Software Requirements and Design Document

for

Restaurant Reservation and Payment Management System

Prepared by Saad Nadeem,Amaar khan,Mustafa iqbal

Intelliware

25 November 2024

Table of Contents

Table of Contents ii

1. Introduction 1

1.1 Purpose 1

1.2 Product Scope 1

1.3 Title 1

1.4 Objectives 1

1.5 Problem Statement 1

2. Overall Description 1

2.1 Product Perspective 1

2.2 Product Functions 2

2.3 List of Use Cases 2

2.4 Extended Use Cases 2

2.5 Use Case Diagram 2

3. Other Nonfunctional Requirements 2

3.1 Performance Requirements 2

3.2 Safety Requirements 2

3.3 Security Requirements 2

3.4 Software Quality Attributes 2

3.5 Business Rules 3

3.6 Operating Environment 3

3.7 User Interfaces 3

4. Domain Model 3

5. System Sequence Diagram 3

6. Sequence Diagram 3

7. Class Diagram 4

8. Package Diagram 4

9. Deployment Diagram 4

# Introduction

## Purpose

The product described in this document is a **Restaurant Reservation and Payment Management System**, which is designed to automate and streamline the reservation and payment processes in a restaurant environment. The version or release number has not been explicitly stated, but it is understood that this document pertains to the first iteration or release of the system.

The SRS document specifically covers the **reservation and payment subsystem** of the restaurant system. It does not describe the entire restaurant management system (e.g., it may not include inventory management, order tracking, or other subsystems). The focus is on automating the manual tasks involved in handling reservations and payments.

This document specifies both the user-facing and administrative components for managing reservations, payment statuses, and related interactions between the restaurant and its customers

## Product Scope

The project focuses on developing a reservation and payment management system for a

restaurant. This system allows customers to reserve tables, manage their bookings, and

process payments. The project is specifically designed for restaurant operations, automating

manual reservation management and payment handling.

## Title

Restaurant Reservation and Payment Management System

## Objectives

The main objective of the system is to streamline restaurant operations by providing the

following features:

1. Reservation Management: Allow customers to reserve tables online, view their

reservations, and manage bookings.

2. Payment Processing: Integrate payment methods to allow users to view and update

payment statuses for their orders.

3. Admin Interface: Provide restaurant staff with administrative tools to manage

customer reservations, orders, and payments efficiently.

4. Customer Interface: Offer a user-friendly interface for customers to interact with the

system, book tables, and manage their payments.

## Problem Statement

In traditional restaurant environments, handling reservations and payments manually is time-

consuming and prone to errors, especially during peak hours. This project addresses the

need for automation in restaurant operations, reducing the workload for staff and improving

the customer experience.

The system mitigates manual processes by allowing customers to make online reservations,

receive automated confirmations and make payments via a streamlined interface. This

project will result in fewer errors in managing bookings and a more efficient payment

process, contributing to overall operational efficiency and customer satisfaction.

# Overall Description

## Product Perspective

The **Restaurant Reservation and Payment Management System** is a self-contained product designed to manage restaurant reservations and payment processing. This software solution is intended to replace traditional, manual methods of managing reservations, which are often error-prone and time-consuming. The system offers a modern approach to customer interactions, automating the reservation process, facilitating payments, and providing an efficient interface for both customers and restaurant staff.

This system is not a follow-up to any previous product but is instead a standalone solution tailored specifically for the restaurant industry. It is not intended as a component of a larger ecosystem, although it can potentially interface with other restaurant management systems in the future, such as POS (Point of Sale) or kitchen management software.

## Product Functions

1. **Reservation Management:**
   * **Make a Reservation**: Users can reserve tables online by providing necessary details such as name, contact, date, and time.
   * **View Reservations**: Users can view their existing reservations, including reservation details.
   * **Cancel Reservations**: Customers can cancel their reservations if needed.
2. **Payment Processing:**
   * **View Payment Status**: Customers can view the current status of their payment for orders associated with their reservation.
   * **Make Payment**: Users can process payments for their reservations and orders directly through the system.
   * **Update Payment Status**: The system updates the payment status to "paid" once a successful payment is made.
3. **Admin Functions:**
   * **Manage Reservations**: Admins can view customer reservations.
   * **Process Payments:** Admins can verify and update cash payment status for orders associated with reservations.
   * **View Reservation Reports:** Admins can access reports such as feedback, reservations over past months, and payment metrics.
4. **Customer Interaction:**
   * **Reservation Confirmation**: After making a reservation, customers receive a confirmation message or popup.
5. **System Alerts:**
   * **Error Handling**: The system provides error messages and alerts in case of issues such as payment failures or invalid reservation data.

