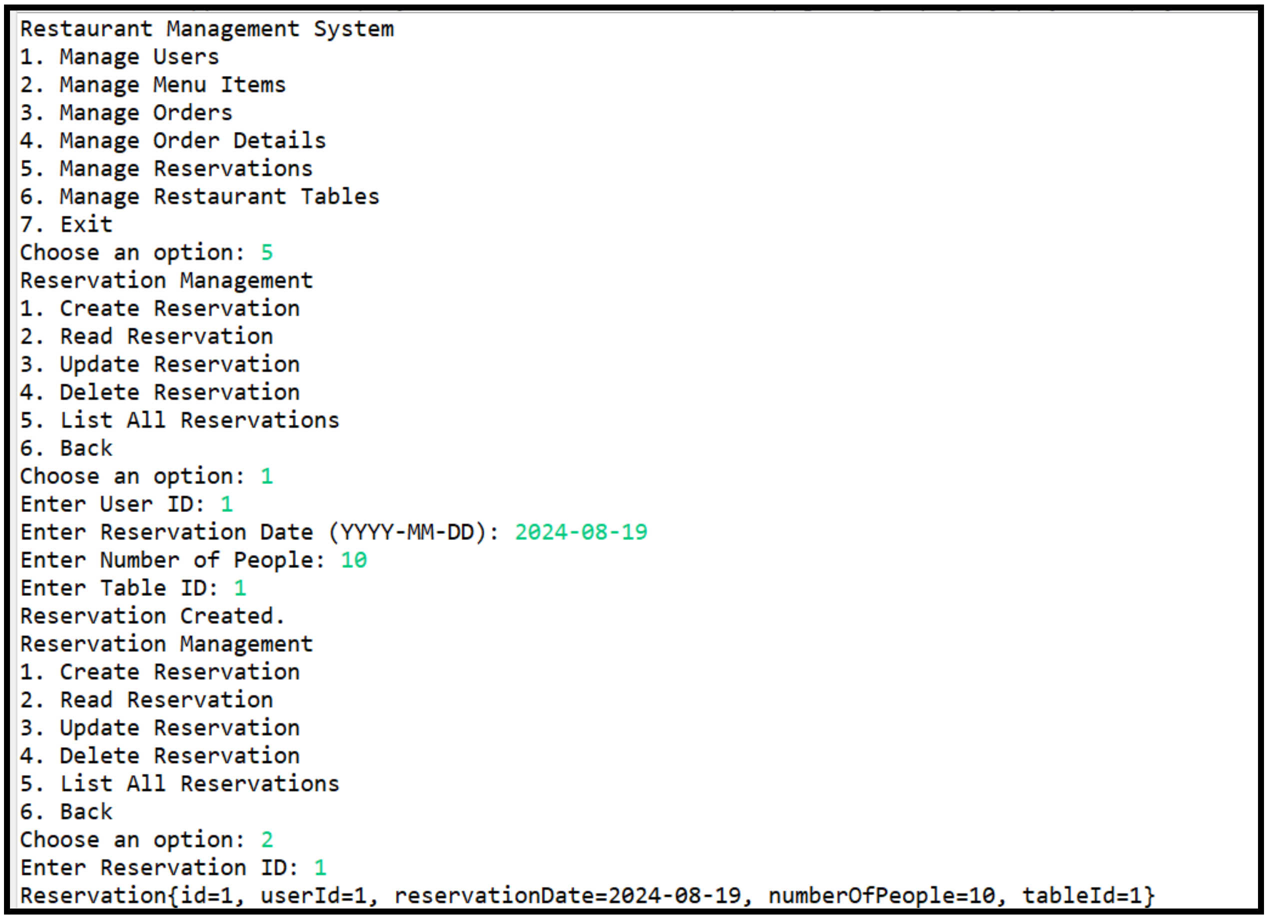
```
Order{id=1, userId=1, orderDate=2024-08-13, totalAmount=500.0, tableId=1}
Order{id=2, userId=2, orderDate=2024-09-14, totalAmount=400.0, tableId=2}
Order Management
1. Create Order
2. Read Order
3. Update Order
4. Delete Order
5. List All Orders
6. Back
Choose an option: 3
Enter Order ID: 2
Enter New User ID: 3
Enter New Order Date (YYYY-MM-DD): 2024-08-19
Enter New Total Amount: 100
Enter New Table ID: 3
Order Updated.
Order Management
1. Create Order
2. Read Order
3. Update Order
4. Delete Order
5. List All Orders
6. Back
Choose an option: 5
Order{id=1, userId=1, orderDate=2024-08-13, totalAmount=500.0, tableId=1}
Order{id=2, userId=3, orderDate=2024-08-19, totalAmount=100.0, tableId=3}
```

Managing Order Details:



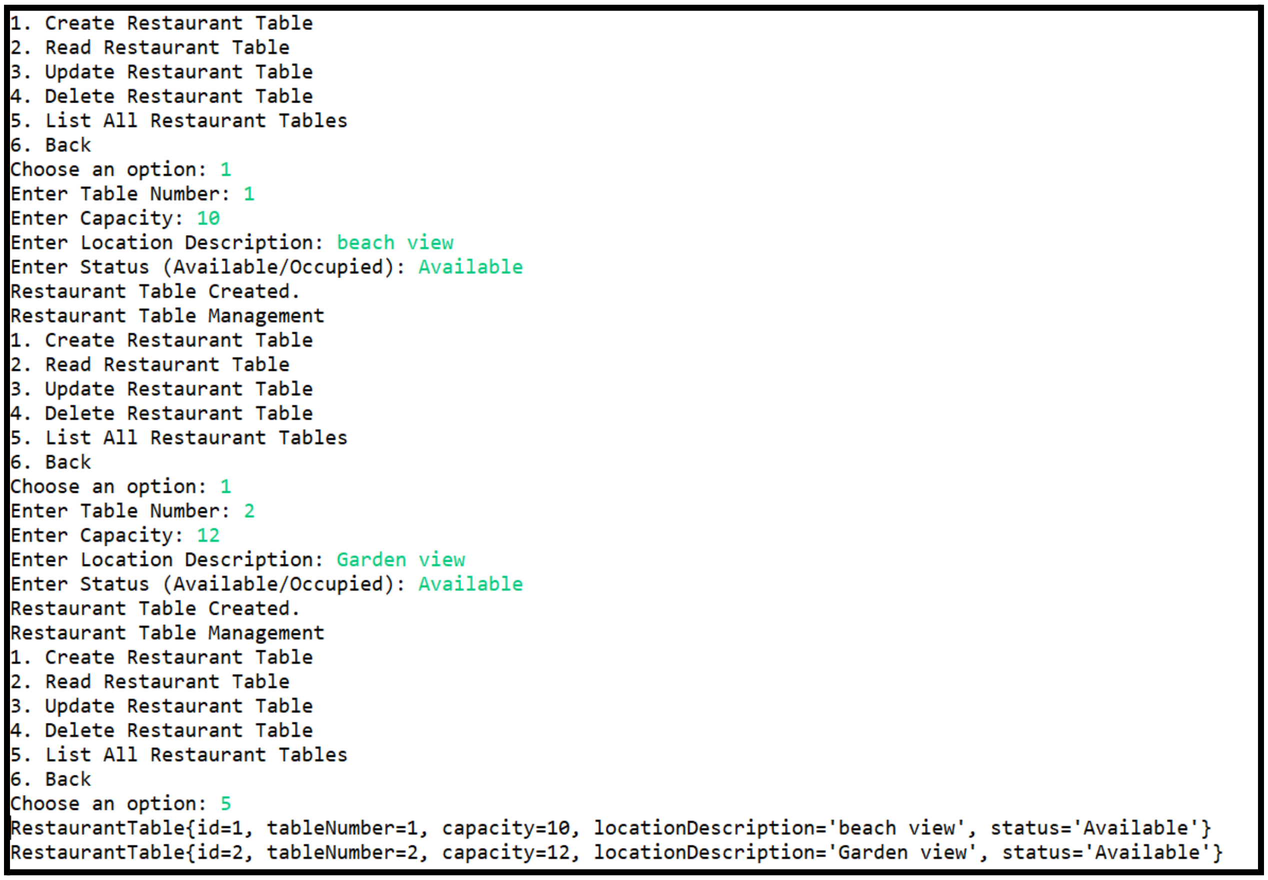
```
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 4
Order Detail Management
1. Create Order Detail
2. Read Order Detail
3. Update Order Detail
4. Delete Order Detail
5. List All Order Details
6. Back
Choose an option: 1
Enter Order ID: 1
Enter Menu Item ID: 1
Enter Quantity: 4
Enter Price: 600
Order Detail Created.
Order Detail Management
1. Create Order Detail
2. Read Order Detail
3. Update Order Detail
4. Delete Order Detail
5. List All Order Details
6. Back
Choose an option: 5
OrderDetail{id=1, orderId=1, menuItemId=1, quantity=4, price=600.0}
```

