Software Requirements Specification

for

Food ordering software

Version 1.0

Prepared by: Vinayak Kumar Singh

Registration No: 23MCA1030

Submitted To: Dr. Prakash B

Date: 08-10-2023

Table of Contents

1. Introduction

Food ordering software has transformed dining, turning it into a convenient digital experience. It allows customers to browse menus, order, and track deliveries with ease. With the global online food ordering market expected to reach US$1.65 trillion by 2027, the significance of this software cannot be overstated.

Importance of Food ordering software:

Food ordering software plays a significant role in today's world. It provides a number of benefits to both customers and restaurants, including:

Convenience: Customers can place food orders at any time and om any location. Families and busy folks who don't have time to prepare or eat out would especially benefit from this.

Efficiency: Food ordering software automates the ordering and delivery process which makes it quicker and more effective for both customers and restaurants.

Accuracy: Since clients can explicitly state their preferences and dietary restrictions online ordering systems help to reduce errors in orders.

Choice: Customers have access to a greater choice of meal alternatives by using food ordering software, including eatery that might not be in their immediate area.

Data insights: Food ordering software can give restaurants useful data insights into the preferences and ordering patterns of their consumers. The menu, prices, and general client experience can all be enhanced with the help of this data.

Advantages of food ordering software:

Software for ordering food has a number of benefits over conventional ordering processes, such as:

Easy customization: Customers can add, remove, or indicate dietary restrictions from toppings, sides, or other components of their orders to make them more to their liking.

Secure payment methods: To safeguard clients' financial information, food ordering software frequently makes use of secure payment options including credit cards and online wallets.

Improved operational efficiency: By automating processes like order processing, kitchen management, and delivery tracking, food ordering software can assist businesses in enhancing their operational effectiveness.

Increased sales: By making it simpler for customers to order and pay for their food restaurants can help increase their sales and raise their revenues.

1.1 Purpose

This SRS document's main goal is to provide a thorough explanation of the requirements, guidelines, and limitations for creating and implementing the Food ordering software (FOS). This document is meant for project managers, stakeholders, developers, as well as end users including clients and employees (waitresses, cooks, etc.).

1.2 Document Conventions

Diagrams: Used for presenting complex information in a user-friendly manner.

Headings and Subheadings: Employed to organize and differentiate sections.

Acronyms: The terms "FOS," "UI," "RDBMS," and "CRM," are frequently used in this work. These abbreviations stand for “Food ordering software”," "User Interface," "Database Management System," and "Customer Relationship Management”.

1.3 Intended Audience and Reading Suggestions

Developers: Technical specifications and architecture

Project Managers: Project scope and timelines

Testers: Functional requirements and testing procedures

Users: User manuals and guides

Documentation Writers: Documentation standards and guidelines

1.4 Product Scope

A web-based tool called FOS aims to increase restaurant sales, save staff members time, and improve order accuracy. Its goal is to make ordering easier by providing a user-friendly interface, effective staff, and effective customer communication.

This document provides a comprehensive list of FOS's specifications, constraints, and development guidelines. It is a resource for developers, project managers, and end users.

1.5 References

IEEE SRS template

2. Overall Description

2.1 Product Perspective

The Food Ordering System (FOS) is a standalone web application, not part of a larger system. It operates independently but interfaces with external systems to ensure seamless functionality:

Payment Gateway: Facilitates secure payment processing by interacting with banking accounts.

Inventory Management System: Checks real-time item availability and updates inventory post-orders.

CRM (Customer Relationship Management): Stores and manages customer data, order history, and feedback.

Social Media Platforms: Enables customers to share their feedback and experiences on social media.

Marketing Tools: Sends promotional materials to customers based on order history and preferences.

FOS uses a web-based GUI that can be accessed from any device with an internet connection and a web browser that is compatible. Customers get access to account login, menus, payment methods, and feedback systems using this GUI. Additionally, it interfaces with the payment gateway to handle payments and confirm orders. It provides administration capabilities to administrators so they can keep track of orders, manage inventories, and check customer feedback. This enables order status updates and feedback viewing.

The system's hardware interfaces include web and database servers for hosting and data storage, without requiring direct connections to peripheral devices. The necessary software interfaces encompass web servers, web frameworks, and database management systems (DBMS). The DBMS manages data storage and retrieval while the HTTP and HTTPS protocols are used for communication between the client and server.