These functions collectively streamline the restaurant's reservation and payment management process, improving efficiency and enhancing the user experience.

* **Data Flow Overview:**

The major components and their interactions can be depicted as follows:

* **Customer Interface**: Manages views and assigns tasks to Customer Service Handler.
* **Customer Service Handler:** Manages business logic for customer interactions.
* **Admin Interface**: Manages Admin related views and assigns tasks to admin service handler.
* **Admin Service Handler**: Manages business logic for administrative functionalities.

This system ensures smooth coordination between customers, admin staff, and the backend services.

## List of Use Cases

1. Make Reservation
2. View Reservation
3. Cancel Reservation
4. Make Payment
5. View All Reservations
6. Generate Reservation Report
7. Generate Feedback Report (new)
8. Generate Revenue Report (new)
9. Update Reservation
10. Send Reservation Confirmation
11. Give Feedback
12. Checkout
13. Verify Cash Payment
14. Add Table
15. Delete Table

## Extended Use Cases

**Use Case 1: Make Reservation (Amaar)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Wants to reserve a table at the restaurant.
  + *Restaurant*: Needs accurate records of reservations to avoid overbooking.
* **Preconditions**: Customer has access to the system.
* **Postconditions**: Reservation is successfully made and recorded in the system.
* **Main Success Scenario**:

Customer accesses the reservation interface.

Customer selects date, time, and number of people.

System checks availability and confirms reservation.

System stores reservation details.

* **Extensions**:
  + *If no tables are available*: The system displays an error message and suggests alternative times.

**Use Case 2: View Reservation (Saad)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Wants to confirm their reservation details.
  + *Restaurant*: Ensures customers have access to accurate reservation information.
* **Preconditions**: Customer has a reservation in the system.
* **Postconditions**: Reservation details are displayed.
* **Main Success Scenario**:

Customer selects "View Reservation" option.

System displays reservation details.

* **Extensions**:
  + *If reservation is not found*: System displays an error message.

**Use Case 3: Cancel Reservation (Mustafa)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Needs to cancel a reservation due to change in plans.
  + *Restaurant*: Avoids no-shows by offering an easy cancellation option.
* **Preconditions**: Customer has a valid reservation.
* **Postconditions**: Reservation is canceled, and the slot becomes available.
* **Main Success Scenario**:

Customer chooses "Cancel Reservation."

System cancels reservation and updates availability.

System verifies reservation.

* **Extensions**:
  + *If cancellation fails*: System displays an error message.

**Use Case 4: Make Payment (Amaar)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Needs to make a payment for their reservation.
  + *Restaurant*: Requires payment for services rendered.
* **Preconditions**: Customer has an existing order linked to the reservation.
* **Postconditions**: Payment is processed and status is updated.
* **Main Success Scenario**:

Customer initiates payment.

Customer Enters payment details.

System prompts for payment details.

System processes payment and updates status.

* **Extensions**:
  + *If payment fails*: System prompts customer to retry.

**Use Case 5: Update Payment Status (Saad)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: Sub-function
* **Primary Actor**: Admin
* **Stakeholders and Interests**:
  + *Admin*: Needs to track and manage payments.
  + *Restaurant*: Ensures accurate financial records.
* **Preconditions**: Payment has been initiated.
* **Postconditions**: Payment status is updated in the system.
* **Main Success Scenario**:

Admin accesses payment details.

Admin updates payment status as “paid.”

System saves updated status.

* **Extensions**:
  + *If update fails*: System logs the error.

**Use Case 6: Check Table Availability(Mustafa)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: Sub-function
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Needs to know if tables are available.
  + *Restaurant*: Aims to prevent overbooking.
* **Preconditions**: Customer specifies date and time.
* **Postconditions**: Availability information is displayed.
* **Main Success Scenario**:

Customer enters date and time.

System checks availability and displays results.

* **Extensions**:
  + *If no tables are available*: System suggests alternative times.