Managing Reservations:



```
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 5
Reservation Management
1. Create Reservation
2. Read Reservation
3. Update Reservation
4. Delete Reservation
5. List All Reservations
6. Back
Choose an option: 1
Enter User ID: 1
Enter Reservation Date (YYYY-MM-DD): 2024-08-19
Enter Number of People: 10
Enter Table ID: 1
Reservation Created.
Reservation Management
1. Create Reservation
2. Read Reservation
3. Update Reservation
4. Delete Reservation
5. List All Reservations
6. Back
Choose an option: 2
Enter Reservation ID: 1
Reservation{id=1, userId=1, reservationDate=2024-08-19, numberOfPeople=10, tableId=1}
```

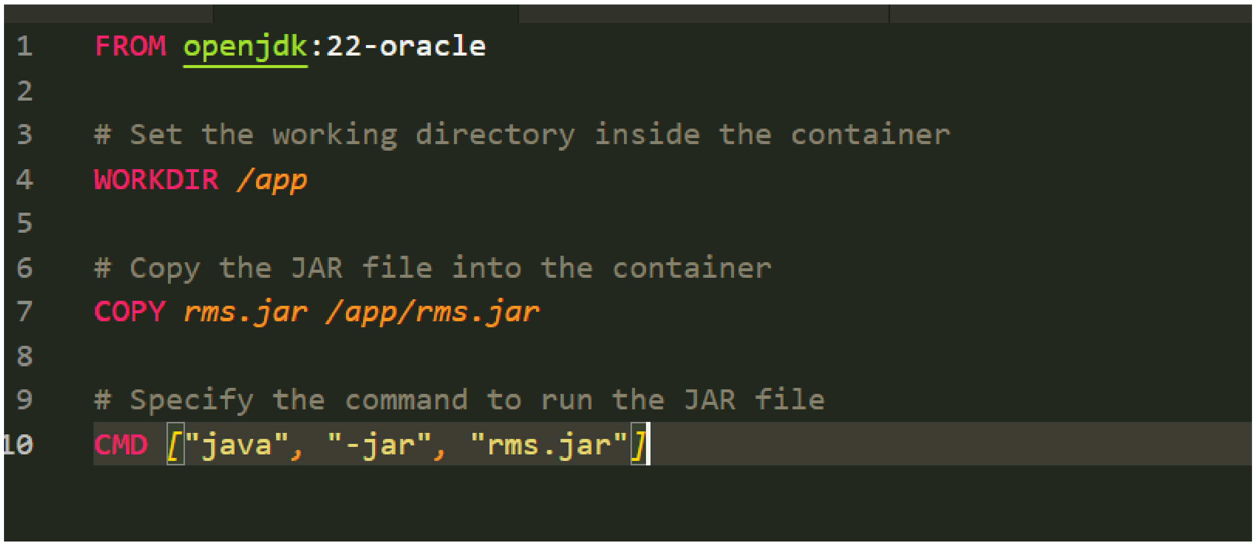
Managing Restaurant table:



```
1. Create Restaurant Table
2. Read Restaurant Table
3. Update Restaurant Table
4. Delete Restaurant Table
5. List All Restaurant Tables
6. Back
Choose an option: 1
Enter Table Number: 1
Enter Capacity: 10
Enter Location Description: beach view
Enter Status (Available/Occupied): Available
Restaurant Table Created.
Restaurant Table Management
1. Create Restaurant Table
2. Read Restaurant Table
3. Update Restaurant Table
4. Delete Restaurant Table
5. List All Restaurant Tables
6. Back
Choose an option: 1
Enter Table Number: 2
Enter Capacity: 12
Enter Location Description: Garden view
Enter Status (Available/Occupied): Available
Restaurant Table Created.
Restaurant Table Management
1. Create Restaurant Table
2. Read Restaurant Table
3. Update Restaurant Table
4. Delete Restaurant Table
5. List All Restaurant Tables
6. Back
Choose an option: 5
RestaurantTable{id=1, tableNumber=1, capacity=10, locationDescription='beach view', status='Available'}
RestaurantTable{id=2, tableNumber=2, capacity=12, locationDescription='Garden view', status='Available'}
```

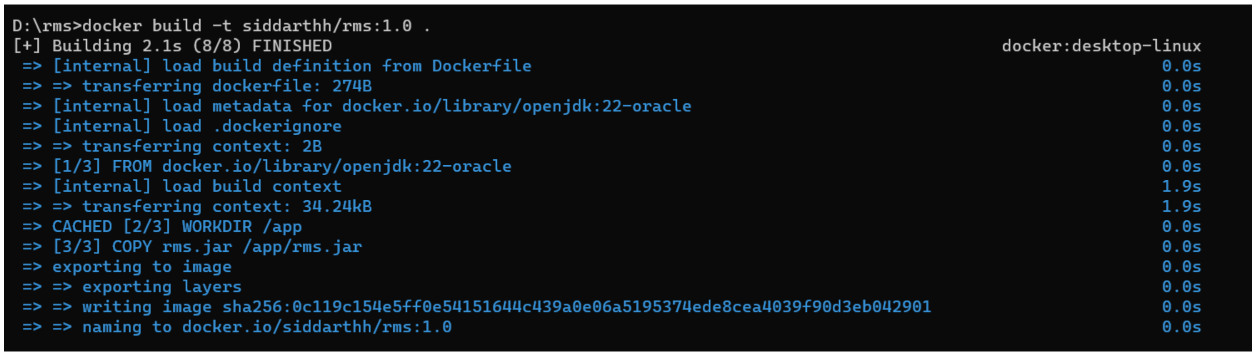
Containerizing the application using docker