FOS operates as a web-based program and doesn't have any unique memory or processing needs, making it compatible with current web browsers. The technology is quite flexible and can be used in a variety of restaurant locations and settings. It allows the customization of the menu items, costs, modes of payment, languages, money, and time zones. This versatility makes sure that FOS can give a customized experience to both consumers and restaurant personnel while smoothly integrating with different restaurant surroundings.

2.2 Product Functions

Customer Functions

- *Browse restaurant menus*
- *Search for restaurants by location, cuisine, or other criteria*
- *Place orders for food and drinks*
- *Customize orders by adding or removing toppings, choosing different sides, or specifying dietary restrictions*
- *Pay for orders securely using a credit card or other online payment method*
- *Track the status of orders in real-time*
- *Rate and review restaurants*

Restaurant Functions

- *Manage online orders*
- *Receive notifications for new orders*
- *Accept or reject orders*
- *Prepare and deliver orders to customers*
- *Generate reports on sales, orders, and other data*

Admin Functions

- *Add, edit, and remove restaurants*
- *Add, edit, and remove menu items*
- *Manage user accounts*
- *Generate reports on system usage*

2.3 User Classes and Characteristics

Customers:

- *Frequency of Use: Varies, from occasional users to frequent orders.*
- *Functions Used: Browsing menus, placing orders, customizing orders, tracking orders, providing feedback.*

- *Technical Expertise: Varied, from basic internet users to tech-savvy individuals.*
- *Security or Privilege Levels: Standard user privileges for ordering and providing feedback.*
- *Educational Level: Diverse educational backgrounds.*
- *Experience: Varies, from new users to returning customers.*

Restaurant Owners and Staff:

- *Frequency of Use: Regular daily usage for order management.*
- *Subset of Product Functions Used: Managing orders, inventory control, communication with customers.*
- *Technical Expertise: Basic computer skills.*
- *Security or Privilege Levels: Admin privileges for order management and inventory control.*
- *Educational Level: Varied educational backgrounds.*
- *Experience: Experience in restaurant operations and order management.*

System Administrators:

- *Frequency of Use: Occasional maintenance and system updates.*
- *Subset of Product Functions Used: System maintenance, security management.*
- *Technical Expertise: Advanced technical skills for system administration.*
- *Security or Privilege Levels: Highest level of access for system maintenance and security.*
- *Educational Level: Typically holds a technical degree or certification.*
- *Experience: Experienced in system administration and maintenance.*

Managers and Business Owners:

- *Frequency of Use: Periodic access for business analysis.*
- *Subset of Product Functions Used: Access to business analytics and reports.*
- *Technical Expertise: Basic understanding of FOS data and reports.*
- *Security or Privilege Levels: Moderate access for business analytics.*

- *Educational Level: Varied educational backgrounds.*
- *Experience: Extensive experience in restaurant management and business analysis.*

Delivery Personnel:

- *Frequency of Use: Daily use for order delivery.*
- *Subset of Product Functions Used: Access to delivery order information.*
- *Technical Expertise: Basic familiarity with the FOS mobile app for order delivery.*
- *Security or Privilege Levels: Delivery-specific access.*
- *Educational Level: Varied educational backgrounds.*
- *Experience: Experience in food delivery services.*

In order to adapt the Food Ordering System (FOS) to meet the various demands and expectations of the people who use it, it is essential to understand these user classes and their characteristics.

2.4 Operating Environment

The Food Ordering System (FOS) functions in a variety of environments:

Hardware Platform: Compatible with standard web server configurations.

Operating System: Supports Windows Server, Linux (e.g., Ubuntu, CentOS), macOS Server, and similar versions. Operating System: Supports Windows Server, Linux (e.g., Ubuntu, CentOS), macOS Server, and similar versions.

Web Server and Frameworks: Utilizes common web servers like Apache or Nginx and frameworks like Django or Node.js.

Database Management System (DBMS): Works with widely-used DBMS solutions like MySQL, PostgreSQL, SQL Server, and Oracle.

Web Browsers: Accessible through modern browsers like Chrome, Firefox, Edge, and Safari.

Mobile Devices: Mobile-friendly design ensures accessibility on smartphones and tablets.

Additional Software Components: Integrates with external systems like payment gateways, inventory management systems, CRM software, social media platforms, and marketing tools.

2.5 Design and Implementation Constraints

The Food Ordering System (FOS) has been strategically designed to accommodate a large and diverse user base, which has led to the identification of several design and implementation constraints, while also adding some unique considerations.