**Use Case 7: View All Reservations (Amaar)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Admin
* **Stakeholders and Interests**:
  + *Admin*: Needs to view all reservations for management purposes.
  + *Restaurant*: Requires an overview of reservations for planning.
* **Preconditions**: Admin is logged into the system.
* **Postconditions**: All reservations are displayed.
* **Main Success Scenario**:

Admin selects "View All Reservations."

System retrieves and displays reservations

.

* **Extensions**:
  + *If no reservations are found*: System displays a message.

**Use Case 8: Generate Reservation Report(Saad)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Admin
* **Stakeholders and Interests**:
  + *Admin*: Needs reservation data for reporting.
  + *Restaurant*: Uses reports for decision-making and resource allocation.
* **Preconditions**: Admin has access to the system.
* **Postconditions**: Report is generated and accessible.
* **Main Success Scenario**:

Admin selects "Generate Report."

System compiles data and generates report.

* **Extensions**:
  + *If report generation fails*: System logs error.

**Use Case 9: Update Reservation(Mustafa)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Needs to modify their reservation details.
  + *Restaurant*: Requires accurate and up-to-date reservation information.
* **Preconditions**: Customer has an existing reservation.
* **Postconditions**: Reservation details are updated.
* **Main Success Scenario**:

Customer selects "Update Reservation."

Customer modifies details and submits.

System displays current details and prompts for changes.

System saves updated information.

* **Extensions**:
  + *If update fails*: System displays an error message.

**Use Case 10: Send Reservation Confirmation(Mustafa)**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: Sub-function
* **Primary Actor**: System (automated process)
* **Stakeholders and Interests**:
  + *Customer*: Wants confirmation of their reservation.
  + *Restaurant*: Ensures customer receives a record of their reservation.
* **Preconditions**: Reservation is successfully created.
* **Postconditions**: Confirmation is sent to the customer.
* **Main Success Scenario**:

System generates confirmation message.

System sends confirmation to the customer’s email.

Customer receives the confirmation Email

* **Extensions**:
  + *If email sending fails*: System logs the error.

**Use Case 11: Give Feedback**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User Goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
  + *Customer*: Wants to give a review of their experience
  + *Restaurant*: Wants to receive customer feedback to improve their services
* **Preconditions**: Customer has completed their dining experience and has access to the feedback system.
* **Postconditions**: Customer feedback is submitted and recorded in the system.
* **Main Success Scenario**:

Customer selects the "Give Feedback" option.

Customer enters comments and suggestions.

System prompts the customer to rate their experience (e.g., 1 to 5 stars).

System validates the input.

System saves the feedback and confirms submission to the customer.

* **Extensions**:
* if input is invalid: System prompts the customer to correct the input.
* If feedback submission fails: System displays an error message.

**Use Case 12: Checkout**

* **Scope**: Restaurant Reservation and Payment Management System
* **Level**: User Goal
* **Primary Actor**: Customer
* **Stakeholders and Interests**:
* Customer: Wants to complete the payment process for their order.
* Restaurant: Needs to process payments accurately and efficiently.
* **Preconditions**: Customer has finished their meal and is ready to pay.
* **Postconditions**: Payment is successfully processed, and the order is marked as complete.
* **Main Success Scenario**:

The customer selects the "Checkout" option.

The customer chooses a payment method (e.g., credit card, cash).

the customer enters payment details.(In case of card)

The Customer Makes the payment

The system displays the order summary and the total amount due.

The System Asks Payment Method

The system asks for payment details(in case of card)

The system processes the payment.

System confirms payment success and generates a receipt.

* **Extensions**:
* If payment fails: System displays an error message and prompts the customer to retry.
* If payment is successful: System sends a confirmation email with the receipt.

**Use Case 13: Verify Cash Payment**

**• Scope: Restaurant Reservation and Payment Management System**

**• Level: User Goal**

**• Primary Actor: Customer**

**• Stakeholders and Interests:**

* + Admin: Needs to ensure that cash payments are accurately recorded and verified.
  + Customer: Expects a smooth transaction process and confirmation of cash payment.
  + Restaurant: Aims to maintain financial integrity and accurate cash handling records.

**• Preconditions**: Cash payment has been initiated, and the admin is logged into the system.

**• Postconditions**: Cash payment verification status is updated in the system**.**

**• Main Success Scenario:**

Admin selects "Verify Cash Payment" from the

dashboard.