1. Dockerfile



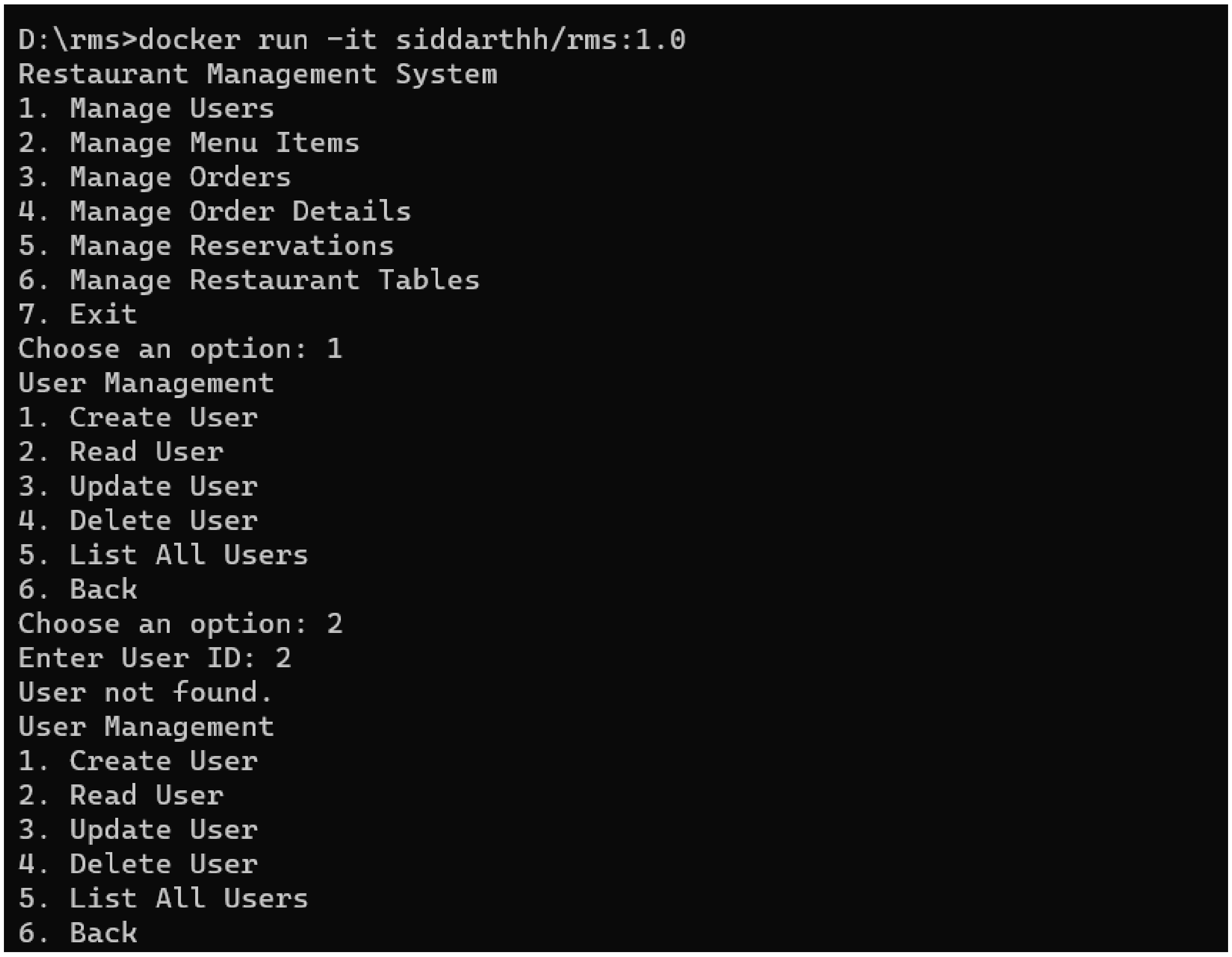
```dockerfile
FROM openjdk:22-oracle

# Set the working directory inside the container
WORKDIR /app

# Copy the JAR file into the container
COPY rms.jar /app/rms.jar

# Specify the command to run the JAR file
CMD ["java", "-jar", "rms.jar"]
```

2. Building image



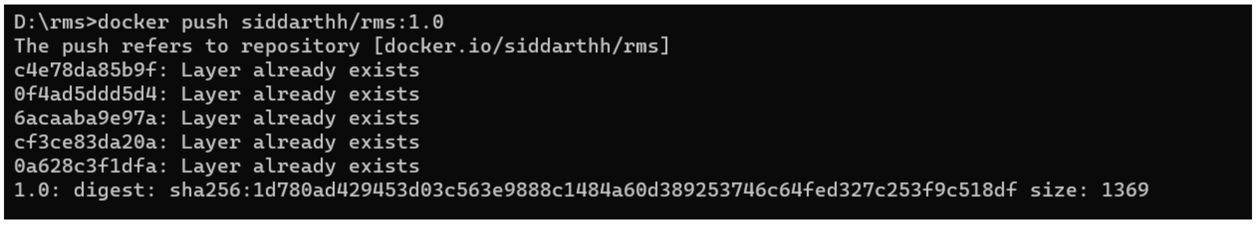
```
D:\rms>docker build -t siddarthh/rms:1.0 .
[+] Building 2.1s (8/8) FINISHED                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                            0.0s
 => => transferring dockerfile: 274B                                                                            0.0s
 => [internal] load metadata for docker.io/library/openjdk:22-oracle                                            0.0s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [1/3] FROM docker.io/library/openjdk:22-oracle                                                              0.0s
 => [internal] load build context                                                                               1.9s
 => => transferring context: 34.24kB                                                                            1.9s
 => CACHED [2/3] WORKDIR /app                                                                                   0.0s
 => [3/3] COPY rms.jar /app/rms.jar                                                                             0.0s
 => exporting to image                                                                                          0.0s
 => => exporting layers                                                                                         0.0s
 => => writing image sha256:0c119c154e5ff0e54151644c439a0e06a5195374ede8cea4039f90d3eb042901                    0.0s
 => => naming to docker.io/siddarthh/rms:1.0                                                                    0.0s
```

3. Running image

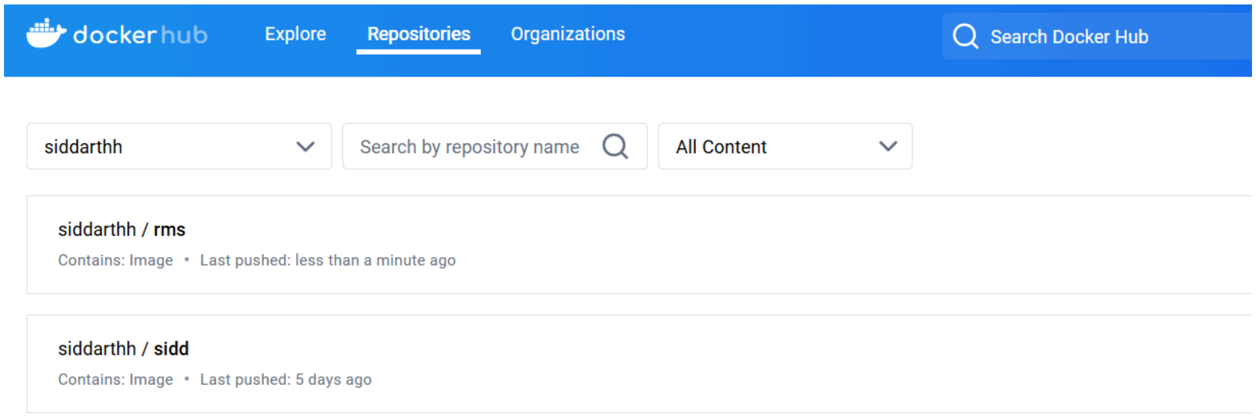


```
D:\rms>docker run -it siddarthh/rms:1.0
Restaurant Management System
1. Manage Users
2. Manage Menu Items
3. Manage Orders
4. Manage Order Details
5. Manage Reservations
6. Manage Restaurant Tables
7. Exit
Choose an option: 1
User Management
1. Create User
2. Read User
3. Update User
4. Delete User
5. List All Users
6. Back
Choose an option: 2
Enter User ID: 2
User not found.
User Management
1. Create User
2. Read User
3. Update User
4. Delete User
5. List All Users
6. Back
```

4. Creating tag and push the image to remote repository



```
D:\rms>docker push siddarthh/rms:1.0
The push refers to repository [docker.io/siddarthh/rms]
c4e78da85b9f: Layer already exists
0f4ad5ddd5d4: Layer already exists
6acaaba9e97a: Layer already exists
cf3ce83da20a: Layer already exists
0a628c3f1dfa: Layer already exists
1.0: digest: sha256:1d780ad429453d03c563e9888c1484a60d389253746c64fed327c253f9c518df size: 1369
```



siddarthh / rms
Contains: Image • Last pushed: less than a minute ago

siddarthh / sidd
Contains: Image • Last pushed: 5 days ago

Code:

RMS_Main.java

```
package com.rms;

import com.rms.service.*;

import java.util.Scanner;

public class RMS_Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    UserService userService = new UserService();
    MenuItemService menuItemService = new MenuItemService();
    OrderService orderService = new OrderService();
    OrderDetailService orderDetailService = new OrderDetailService();
    ReservationService reservationService = new ReservationService();
    RestaurantTableService restaurantTableService = new
RestaurantTableService();

    while (true) {
      System.out.println("Restaurant Management System");
      System.out.println("1. Manage Users");
      System.out.println("2. Manage Menu Items");
      System.out.println("3. Manage Orders");
      System.out.println("4. Manage Order Details");
      System.out.println("5. Manage Reservations");
      System.out.println("6. Manage Restaurant Tables");
      System.out.println("7. Exit");
      System.out.print("Choose an option: ");

      int choice = scanner.nextInt();
      scanner.nextLine(); // consume newline

      switch (choice) {
        case 1:
          userService.manageUsers(scanner);
          break;
        case 2:
          menuItemService.manageMenuItems(scanner);
          break;
        case 3:
          orderService.manageOrders(scanner);
          break;
        case 4:
          orderDetailService.manageOrderDetails(scanner);
          break;
        case 5:
          reservationService.manageReservations(scanner);
          break;
        case 6:
          restaurantTableService.manageRestaurantTables(scanner);
          break;
        case 7:
          System.out.println("Exiting...");
          scanner.close();
```

```
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
      }
    }
}
```

Interfaces

intf/MenuItemIntf.java

```
package com.rms.intf;

import com.rms.models.MenuItem;

import java.util.List;

public interface MenuItemIntf {
    void createMenuItem(MenuItem menuItem);
    MenuItem readMenuItem(int id);
    void updateMenuItem(MenuItem menuItem);
    void deleteMenuItem(int id);
    List<MenuItem> getAllMenuItems();
}
```

intf/OrderDetailIntf.java

```
package com.rms.intf;

import com.rms.models.OrderDetail;

import java.util.List;
```

```
public interface OrderDetailIntf {
   void createOrderDetail(OrderDetail orderDetail);
   OrderDetail readOrderDetail(int id);
   void updateOrderDetail(OrderDetail orderDetail);
   void deleteOrderDetail(int id);
   List<OrderDetail> getAllOrderDetails();
}
```

intf/OrderIntf.java

```
package com.rms.intf;

import com.rms.models.Order;

import java.util.List;

public interface OrderIntf {
   void createOrder(Order order);
   Order readOrder(int id);
   void updateOrder(Order order);
   void deleteOrder(int id);
   List<Order> getAllOrders();
}
```

intf/ReservationIntf.java

```
package com.rms.intf;

import com.rms.models.Reservation;

import java.util.List;
```

```
public interface ReservationIntf {
    void createReservation(Reservation reservation);
    Reservation readReservation(int id);
    void updateReservation(Reservation reservation);
    void deleteReservation(int id);
    List<Reservation> getAllReservations();
}
```

intf/RestaurantTableIntf.java

```
package com.rms.intf;

import com.rms.models.RestaurantTable;

import java.util.List;

public interface RestaurantTableIntf {
    void createRestaurantTable(RestaurantTable table);
    RestaurantTable readRestaurantTable(int id);
    void updateRestaurantTable(RestaurantTable table);
    void deleteRestaurantTable(int id);
    List<RestaurantTable> getAllRestaurantTables();
}
```

intf/UserIntf.java

```
package com.rms.intf;

import com.rms.models.User;

import java.util.List;
```

```
public interface UserIntf {
  void createUser(User user);
  User readUser(int id);
  void updateUser(User user);
  void deleteUser(int id);
  List<User> getAllUsers();
}
```

Implementation:

impl/MenuItemImpl.java

```
package com.rms.impl;

import com.rms.intf.MenuItemIntf;
import com.rms.models.MenuItem;
import com.rms.repository.MenuItemRepository;

import java.util.List;

public class MenuItemImpl implements MenuItemIntf {
  private MenuItemRepository repository = new MenuItemRepository();

  @Override
  public void createMenuItem(MenuItem menuItem) {
    repository.add(menuItem);
  }

  @Override
  public MenuItem readMenuItem(int id) {
    return repository.get(id);
  }
  @Override
  public void updateMenuItem(MenuItem menuItem) {
    repository.update(menuItem);
  }

  @Override
  public void deleteMenuItem(int id) {
    repository.remove(id);
  }

  @Override
  public List<MenuItem> getAllMenuItems() {
    return repository.getAll();
  }
}
```

impl/OrderDetailImpl.java

```
package com.rms.impl;

import com.rms.intf.OrderDetailIntf;
import com.rms.models.OrderDetail;
import com.rms.repository.OrderDetailRepository;

import java.util.List;

public class OrderDetailImpl implements OrderDetailIntf {
  private OrderDetailRepository repository = new OrderDetailRepository();

  @Override
  public void createOrderDetail(OrderDetail orderDetail) {
    repository.add(orderDetail);
  }
```

```
    @Override
    public OrderDetail readOrderDetail(int id) {
        return repository.get(id);
    }

    @Override
    public void updateOrderDetail(OrderDetail orderDetail) {
        repository.update(orderDetail);
    }

    @Override
    public void deleteOrderDetail(int id) {
        repository.remove(id);
    }

    @Override
    public List<OrderDetail> getAllOrderDetails() {
        return repository.getAll();
    }
}
```

impl/OrderImpl.java

```
package com.rms.impl;

import com.rms.intf.OrderIntf;
import com.rms.models.Order;
import com.rms.repository.OrderRepository;

import java.util.List;

public class OrderImpl implements OrderIntf {
    private OrderRepository repository = new OrderRepository();

    @Override
    public void createOrder(Order order) {
        repository.add(order);
    }

    @Override
    public Order readOrder(int id) {
        return repository.get(id);
    }

    @Override
    public void updateOrder(Order order) {
        repository.update(order);
    }

    @Override
    public void deleteOrder(int id) {
        repository.remove(id);
    }

    @Override
    public List<Order> getAllOrders() {
        return repository.getAll();
    }
}
```

impl/ReservationImpl.java

```
package com.rms.impl;

import com.rms.intf.ReservationIntf;
import com.rms.models.Reservation;
import com.rms.repository.ReservationRepository;

import java.util.List;

public class ReservationImpl implements ReservationIntf {
  private ReservationRepository repository = new ReservationRepository();

  @Override
  public void createReservation(Reservation reservation) {
    repository.add(reservation);
  }

  @Override
  public Reservation readReservation(int id) {
    return repository.get(id);
  }

  @Override
  public void updateReservation(Reservation reservation) {
    repository.update(reservation);
  }

  @Override
  public void deleteReservation(int id) {
    repository.remove(id);
  }

  @Override
  public List<Reservation> getAllReservations() {
    return repository.getAll();
  }
}
```

impl/RestaurantTableImpl.java

```
package com.rms.impl;

import com.rms.intf.RestaurantTableIntf;
import com.rms.models.RestaurantTable;
import com.rms.repository.RestaurantTableRepository;

import java.util.List;

public class RestaurantTableImpl implements RestaurantTableIntf {
  private RestaurantTableRepository repository = new RestaurantTableRepository();

  @Override
  public void createRestaurantTable(RestaurantTable table) {
    repository.add(table);
  }

  @Override
  public RestaurantTable readRestaurantTable(int id) {
    return repository.get(id);
  }

  @Override
  public void updateRestaurantTable(RestaurantTable table) {
    repository.update(table);
  }

  @Override
  public void deleteRestaurantTable(int id) {
    repository.remove(id);
  }

  @Override
  public List<RestaurantTable> getAllRestaurantTables() {
    return repository.getAll();
  }
}
```

impl/UserImpl.java

```
package com.rms.impl;

import com.rms.intf.UserIntf;
import com.rms.models.User;
import com.rms.repository.UserRepository;

import java.util.List;

public class UserImpl implements UserIntf {
  private UserRepository repository = new UserRepository();

  @Override
  public void createUser(User user) {
    repository.add(user);
  }

  @Override
  public User readUser(int id) {
    return repository.get(id);
  }

  @Override
  public void updateUser(User user) {
    repository.update(user);
  }
```

```
    @Override
    public void deleteUser(int id) {
        repository.remove(id);
    }

    @Override
    public List<User> getAllUsers() {
        return repository.getAll();
    }
}
```