Programming Language: The system primarily uses Python and Django, which may limit developers with expertise in other languages.

Database Dependency: Reliance on a relational database system can restrict scalability and flexibility.

Third-party Integrations: External service integrations, like payment gateways, can affect system performance.

Security Regulations: Strict security and privacy rules influence data handling and access controls.

Limited Language Support: Expanding language options requires substantial development effort.

Payment Gateway Reliance: The FOS depends on third-party payment gateways.

Hardware and Connectivity: Users need modern web browsers and internet access.

User Training: Basic computer Internet literacy is expected.

Scalability Challenges: Growing user numbers require careful scaling.

Maintenance and Updates: Ongoing upkeep must be well-managed.

2.6 Assumptions and Dependencies

Assumptions:

- *Third-Party Services: Continuous availability and reliability of third-party services, such as payment gateways and inventory management systems, are assumed. Disruptions in these services can impact system functionality.*
- *Internet Connectivity: Users are expected to have consistent internet access to use the system effectively. Limited connectivity can affect user experience.*
- *Data Accuracy: Assumed that restaurant owners maintain accurate menu and inventory information within the system to prevent order issues.*
- *User Proficiency: Users are presumed to possess basic computer literacy and web browser familiarity for effective system interaction.*
- *Security Compliance: The assumption is that security and privacy regulations will remain relatively stable during development.*

Dependencies:

- *Database Management System (DBMS): The system relies on a specific DBMS for data storage. Changes or issues with the chosen DBMS could affect system performance.*
- *Payment Gateways: Integration with third-party payment gateways is critical for transaction processing. Changes or disruptions in these gateways may impact payment operations.*
- *Web Frameworks and Libraries: Dependencies on specific web frameworks, libraries, and development tools are assumed. Changes may necessitate adjustments in system development.*
- *External APIs: Integration with external services and APIs depends on their availability and stability.*

- *Server Infrastructure: The availability and performance of web and database servers are essential for system operation. Server-related issues could impact system responsiveness.*
- *Regulatory Compliance: Compliance with food safety and data privacy regulations is critical. Changes in regulations may necessitate updates to the system.*

3. External Interface Requirements

3.1 User Interfaces

The FOS will have two primary user interfaces: one for customers and one for restaurants. These interfaces are designed for simplicity, accessibility, and efficiency.

Customer User Interface:

- *Customers can easily browse menus, search for restaurants, place orders, track them, and provide ratings and reviews.*
- *Home Screen: Features popular restaurants and a search bar for quick access.*
- *Restaurant Menu: Allows item selection and customization.*
- *Cart: Displays selected items for review before checkout.*
- *Checkout: For entering shipping and payment details.*
- *Order Tracking: Real-time order status tracking.*
- *Rating & Reviews: Customers can rate and review restaurants.*

Restaurant User Interface:

- *Restaurants can efficiently manage orders, receive notifications, accept or reject orders, prepare and deliver them, and generate reports.*
- *Dashboard: Offers an overview of orders and detailed order information.*
- *Inventory Management: Real-time inventory updates and menu item availability management.*
- *Orders: Lists all orders with options to accept or reject and view details.*
- *Reports: Generates sales and order reports.*

Design Standards:

- *Adherence to Web Accessibility Guidelines (WCAG) 2.1 for accessibility.*
- *Implementation of Google Material Design principles for a consistent and user-friendly experience.*

Layout and Accessibility:

- *Responsive design for various devices.*
- *Supports different screen sizes and resolutions.*

Usability Features:

- *Standard buttons and functions.*
- *Keyboard shortcuts for efficient navigation.*
- *Clear and concise error message displays.*

3.2 Hardware Interfaces

The FOS interacts with the following hardware components:

- *Web Servers: Host the FOS web application, supporting user access via web browsers.*
- *Database Servers: Store and manage FOS data, including menus, orders, and user accounts.*
- *Network Equipment: Facilitate data transmission between users' devices and FOS servers.*
- *Kitchen Display Devices: Used in restaurant kitchens to receive and process food orders.*

3.3 Software Interfaces

The FOS will interact with the following software components:

- *Operating System: Linux*
- *Web Server: Apache or Nginx*

- *Database Server: MySQL or PostgreSQL*
- *Payment Gateway: Stripe or PayPal*
- *Third-party APIs: Used for mapping and social media*
- *Web Browsers: Compatible with Chrome, Firefox, Safari, Edge*

3.4 Communications Interfaces

The FOS requires the following communication interfaces:

- *HTTP/HTTPS: for web browser and web server interaction.*
- *RESTful APIs: for third-party service integration.*
- *Encryption: HTTPS for data security.*
- *Low Latency: Optimized for real-time order tracking*
- *Email Notifications: for order confirmations and updates.*
- *Database Connectivity: for linking with the database server (MySQL or PostgreSQL).*

4. System Features

This section outlines the major system features of the Food Ordering System (FOS). Each feature provides specific functionalities to cater to the needs of users, including customers, restaurant staff, and administrators.