System prompts for payment receipt number or transaction details.

Admin enters the required information and submits.

System checks the cash payment details against the records.

System updates payment status to "Verified" or "Failed" based on the verification outcome.

Admin is notified of the verification outcome.

**• Extensions:**

If payment record is not found: System displays an error message indicating no matching cash payment record.

If verification fails: System prompts the admin to retry or check the cash payment details.

**Use Case 14: Add Table**

* **Scope:** Restaurant Reservation and Payment Management System
* **Level:** User Goal
* **Primary Actor:** Admin
* **Stakeholders and Interests:**
  + **Admin:** Needs to add a new table to accommodate more customers.
  + **Restaurant:** Requires an accurate record of tables to manage reservations effectively.
* **Preconditions:** Admin is logged into the system with permission to modify table data.
* **Postconditions:** New table is successfully added to the system and available for reservations.
* **Main Success Scenario:**
  + Admin selects the "Add Table" option from the system dashboard.
  + System prompts the admin to enter table details (e.g., table number, seating capacity).
  + Admin enters the required information and submits.
  + System saves the new table information and confirms the addition.
* **Extensions:**
  + **If table addition fails:** System logs the error and displays an error message.
  + **If required details are missing:** System prompts the admin to complete the information.

**Use Case 15: Delete Table**

* **Scope:** Restaurant Reservation and Payment Management System
* **Level:** User Goal
* **Primary Actor:** Admin
* **Stakeholders and Interests:**
  + **Admin:** Needs to remove a table no longer in service or to reallocate seating.
  + **Restaurant:** Requires accurate records of tables to avoid reservation errors.
* **Preconditions:** Admin is logged into the system with permission to modify table data.
* **Postconditions:** Table is successfully deleted from the system and removed from availability.
* **Main Success Scenario:**
  + Admin selects the "Delete Table" option from the system dashboard.
  + System displays a list of tables and prompts the admin to choose a table to delete.
  + Admin selects the table to delete and confirms the action.
  + System removes the table from records and updates availability.
* **Extensions:**
  + **If table deletion fails:** System logs the error and displays an error message.
  + **If table is currently reserved:** System displays a message preventing deletion and suggests rescheduling or canceling associated reservations.

## Use Case Diagram

# 

# 3.Other Nonfunctional Requirements

## 3.1 Performance Requirements

The system must meet the following performance requirements to ensure smooth operation under various circumstances, maintaining a high level of efficiency and responsiveness:

1. **Reservation Response Time**:
   * **Requirement**: The system must process a reservation request and provide a confirmation response within **2 seconds**.
   * **Rationale**: This ensures a quick and efficient user experience, allowing customers to confirm their reservations without delays. If response times exceed 2 seconds, users may perceive the system as slow, leading to dissatisfaction and potentially lost reservations.
2. **Payment Processing Time**:
   * **Requirement**: The system must process payment and update the payment status in the database within **5 seconds**.
   * **Rationale**: Payment processing speed is crucial for both customer satisfaction and operational efficiency. Delays in payment processing could cause queues or interruptions in service, negatively impacting the customer experience.
3. **Database Query Performance**:
   * **Requirement**: SQL queries to retrieve reservation details or update payment statuses must return results within **1 second** for typical requests involving a single reservation or payment.
   * **Rationale**: Fast data retrieval and updates are essential for keeping the system responsive and ensuring that users don’t experience significant delays while checking reservation or payment statuses.
4. **Scalability**:
   * **Requirement**: The system should be able to handle up to **500 concurrent users** (simultaneous reservations or payments) without a significant degradation in performance.
   * **Rationale**: As the restaurant expands, the system should be able to accommodate an increasing number of users, ensuring reliability and maintaining good performance even under higher load.
5. **Availability**:
   * **Requirement**: The system should achieve **99.9% uptime** to ensure availability for customers making reservations and processing payments.
   * **Rationale**: High availability is critical for the system’s effectiveness, especially in a customer-facing application like this one, where downtime could directly lead to lost business opportunities.
6. **System Load Handling**:
   * **Requirement**: During peak reservation hours (e.g., weekends, holidays), the system should maintain performance within acceptable limits, even if the number of concurrent users exceeds normal usage by **up to 50%**.
   * **Rationale**: The system should be resilient to traffic spikes and be able to scale resources dynamically or handle peak load without crashing or slowing down significantly.