Models:

models/MenuItem.java

```
package com.rms.models;

public class MenuItem {
    private int id;
    private String name;
    private String description;
    private double price;
    private boolean available;

    public MenuItem(int id, String name, String description, double price, boolean
available) {
        this.id = id;
        this.name = name;
        this.description = description;
        this.price = price;
        this.available = available;
    }

    public int getId() {
```

```
  return id;
}

public void setId(int id) {
  this.id = id;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String getDescription() {
  return description;
}

public void setDescription(String description) {
  this.description = description;
}

public double getPrice() {
  return price;
}

public void setPrice(double price) {
  this.price = price;
}

public boolean isAvailable() {
  return available;
}
```

```
  public void setAvailable(boolean available) {
    this.available = available;
  }

  @Override
  public String toString() {
    return "MenuItem{" +
        "id=" + id +
        ", name='" + name + '\'' +
        ", description='" + description + '\'' +
        ", price=" + price +
        ", available=" + available +
        '}';
  }
}
```

models/Order.java

```
package com.rms.models;

import java.util.Date;

public class Order {
  private int id;
  private int userId;
  private Date orderDate;
  private double totalAmount;
  private int tableId;

  public Order(int id, int userId, Date orderDate, double totalAmount, int tableId) {
    this.id = id;
    this.userId = userId;
    this.orderDate = orderDate;
    this.totalAmount = totalAmount;
    this.tableId = tableId;
  }

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public int getUserId() {
    return userId;
  }

  public void setUserId(int userId) {
    this.userId = userId;
  }

  public Date getOrderDate() {
    return orderDate;
  }

  public void setOrderDate(Date orderDate) {
    this.orderDate = orderDate;
  }

  public double getTotalAmount() {
    return totalAmount;
  }

  public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }

    public int getTableId() {
        return tableId;
    }

    public void setTableId(int tableId) {
        this.tableId = tableId;
    }

    @Override
    public String toString() {
        return "Order{" +
                "id=" + id +
                ", userId=" + userId +
                ", orderDate=" + orderDate +
                ", totalAmount=" + totalAmount +
                ", tableId=" + tableId +
                '}';
    }
}
```

models/OrderDetail.java

```
package com.rms.models;

public class OrderDetail {
    private int id;
    private int orderId;
    private int menuItemId;
    private int quantity;
    private double price;
```

```
public OrderDetail(int id, int orderId, int menuItemId, int quantity, double price) {
  this.id = id;
  this.orderId = orderId;
  this.menuItemId = menuItemId;
  this.quantity = quantity;
  this.price = price;
}

public int getId() {
  return id;
}

public void setId(int id) {
  this.id = id;
}

public int getOrderId() {
  return orderId;
}

public void setOrderId(int orderId) {
  this.orderId = orderId;
}

public int getMenuItemId() {
  return menuItemId;
}

public void setMenuItemId(int menuItemId) {
  this.menuItemId = menuItemId;
}

public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @Override
    public String toString() {
        return "OrderDetail{" +
                "id=" + id +
                ", orderId=" + orderId +
                ", menuItemId=" + menuItemId +
                ", quantity=" + quantity +
                ", price=" + price +
                '}';
    }
}
```

models/Reservation.java

```
package com.rms.models;

import java.util.Date;
```

```
public class Reservation {
  private int id;
  private int userId;
  private Date reservationDate;
  private int numberOfPeople;
  private int tableId;

  // Default constructor
  public Reservation() {}

  // Constructor with all fields
  public Reservation(int id, int userId, Date reservationDate, int numberOfPeople, int
tableId) {
    this.id = id;
    this.userId = userId;
    this.reservationDate = reservationDate;
    this.numberOfPeople = numberOfPeople;
    this.tableId = tableId;
  }

  // Getters and setters
  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public int getUserId() {
    return userId;
  }

  public void setUserId(int userId) {
    this.userId = userId;
  }

  public Date getReservationDate() {
    return reservationDate;
  }

  public void setReservationDate(Date reservationDate) {
    this.reservationDate = reservationDate;
  }

  public int getNumberOfPeople() {
    return numberOfPeople;
  }

  public void setNumberOfPeople(int numberOfPeople) {
    this.numberOfPeople = numberOfPeople;
  }

  public int getTableId() {
    return tableId;
  }

  public void setTableId(int tableId) {
    this.tableId = tableId;
  }

  @Override
  public String toString() {
    return "Reservation{" +
            "id=" + id +
            ", userId=" + userId +
            ", reservationDate=" + reservationDate +
            ", numberOfPeople=" + numberOfPeople +
            ", tableId=" + tableId +
            '}';
    }
  }
```

models/RestaurantTable.java

```
package com.rms.models;

public class RestaurantTable {
    private int id;
    private int tableNumber;
    private int capacity;
    private String locationDescription;
    private String status;

    // Default constructor
    public RestaurantTable() {}

    // Constructor with all fields
    public RestaurantTable(int id, int tableNumber, int capacity, String
locationDescription, String status) {
        this.id = id;
        this.tableNumber = tableNumber;
        this.capacity = capacity;
        this.locationDescription = locationDescription;
        this.status = status;
    }

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
```

```
    this.id = id;
}

public int getTableNumber() {
    return tableNumber;
}

public void setTableNumber(int tableNumber) {
    this.tableNumber = tableNumber;
}

public int getCapacity() {
    return capacity;
}

public void setCapacity(int capacity) {
    this.capacity = capacity;
}

public String getLocationDescription() {
    return locationDescription;
}

public void setLocationDescription(String locationDescription) {
    this.locationDescription = locationDescription;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}
```

```
  @Override
  public String toString() {
    return "RestaurantTable{" +
        "id=" + id +
        ", tableNumber=" + tableNumber +
        ", capacity=" + capacity +
        ", locationDescription='" + locationDescription + '\'' +
        ", status='" + status + '\'' +
        '}';
  }
}
```

models/User.java

```
package com.rms.models;

public class User {
  private int id;
  private String name;
  private String email;
  private String password;
  private String role;

  public User(int id, String name, String email, String password, String role) {
    this.id = id;
    this.name = name;
    this.email = email;
    this.password = password;
    this.role = role;
  }

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", password='" + password + '\'' +
                ", role='" + role + '\'' +
                '}';
    }
}
```