4.1 Feature 1: User Authentication and Ordering

4.1.1 Description and Priority

This feature allows users to create accounts, log in, and place orders. It is a high-priority feature, as it is essential to the basic functionality of the Food Ordering System (FOS)

4.1.2 Stimulus/Response Sequences

- *User creates an account by providing their name, email address, and password.*
- *User logs in to their account by entering their email address and password.*
- *User browses the menus of restaurants.*
- *User adds items to their cart.*
- *User reviews their cart and proceeds to checkout.*
- *User enters their delivery address and payment information.*

- *User places their order.*
- *FOS confirms the order with the user.*
- *FOS sends the order to the restaurant.*

4.1.3 Functional Requirements

- *The FOS must allow users to create accounts.*
- *The FOS must allow users to log in to their accounts.*
- *The FOS must allow users to browse the menus of restaurants.*
- *The FOS must allow users to add items to their cart.*
- *The FOS must allow users to review their cart and make changes before placing their order.*
- *The FOS must allow users to enter their delivery address and payment information.*
- *The FOS must confirm the order with the user before placing it with the restaurant.*
- *The FOS must send the order to the restaurant in a timely manner.*
- *The FOS must handle error conditions and invalid inputs gracefully.*

Specific functional requirements:

- *REQ-1: The FOS must allow users to create accounts using their email address and password.*
- *REQ-2: The FOS must allow users to log in to their accounts using their email address and password.*
- *REQ-3: The FOS must allow users to browse the menus of restaurants without creating an account.*
- *REQ-4: The FOS must allow users to save their delivery address and payment information for future orders.*
- *REQ-5: The FOS must send a confirmation email to the user after their order has been placed*

4.2 Feature 2: Order Management and Communication

4.2.1Description and Priority

This feature allows restaurants to manage orders and communicate with customers. It is a high-priority feature, as it is essential to the core functionality of the FOS.

4.2.2 Stimulus/Response Sequences

- *Restaurant receives an order from the FOS.*
- *Restaurant accepts or rejects the order.*
- *Restaurant prepares the order.*
- *Restaurant sets the order status to "Ready for pickup" or "Ready for delivery."*
- *FOS sends a notification to the customer when their order is ready.*
- *Customer picks up their order from the restaurant or has it delivered.*
- *Customer rates and reviews the restaurant.*

4.2.3 Functional Requirements

- *The FOS must allow restaurants to accept or reject orders.*
- *The FOS must allow restaurants to receive orders from customers.*
- *The FOS must allow restaurants to prepare orders.*
- *The FOS must allow restaurants to set the order status to "Ready for pickup" or "Ready for delivery."*
- *The FOS must send a notification to the customer when their order is ready.*
- *The FOS must allow customers to pick up their orders from the restaurant or have them delivered.*
- *The FOS must allow customers to rate and review restaurants .*

Specific functional requirements:

- *REQ-1: The FOS must allow restaurants to receive orders from customers in a timely manner.*
- *REQ-2: The FOS must allow restaurants to accept or reject orders within a specified timeframe*
- *REQ-3: The FOS must provide restaurants with a way to track the status of their orders.*
- *REQ-4: The FOS must send a notification to the customer when their order is ready for pickup or delivery.*
- *REQ-5: The FOS must allow customers to rate and review restaurants on a scale of 1 to 5 stars.*

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Performance requirements must be met in order for the Food Ordering System (FOS) to meet the needs of its users. Without losing performance, the FOS should be able to handle numerous concurrent users and orders.