These performance requirements will guide the development team in making design choices that prioritize user experience, reliability, and the overall efficiency of the system. Ensuring that these performance targets are met will allow the restaurant to operate effectively and provide a seamless experience for customers and staff alike.

## 3.2 Safety Requirements

To ensure the safe operation of the Reservation and Payment Management System, the following safety measures must be implemented:

1. **Data Protection**: The system must protect sensitive customer and payment data by complying with relevant data protection laws (e.g., GDPR, PCI DSS). Encryption should be used to safeguard sensitive information.
2. **Error Handling**: The system must be able to handle errors gracefully to avoid data corruption or loss. Proper validation should be performed on user inputs to prevent invalid data from being processed.
3. **Payment Security**: All payment transactions must be securely processed through trusted payment gateways. Prevent unauthorized access to payment systems and protect user credentials.
4. **User Authentication**: Ensure that only authorized staff can access administrative functions through secure login mechanisms.
5. **Compliance**: Adhere to any applicable local and international safety regulations for software used in public services, including certifications for handling payment transactions.

Safety certifications like **PCI DSS** for payment processing and data encryption standards must be met to ensure compliance with safety protocols.

## 3.3 Security Requirements

## The system must ensure the privacy and security of customer data, including reservation details and payment information. All sensitive data, such as payment details, should be encrypted using industry-standard encryption protocols (e.g., TLS for data transmission). User authentication will be required for access to the system, with secure login procedures (username/password and possibly multi-factor authentication). The system should also comply with relevant data protection regulations (e.g., GDPR or PCI-DSS for payment data security). Regular security audits should be conducted to ensure compliance and prevent data breaches.

## 3.4 Software Quality Attributes

 **Usability**: The system will be designed with a simple and intuitive user interface, ensuring that customers can easily navigate through the reservation and payment processes. Usability testing will be conducted to ensure ease of use for both customers and administrative staff, with a target task completion time of under 5 minutes for common operations (e.g., making a reservation).

 **Reliability**: The system must have an uptime of at least 99.5%. Automated error handling and logging mechanisms will be implemented to ensure the system remains operational under varying loads, especially during peak times.

 **Maintainability**: The system's codebase will be modular and well-documented to ensure easy updates and feature expansions. Routine maintenance tasks should not require more than 4 hours per month.

 **Flexibility**: The system will be built with extensibility in mind, allowing for future integration with third-party payment gateways or reservation platforms. Modifications to functionality (e.g., adding a new payment method) should not require major system overhauls.

 **Testability**: The system will include unit tests for critical components, such as payment processing and reservation management. Test coverage will aim for at least 85%, with automated testing in place for frequent tasks.

 **Interoperability**: The system will support integration with existing restaurant tools such as customer databases and POS systems. Compatibility will be ensured with standard web browsers (Chrome, Firefox, Safari) and mobile devices (iOS, Android).

 **Robustness**: The system should handle unexpected errors gracefully, providing informative error messages without crashing. The system should be capable of recovering from server failures with minimal downtime.

 **Portability**: The system will be developed with platform independence in mind, ensuring that it can run on any standard web server or cloud platform without the need for major modifications0

## 3.6 Business Rules

1. **Customer Role:**
   * Customers can reserve tables, view their reservation details, and make payments for their orders.
   * A customer can only modify or cancel a reservation.
   * Customers can only access their own reservation and payment information.
2. **Admin Role:**
   * Administrators can view and manage all reservations.
   * Admins can view and modify payment statuses, including marking payments as "paid" once completed.
   * Administrators view several types of reports.
3. **Reservation Limitation:**
   * A reservation can only be made if there are available tables for the selected time.
   * The system will prevent double-booking of tables.
4. **Payment Authorization:**
   * Payment status can only be updated once the transaction is successfully processed.
   * Payment details can be accessed and updated only by the customer or authorized admin staff.

These rules ensure that both customer and admin actions are performed within the defined system boundaries, preventing errors and maintaining data integrity.

## 3.7 Operating Environment

The software will operate in a typical restaurant setting, utilizing standard desktop or laptop hardware. The required specifications include a modern computer with at least 1GB of RAM, a 1.8 GHz processor, and 10GB of free disk space

The system is compatible with Windows 10/11, macOS 10.15 or higher, and Linux (Ubuntu 20.04 or newer). Java 8 or higher is required for the application, along with a JDBC-compatible database (Microsoft SQL SERVER) for managing reservation and payment data.