Repository:

repository/MenuItemRepository.java

```
package com.rms.repository;

import com.rms.models.MenuItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
public class MenuItemRepository {
  private Map<Integer, MenuItem> menuItemMap = new HashMap<>();
  private static int idCounter = 1;

  public void add(MenuItem menuItem) {
    menuItem.setId(idCounter++);
    menuItemMap.put(menuItem.getId(), menuItem);
  }

  public MenuItem get(int id) {
    return menuItemMap.get(id);
  }

  public void update(MenuItem menuItem) {
    menuItemMap.put(menuItem.getId(), menuItem);
  }

  public void remove(int id) {
    menuItemMap.remove(id);
  }

  public List<MenuItem> getAll() {
    return new ArrayList<>(menuItemMap.values());
  }
}
```

repository/OrderDetailRepository.java

```
package com.rms.repository;

import com.rms.models.OrderDetail;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class OrderDetailRepository {
  private Map<Integer, OrderDetail> orderDetailMap = new HashMap<>();
  private static int idCounter = 1;

  public void add(OrderDetail orderDetail) {
    orderDetail.setId(idCounter++);
    orderDetailMap.put(orderDetail.getId(), orderDetail);
  }

  public OrderDetail get(int id) {
    return orderDetailMap.get(id);
  }

  public void update(OrderDetail orderDetail) {
    orderDetailMap.put(orderDetail.getId(), orderDetail);
  }

  public void remove(int id) {
    orderDetailMap.remove(id);
  }

  public List<OrderDetail> getAll() {
    return new ArrayList<>(orderDetailMap.values());
  }
}
```

repository/OrderRepository.java

```
package com.rms.repository;

import com.rms.models.Order;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class OrderRepository {
  private Map<Integer, Order> orderMap = new HashMap<>();
  private static int idCounter = 1;

  public void add(Order order) {
    order.setId(idCounter++);
    orderMap.put(order.getId(), order);
  }

  public Order get(int id) {
    return orderMap.get(id);
  }

  public void update(Order order) {
    orderMap.put(order.getId(), order);
  }

  public void remove(int id) {
    orderMap.remove(id);
  }

  public List<Order> getAll() {
    return new ArrayList<>(orderMap.values());
  }
}
```

repository/ReservationRepository.java

```
package com.rms.repository;

import com.rms.models.Reservation;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class ReservationRepository {
  private Map<Integer, Reservation> reservationMap = new HashMap<>();
  private static int idCounter = 1;

  public void add(Reservation reservation) {
    reservation.setId(idCounter++);
    reservationMap.put(reservation.getId(), reservation);
  }

  public Reservation get(int id) {
    return reservationMap.get(id);
  }

  public void update(Reservation reservation) {
    reservationMap.put(reservation.getId(), reservation);
  }

  public void remove(int id) {
    reservationMap.remove(id);
  }
```

```
    public List<Reservation> getAll() {
        return new ArrayList<>(reservationMap.values());
    }
}
```

repository/RestaurantTableRepository.java

```
package com.rms.repository;

import com.rms.models.RestaurantTable;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class RestaurantTableRepository {
    private Map<Integer, RestaurantTable> tableMap = new HashMap<>();
    private static int idCounter = 1;

    public void add(RestaurantTable table) {
        table.setId(idCounter++);
        tableMap.put(table.getId(), table);
    }

    public RestaurantTable get(int id) {
        return tableMap.get(id);
    }

    public void update(RestaurantTable table) {
        tableMap.put(table.getId(), table);
    }
    public void remove(int id) {
        tableMap.remove(id);
    }

    public List<RestaurantTable> getAll() {
        return new ArrayList<>(tableMap.values());
    }
}
```

repository/UserRepository.java

```
package com.rms.repository;

import com.rms.models.User;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class UserRepository {
    private Map<Integer, User> userMap = new HashMap<>();
    private static int idCounter = 1;

    public void add(User user) {
        user.setId(idCounter++);
        userMap.put(user.getId(), user);
    }

    public User get(int id) {
        return userMap.get(id);
    }
    public void update(User user) {
        userMap.put(user.getId(), user);
    }

    public void remove(int id) {
        userMap.remove(id);
    }

    public List<User> getAll() {
        return new ArrayList<>(userMap.values());
    }
}
```

Services

service/MenuItemService.java

```
package com.rms.service;

import com.rms.impl.MenuItemImpl;
import com.rms.models.MenuItem;

import java.util.List;
import java.util.Scanner;

public class MenuItemService {
    private MenuItemImpl menuItemImpl = new MenuItemImpl();

    public void manageMenuItems(Scanner scanner) {
        while (true) {
            System.out.println("Menu Item Management");
            System.out.println("1. Create Menu Item");
            System.out.println("2. Read Menu Item");
            System.out.println("3. Update Menu Item");
            System.out.println("4. Delete Menu Item");
            System.out.println("5. List All Menu Items");
            System.out.println("6. Back");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
              case 1:
                createMenuItem(scanner);
                break;
              case 2:
                readMenuItem(scanner);
                break;
              case 3:
                updateMenuItem(scanner);
                break;
              case 4:
                deleteMenuItem(scanner);
                break;
              case 5:
                listAllMenuItems();
                break;
              case 6:
                return;
              default:
                System.out.println("Invalid choice. Try again.");
            }
        }
    }

    private void createMenuItem(Scanner scanner) {
        System.out.print("Enter Menu Item Name: ");
        String name = scanner.nextLine();
```

```
        System.out.print("Enter Description: ");
        String description = scanner.nextLine();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        scanner.nextLine(); // consume newline
        System.out.print("Is Available (true/false): ");
        boolean available = scanner.nextBoolean();
        scanner.nextLine(); // consume newline

        MenuItem menuItem = new MenuItem(0, name, description, price, available);
        menuItemImpl.createMenuItem(menuItem);
        System.out.println("Menu Item Created.");
    }

    private void readMenuItem(Scanner scanner) {
        System.out.print("Enter Menu Item ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        MenuItem menuItem = menuItemImpl.readMenuItem(id);
        if (menuItem != null) {
            System.out.println(menuItem);
        } else {
            System.out.println("Menu Item not found.");
        }
    }

    private void updateMenuItem(Scanner scanner) {
        System.out.print("Enter Menu Item ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline

        MenuItem menuItem = menuItemImpl.readMenuItem(id);
        if (menuItem != null) {
            System.out.print("Enter New Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter New Description: ");
            String description = scanner.nextLine();
            System.out.print("Enter New Price: ");
            double price = scanner.nextDouble();
            scanner.nextLine(); // consume newline
            System.out.print("Is Available (true/false): ");
            boolean available = scanner.nextBoolean();
            scanner.nextLine(); // consume newline

            menuItem.setName(name);
            menuItem.setDescription(description);
            menuItem.setPrice(price);
            menuItem.setAvailable(available);

            menuItemImpl.updateMenuItem(menuItem);
            System.out.println("Menu Item Updated.");
        } else {
            System.out.println("Menu Item not found.");
        }
    }

    private void deleteMenuItem(Scanner scanner) {
        System.out.print("Enter Menu Item ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        menuItemImpl.deleteMenuItem(id);
        System.out.println("Menu Item Deleted.");
    }

    private void listAllMenuItems() {
        List<MenuItem> menuItems = menuItemImpl.getAllMenuItems();
        if (menuItems.isEmpty()) {
            System.out.println("No Menu Items found.");
        } else {
            for (MenuItem menuItem : menuItems) {
                System.out.println(menuItem);
            }
        }
    }
}
```