- *The FOS should be able to respond to user requests within 1 second.*
- *The FOS should be able to process orders within 5 seconds.*
- *The FOS should be capable of supporting up to 10,000 concurrent users.*
- *The FOS should have the capacity to handle up to 100,000 orders per day.*
- *The order placement process shall not exceed 3 minutes for the majority of users.*
- *Error messages shall be displayed to users within 5 seconds of encountering an issue.*
- *The application should operate smoothly with a memory footprint of 2 GB.*
- *A minimum of 99.99 percent of orders must successfully reach the kitchen display screen without data loss.*
- *The application's total size must not exceed 50 MB to ensure ease of download and installation.*

5.2 Safety Requirements

The Food Ordering System (FOS) must adhere to the following safety requirements to ensure the security and well-being of users and their data:

- *The FOS must protect user data from unauthorized access, disclosure, modification, or destruction.*
- *The FOS must be designed to be easy to use and understand, even for users with limited technical skill.*
- *The FOS shall comply with industry-standard encryption and security protocols to ensure the secure processing of payment transactions.*
- *The FOS must use strong authentication mechanisms to prevent unauthorized access to user accounts.*

- *The FOS shall have mechanisms in place to handle system failures, ensuring that orders are not lost or compromised during technical issues.*
- *The FOS shall adhere to relevant data protection and privacy regulations, including GDPR and PCI DSS, to safeguard user information.*
- *The FOS shall obtain necessary safety certifications and comply with industry standards to ensure the safe and secure operation of the platform.*
- *The FOS must provide users with the ability to report problems and get help.*

5.3 Security Requirements

The Food Ordering System (FOS) must be designed and implemented in a way that protects user data and privacy. The following security requirements must be met:

- *User Authentication: Users will need to prove their identity with a username and password to access their accounts.*
- *Data Encryption: The FOS must use encryption to protect user data in transit and at rest.*
- *Access Control: Only authorized personnel will have access to sensitive system areas.*
- *Regular Security Audits: The FOS must be regularly scanned for vulnerabilities and patched promptly.*
- *Privacy Compliance: The FOS should be designed to minimize data collection and storage, provide users with data control, be transparent about data collection and privacy practices, and have a process for handling data privacy complaints and concerns.*
- *Secure Connections: All communication between the system and users will be over secure and encrypted channels. The FOS must implement security measures to prevent common attacks, such as SQL injection, cross-site scripting, and denial-of-service attacks.*

5.4 Software Quality Attributes

The following qualities must be present in order for the Food Ordering System (FOS) to function properly:

- *Usability: The FOS should be simple for both customers and restaurants to use. The user interface ought to be clear, concise, and simple to use.*
- *Reliability: The FOS needs to be reliable and easily accessible all the time. The FOS should be able to handle a high volume of concurrent users and orders while maintaining performance. The FOS should also have quick and effective failure recovery capabilities.*
- *Security: The FOS must be protected against unauthorized access, disclosure, modification, and destruction of user data. Additionally, the FOS needs to be safeguarded against typical security attacks.*
- *Maintainability: The FOS should be easy to maintain and update. The FOS code should be well-organized and documented. The FOS should also be modular and extensible.*
- *Testability: The FOS should be easy to test. The FOS should be designed with unit testing, integration testing, and system testing in mind. The FOS should also have a comprehensive test suite.*

5.5 Business Rules

- *Ordering is limited to registered customers only.*
- *Restaurants can exclusively manage their own orders.*
- *Only restaurants that are open can accept orders.*
- *Administrators have oversight of all orders and restaurants.*
- *Restaurants can freely update their menu items.*
- *Restaurants can choose to deliver to specific areas only.*
- *Customers can choose between cash or credit card payments.*
- *User data is protected in accordance with privacy regulations.*
- *Order status updates are the responsibility of restaurant staff.*
- *Customers have real-time order tracking capabilities.*
- *Restaurants can set their own minimum order amount.*
- *Restaurants can promptly update inventory levels to maintain menu item availability.*
- *User feedback is encouraged through ratings and reviews for restaurants and menu items.*

6. Other Requirements

- *Database Requirements: The FOS will keep its data in a MySQL database. The database schema will be optimized for efficiency and*

normalization. In order to avoid data loss, the database will also be frequently backed up.

- *Internationalization Requirements: The FOS should be internationalized to support users from different countries. This means that the FOS should be able to display text in different languages and currencies. The FOS should also be able to handle different time zones and date formats.*
- *Legal Requirements: The FOS is required to comply with all relevant laws and rules. This consists of laws and regulations related to data security, privacy, and accessibility.*
- *Reuse Objectives: The FOS should be designed to be reusable. This means that the FOS code should be modular and extensible. The FOS should also be well-documented so that it can be easily understood and maintained.*

Appendix A: Glossary

Term	***Meaning***
FOS	*Food Ordering System*
SRS	*Software Requirements Specification*
API	*Application Programming Interface*
RDBMS	*Relational Database Management System*
GUI	*Graphical User Interface*
WCAG	*Web Content Accessibility Guidelines*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*

DBMS	*Database Management System*

Appendix B: Analysis Models

1.Use Case Diagram:

A use case diagram provides a high-level overview of the relationships between use cases, actors, and the system. It serves as a visual representation of the project's functional components.