For the front-end, the software integrates with JavaFX for the user interface. It can operate and connect with other business applications as needed.

## User Interfaces

The software product will include two main user interfaces: the **Admin Interface** and the **Customer Interface**. Both will follow a simple, intuitive design with consistent style elements to ensure ease of use.

# Admin Interface:

* **Purpose**: Enables restaurant staff to view reservations, verify payments.
* **Screen Layout**: Includes a tables displaying reservations, charts displaying reports, screens displaying data etc.
* **Buttons:** “search” buttons to search for reservations or customers “delete” “Update” buttons to delete or update tables etc.

1. **Customer Interface:**

* **Purpose:** Allows customers to reserve tables and make payments.
* **Screen Layout:** Tab Pane including reservations and checkout with each of them having sub tabs for specific functionalities.
* **Buttons**: "View Reservations ", "Cancel Reservation", "Make Reservations", “Pay by Cash” “Pay by Card” etc.
* **Functions:** View reservation details, book a table, cancel reservation, and process payment securely.
* **Standard Features:** Input validation (e.g., ensure email format is correct), confirmation pop-ups upon successful booking, and error messages if a reservation cannot be processed.

Both interfaces will follow a clean, modern design, with readable fonts, intuitive navigation, and a responsive layout for different screen sizes.

# Domain Model

# 

# System Sequence Diagram

# 

# 

# 

# 

# 

# 

# Sequence Diagram

# 

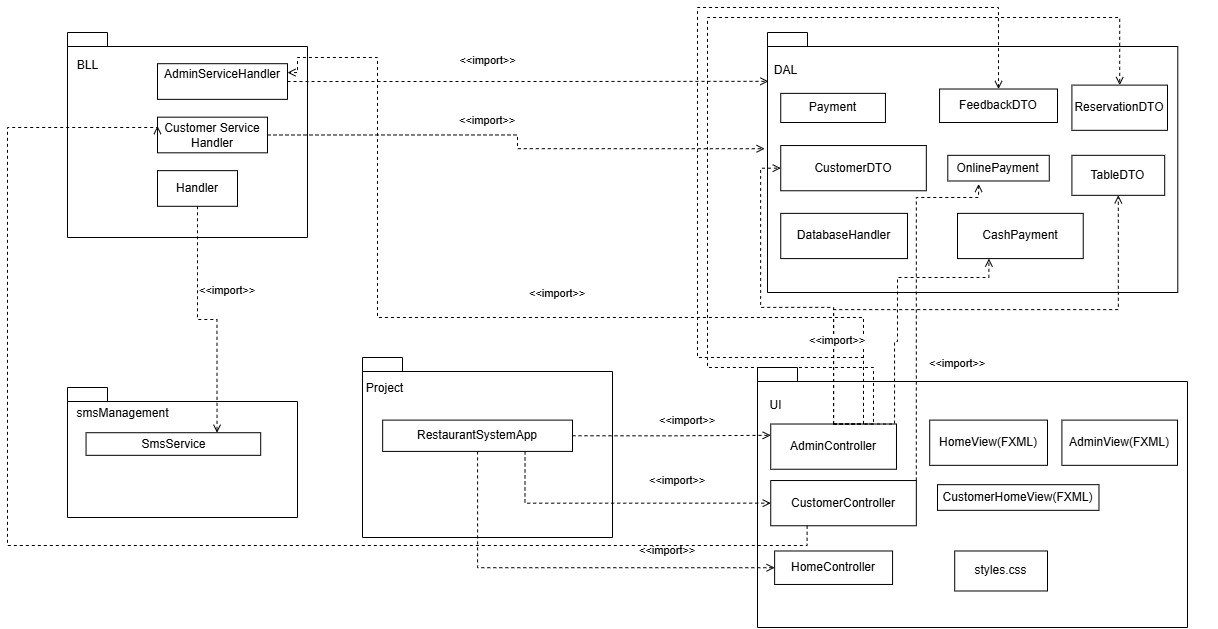
# Class Diagram

# A diagram of a computer program Description automatically generated with medium confidence

7 Component Diagram

# A diagram of a company Description automatically generated

# Package Diagram



# Deployment Diagram