service/OrderDetailService.java

```
package com.rms.service;

import com.rms.impl.OrderDetailImpl;
import com.rms.models.OrderDetail;

import java.util.List;
import java.util.Scanner;

public class OrderDetailService {
    private OrderDetailImpl orderDetailImpl = new OrderDetailImpl();

    public void manageOrderDetails(Scanner scanner) {
        while (true) {
            System.out.println("Order Detail Management");
            System.out.println("1. Create Order Detail");
            System.out.println("2. Read Order Detail");
            System.out.println("3. Update Order Detail");
            System.out.println("4. Delete Order Detail");
            System.out.println("5. List All Order Details");
            System.out.println("6. Back");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
        scanner.nextLine(); // consume newline

        switch (choice) {
          case 1:
            createOrderDetail(scanner);
            break;
          case 2:
            readOrderDetail(scanner);
            break;
          case 3:
            updateOrderDetail(scanner);
            break;
          case 4:
            deleteOrderDetail(scanner);
            break;
          case 5:
            listAllOrderDetails();
            break;
          case 6:
            return;
          default:
            System.out.println("Invalid choice. Try again.");
        }
      }
    }

    private void createOrderDetail(Scanner scanner) {
      System.out.print("Enter Order ID: ");
      int orderId = scanner.nextInt();
      System.out.print("Enter Menu Item ID: ");
      int menuItemId = scanner.nextInt();
      System.out.print("Enter Quantity: ");
      int quantity = scanner.nextInt();
      System.out.print("Enter Price: ");
```

```
        double price = scanner.nextDouble();
        scanner.nextLine(); // consume newline

        OrderDetail orderDetail = new OrderDetail(0, orderId, menuItemId, quantity,
price);
        orderDetailImpl.createOrderDetail(orderDetail);
        System.out.println("Order Detail Created.");
    }

    private void readOrderDetail(Scanner scanner) {
        System.out.print("Enter Order Detail ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        OrderDetail orderDetail = orderDetailImpl.readOrderDetail(id);
        if (orderDetail != null) {
            System.out.println(orderDetail);
        } else {
            System.out.println("Order Detail not found.");
        }
    }

    private void updateOrderDetail(Scanner scanner) {
        System.out.print("Enter Order Detail ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline

        OrderDetail orderDetail = orderDetailImpl.readOrderDetail(id);
        if (orderDetail != null) {
            System.out.print("Enter New Order ID: ");
            int orderId = scanner.nextInt();
            System.out.print("Enter New Menu Item ID: ");
            int menuItemId = scanner.nextInt();
            System.out.print("Enter New Quantity: ");
            int quantity = scanner.nextInt();
        System.out.print("Enter New Price: ");
        double price = scanner.nextDouble();
        scanner.nextLine(); // consume newline

        orderDetail.setOrderId(orderId);
        orderDetail.setMenuItemId(menuItemId);
        orderDetail.setQuantity(quantity);
        orderDetail.setPrice(price);

        orderDetailImpl.updateOrderDetail(orderDetail);
        System.out.println("Order Detail Updated.");
    } else {
        System.out.println("Order Detail not found.");
    }
}

private void deleteOrderDetail(Scanner scanner) {
    System.out.print("Enter Order Detail ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // consume newline
    orderDetailImpl.deleteOrderDetail(id);
    System.out.println("Order Detail Deleted.");
}

private void listAllOrderDetails() {
    List<OrderDetail> orderDetails = orderDetailImpl.getAllOrderDetails();
    if (orderDetails.isEmpty()) {
        System.out.println("No Order Details found.");
    } else {
        for (OrderDetail orderDetail : orderDetails) {
            System.out.println(orderDetail);
        }
    }
}
```

```
}
```

service/OrderService.java

```
package com.rms.service;

import com.rms.impl.OrderImpl;
import com.rms.models.Order;

import java.util.Date;
import java.util.List;
import java.util.Scanner;

public class OrderService {
  private OrderImpl orderImpl = new OrderImpl();

  public void manageOrders(Scanner scanner) {
    while (true) {
      System.out.println("Order Management");
      System.out.println("1. Create Order");
      System.out.println("2. Read Order");
      System.out.println("3. Update Order");
      System.out.println("4. Delete Order");
      System.out.println("5. List All Orders");
      System.out.println("6. Back");
      System.out.print("Choose an option: ");

      int choice = scanner.nextInt();
      scanner.nextLine(); // consume newline

      switch (choice) {
        case 1:
          createOrder(scanner);
```

```
                break;
            case 2:
                readOrder(scanner);
                break;
            case 3:
                updateOrder(scanner);
                break;
            case 4:
                deleteOrder(scanner);
                break;
            case 5:
                listAllOrders();
                break;
            case 6:
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}

private void createOrder(Scanner scanner) {
    System.out.print("Enter User ID: ");
    int userId = scanner.nextInt();
    System.out.print("Enter Order Date (YYYY-MM-DD): ");
    String dateString = scanner.next();
    Date orderDate = java.sql.Date.valueOf(dateString);
    System.out.print("Enter Total Amount: ");
    double totalAmount = scanner.nextDouble();
    System.out.print("Enter Table ID: ");
    int tableId = scanner.nextInt();
    scanner.nextLine(); // consume newline

    Order order = new Order(0, userId, orderDate, totalAmount, tableId);
    orderImpl.createOrder(order);
    System.out.println("Order Created.");
}

private void readOrder(Scanner scanner) {
    System.out.print("Enter Order ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // consume newline
    Order order = orderImpl.readOrder(id);
    if (order != null) {
        System.out.println(order);
    } else {
        System.out.println("Order not found.");
    }
}

private void updateOrder(Scanner scanner) {
    System.out.print("Enter Order ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // consume newline

    Order order = orderImpl.readOrder(id);
    if (order != null) {
        System.out.print("Enter New User ID: ");
        int userId = scanner.nextInt();
        System.out.print("Enter New Order Date (YYYY-MM-DD): ");
        String dateString = scanner.next();
        Date orderDate = java.sql.Date.valueOf(dateString);
        System.out.print("Enter New Total Amount: ");
        double totalAmount = scanner.nextDouble();
        System.out.print("Enter New Table ID: ");
        int tableId = scanner.nextInt();
        scanner.nextLine(); // consume newline
```

```
            order.setUserId(userId);
            order.setOrderDate(orderDate);
            order.setTotalAmount(totalAmount);
            order.setTableId(tableId);

            orderImpl.updateOrder(order);
            System.out.println("Order Updated.");
        } else {
            System.out.println("Order not found.");
        }
    }

    private void deleteOrder(Scanner scanner) {
        System.out.print("Enter Order ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        orderImpl.deleteOrder(id);
        System.out.println("Order Deleted.");
    }

    private void listAllOrders() {
        List<Order> orders = orderImpl.getAllOrders();
        if (orders.isEmpty()) {
            System.out.println("No Orders found.");
        } else {
            for (Order order : orders) {
                System.out.println(order);
            }
        }
    }
}
```

service/ReservationService.java

```
package com.rms.service;

import com.rms.impl.ReservationImpl;
import com.rms.models.Reservation;

import java.util.Date;
import java.util.List;
import java.util.Scanner;

public class ReservationService {
  private ReservationImpl reservationImpl = new ReservationImpl();

  public void manageReservations(Scanner scanner) {
    while (true) {
      System.out.println("Reservation Management");
      System.out.println("1. Create Reservation");
      System.out.println("2. Read Reservation");
      System.out.println("3. Update Reservation");
      System.out.println("4. Delete Reservation");
      System.out.println("5. List All Reservations");
      System.out.println("6. Back");
      System.out.print("Choose an option: ");

      int choice = scanner.nextInt();
      scanner.nextLine(); // consume newline

      switch (choice) {
        case 1:
          createReservation(scanner);
          break;
        case 2:
          readReservation(scanner);
          break;
```

```
            case 3:
                updateReservation(scanner);
                break;
            case 4:
                deleteReservation(scanner);
                break;
            case 5:
                listAllReservations();
                break;
            case 6:
                return;
            default:
                System.out.println("Invalid choice. Try again.");
          }
        }
      }

      private void createReservation(Scanner scanner) {
        System.out.print("Enter User ID: ");
        int userId = scanner.nextInt();
        System.out.print("Enter Reservation Date (YYYY-MM-DD): ");
        String dateString = scanner.next();
        Date reservationDate = java.sql.Date.valueOf(dateString);
        System.out.print("Enter Number of People: ");
        int numberOfPeople = scanner.nextInt();
        System.out.print("Enter Table ID: ");
        int tableId = scanner.nextInt();
        scanner.nextLine(); // consume newline

        Reservation reservation = new Reservation(0, userId, reservationDate,
numberOfPeople, tableId);
        reservationImpl.createReservation(reservation);
        System.out.println("Reservation Created.");
      }
```

```
private void readReservation(Scanner scanner) {
  System.out.print("Enter Reservation ID: ");
  int id = scanner.nextInt();
  scanner.nextLine(); // consume newline
  Reservation reservation = reservationImpl.readReservation(id);
  if (reservation != null) {
    System.out.println(reservation);
  } else {
    System.out.println("Reservation not found.");
  }
}

private void updateReservation(Scanner scanner) {
  System.out.print("Enter Reservation ID: ");
  int id = scanner.nextInt();
  scanner.nextLine(); // consume newline

  Reservation reservation = reservationImpl.readReservation(id);
  if (reservation != null) {
    System.out.print("Enter New User ID: ");
    int userId = scanner.nextInt();
    System.out.print("Enter New Reservation Date (YYYY-MM-DD): ");
    String dateString = scanner.next();
    Date reservationDate = java.sql.Date.valueOf(dateString);
    System.out.print("Enter New Number of People: ");
    int numberOfPeople = scanner.nextInt();
    System.out.print("Enter New Table ID: ");
    int tableId = scanner.nextInt();
    scanner.nextLine(); // consume newline

    reservation.setUserId(userId);
    reservation.setReservationDate(reservationDate);
    reservation.setNumberOfPeople(numberOfPeople);
```

```
            reservation.setTableId(tableId);

            reservationImpl.updateReservation(reservation);
            System.out.println("Reservation Updated.");
        } else {
            System.out.println("Reservation not found.");
        }
    }

    private void deleteReservation(Scanner scanner) {
        System.out.print("Enter Reservation ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        reservationImpl.deleteReservation(id);
        System.out.println("Reservation Deleted.");
    }

    private void listAllReservations() {
        List<Reservation> reservations = reservationImpl.getAllReservations();
        if (reservations.isEmpty()) {
            System.out.println("No Reservations found.");
        } else {
            for (Reservation reservation : reservations) {
                System.out.println(reservation);
            }
        }
    }
}
```