Symbols and Notations:

The notation used for the use case diagram is straightforward and complies to the standards of UML (Unified Modeling Language):

- ***Use Cases:*** *These are represented as horizontally shaped ovals, denoting the various functionalities or actions that users can perform within the system.*

- ***Actors:*** *Actors are depicted as stick figures, symbolizing the individuals or entities interacting with the system by utilizing the use cases.*

- ***Associations:*** *Associations are depicted as lines connecting actors and use cases. In more complex diagrams, it is crucial to establish clear*

associations between actors and the specific use cases they are linked to.

- ***System Boundary Boxes:*** *A system boundary box is used to delineate the scope of the system and the encompassed use cases. Anything outside this box is considered beyond the system's scope.*

Use Case Diagram for Customer: Place Food Order

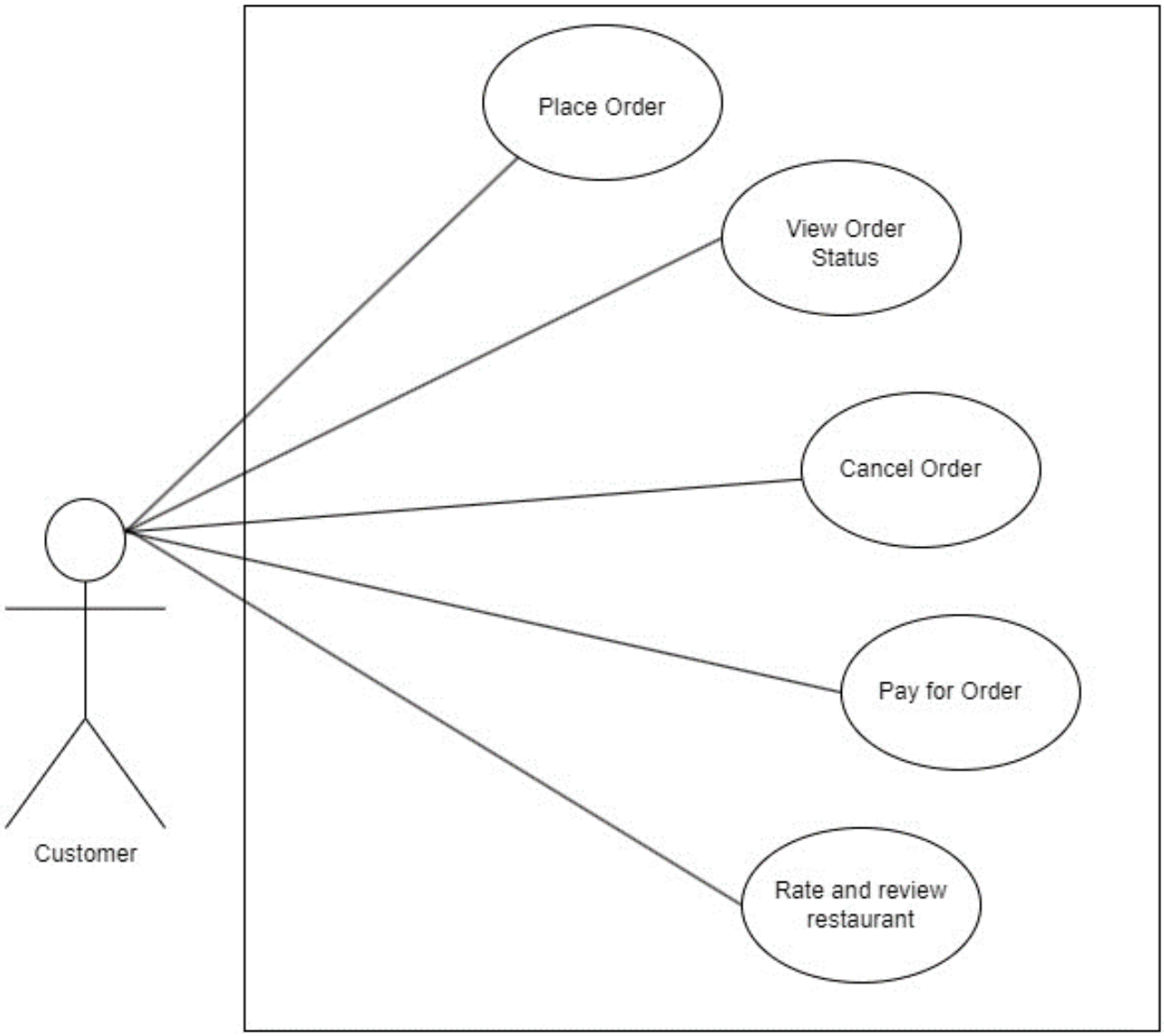
Description: This use case represents the actions taken by a customer when placing a food order through the system.