service/RestaurantTableService.java

```
package com.rms.service;
```

```
import com.rms.impl.RestaurantTableImpl;
import com.rms.models.RestaurantTable;

import java.util.List;
import java.util.Scanner;

public class RestaurantTableService {
  private RestaurantTableImpl restaurantTableImpl = new RestaurantTableImpl();

  public void manageRestaurantTables(Scanner scanner) {
    while (true) {
      System.out.println("Restaurant Table Management");
      System.out.println("1. Create Restaurant Table");
      System.out.println("2. Read Restaurant Table");
      System.out.println("3. Update Restaurant Table");
      System.out.println("4. Delete Restaurant Table");
      System.out.println("5. List All Restaurant Tables");
      System.out.println("6. Back");
      System.out.print("Choose an option: ");

      int choice = scanner.nextInt();
      scanner.nextLine(); // consume newline

      switch (choice) {
        case 1:
          createRestaurantTable(scanner);
          break;
        case 2:
          readRestaurantTable(scanner);
          break;
        case 3:
          updateRestaurantTable(scanner);
          break;
        case 4:
                deleteRestaurantTable(scanner);
                break;
            case 5:
                listAllRestaurantTables();
                break;
            case 6:
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}

private void createRestaurantTable(Scanner scanner) {
    System.out.print("Enter Table Number: ");
    int tableNumber = scanner.nextInt();
    System.out.print("Enter Capacity: ");
    int capacity = scanner.nextInt();
    scanner.nextLine(); // consume newline
    System.out.print("Enter Location Description: ");
    String locationDescription = scanner.nextLine();
    System.out.print("Enter Status (Available/Occupied): ");
    String status = scanner.nextLine();

    RestaurantTable table = new RestaurantTable(0, tableNumber, capacity,
locationDescription, status);
    restaurantTableImpl.createRestaurantTable(table);
    System.out.println("Restaurant Table Created.");
}

private void readRestaurantTable(Scanner scanner) {
    System.out.print("Enter Table ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // consume newline
```

```
        RestaurantTable table = restaurantTableImpl.readRestaurantTable(id);
        if (table != null) {
            System.out.println(table);
        } else {
            System.out.println("Restaurant Table not found.");
        }
    }

    private void updateRestaurantTable(Scanner scanner) {
        System.out.print("Enter Table ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline

        RestaurantTable table = restaurantTableImpl.readRestaurantTable(id);
        if (table != null) {
            System.out.print("Enter New Table Number: ");
            int tableNumber = scanner.nextInt();
            System.out.print("Enter New Capacity: ");
            int capacity = scanner.nextInt();
            scanner.nextLine(); // consume newline
            System.out.print("Enter New Location Description: ");
            String locationDescription = scanner.nextLine();
            System.out.print("Enter New Status (Available/Occupied): ");
            String status = scanner.nextLine();

            table.setTableNumber(tableNumber);
            table.setCapacity(capacity);
            table.setLocationDescription(locationDescription);
            table.setStatus(status);

            restaurantTableImpl.updateRestaurantTable(table);
            System.out.println("Restaurant Table Updated.");
        } else {
            System.out.println("Restaurant Table not found.");
        }
    }

    private void deleteRestaurantTable(Scanner scanner) {
        System.out.print("Enter Table ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // consume newline
        restaurantTableImpl.deleteRestaurantTable(id);
        System.out.println("Restaurant Table Deleted.");
    }

    private void listAllRestaurantTables() {
        List<RestaurantTable> tables = restaurantTableImpl.getAllRestaurantTables();
        if (tables.isEmpty()) {
            System.out.println("No Restaurant Tables found.");
        } else {
            for (RestaurantTable table : tables) {
                System.out.println(table);
            }
        }
    }
}
```

service/UserService.java

```
package com.rms.service;

import com.rms.impl.UserImpl;
import com.rms.models.User;

import java.util.List;
import java.util.Scanner;
```

```
public class UserService {
  private UserImpl userImpl = new UserImpl();

  public void manageUsers(Scanner scanner) {
    while (true) {
      System.out.println("User Management");
      System.out.println("1. Create User");
      System.out.println("2. Read User");
      System.out.println("3. Update User");
      System.out.println("4. Delete User");
      System.out.println("5. List All Users");
      System.out.println("6. Back");
      System.out.print("Choose an option: ");

      int choice = scanner.nextInt();
      scanner.nextLine(); // consume newline

      switch (choice) {
        case 1:
          createUser(scanner);
          break;
        case 2:
          readUser(scanner);
          break;
        case 3:
          updateUser(scanner);
          break;
        case 4:
          deleteUser(scanner);
          break;
        case 5:
          listAllUsers();
          break;
        case 6:
          return;
        default:
          System.out.println("Invalid choice. Try again.");
      }
    }
  }

  private void createUser(Scanner scanner) {
    System.out.print("Enter User Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Email: ");
    String email = scanner.nextLine();
    System.out.print("Enter Password: ");
    String password = scanner.nextLine();
    System.out.print("Enter Role: ");
    String role = scanner.nextLine();

    User user = new User(0, name, email, password, role);
    userImpl.createUser(user);
    System.out.println("User Created.");
  }

  private void readUser(Scanner scanner) {
    System.out.print("Enter User ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // consume newline
    User user = userImpl.readUser(id);
    if (user != null) {
      System.out.println(user);
    } else {
      System.out.println("User not found.");
    }
  }
```

```
private void updateUser(Scanner scanner) {
  System.out.print("Enter User ID: ");
  int id = scanner.nextInt();
  scanner.nextLine(); // consume newline

  User user = userImpl.readUser(id);
  if (user != null) {
    System.out.print("Enter New Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter New Email: ");
    String email = scanner.nextLine();
    System.out.print("Enter New Password: ");
    String password = scanner.nextLine();
    System.out.print("Enter New Role: ");
    String role = scanner.nextLine();

    user.setName(name);
    user.setEmail(email);
    user.setPassword(password);
    user.setRole(role);

    userImpl.updateUser(user);
    System.out.println("User Updated.");
  } else {
    System.out.println("User not found.");
  }
}

private void deleteUser(Scanner scanner) {
  System.out.print("Enter User ID: ");
  int id = scanner.nextInt();
  scanner.nextLine(); // consume newline
  userImpl.deleteUser(id);
  System.out.println("User Deleted.");
```

```
    }

    private void listAllUsers() {
        List<User> users = userImpl.getAllUsers();
        if (users.isEmpty()) {
            System.out.println("No Users found.");
        } else {
            for (User user : users) {
                System.out.println(user);
            }
        }
    }
}
```