Actors: Customer

Preconditions:

- *The Customer is logged into their account.*
- *Restaurants are available for ordering.*

Main Flow:



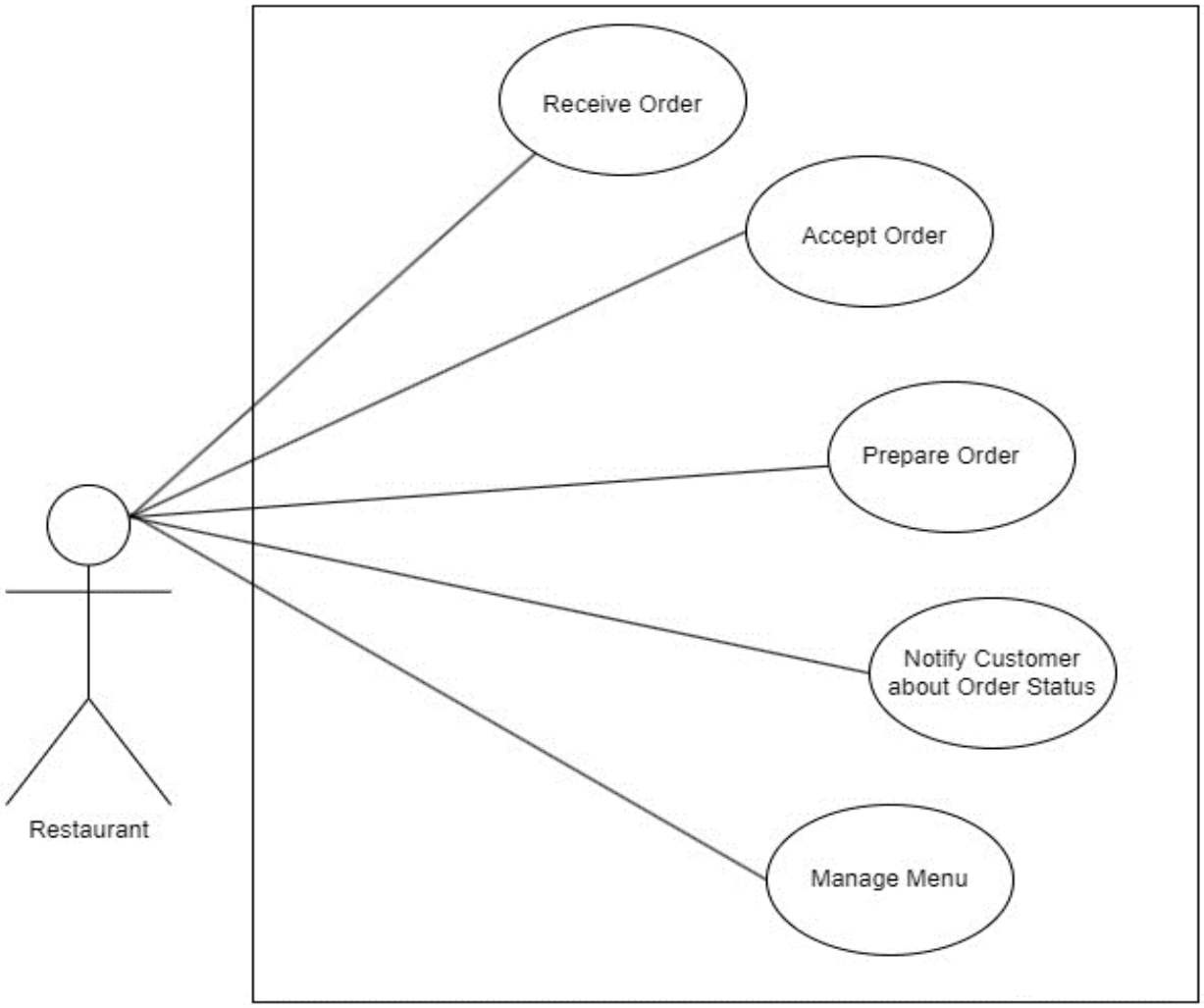
Use Case for Restaurant: Manage Orders

Description: This use case represents the actions taken by a Restaurant to manage incoming food orders from customers.

Actors: Restaurant

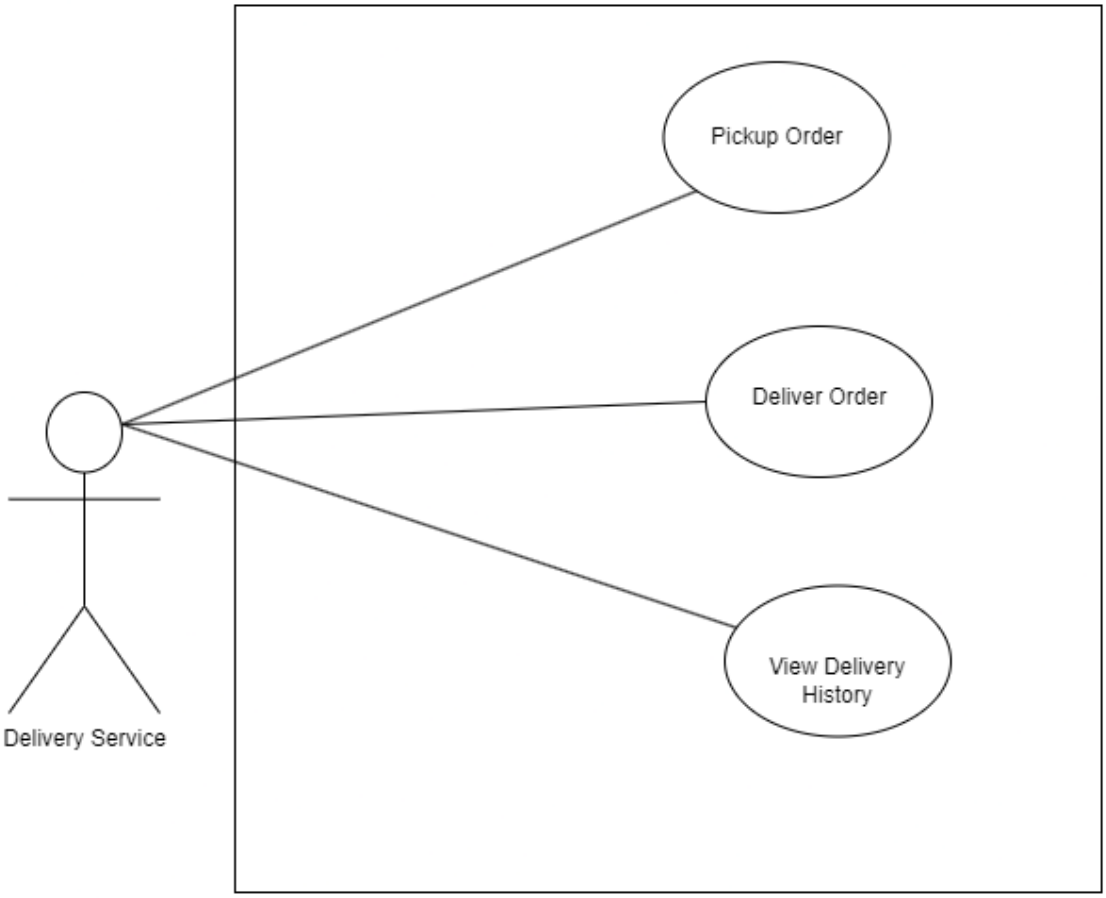
Preconditions: The Restaurant is registered and logged into the system.

Main Flow:



Use Case for Delivery Service to: Deliver Order

Description: This use case represents the actions taken by the Delivery Service to deliver food orders to the customer's provided address.



2. UML class diagram:

A UML class diagram is a defined diagram type within the Unified Modeling Language (UML) that is used to depict a system's static structure. In software engineering, it is frequently used to represent classes, their properties, interactions, and interrelationships.

Class Name: The class name dominates the first division of the diagram, highlighting its distinction within the system.

Class Attributes: The attributes are elegantly displayed in the second partition, with their respective types indicated following a colon. These

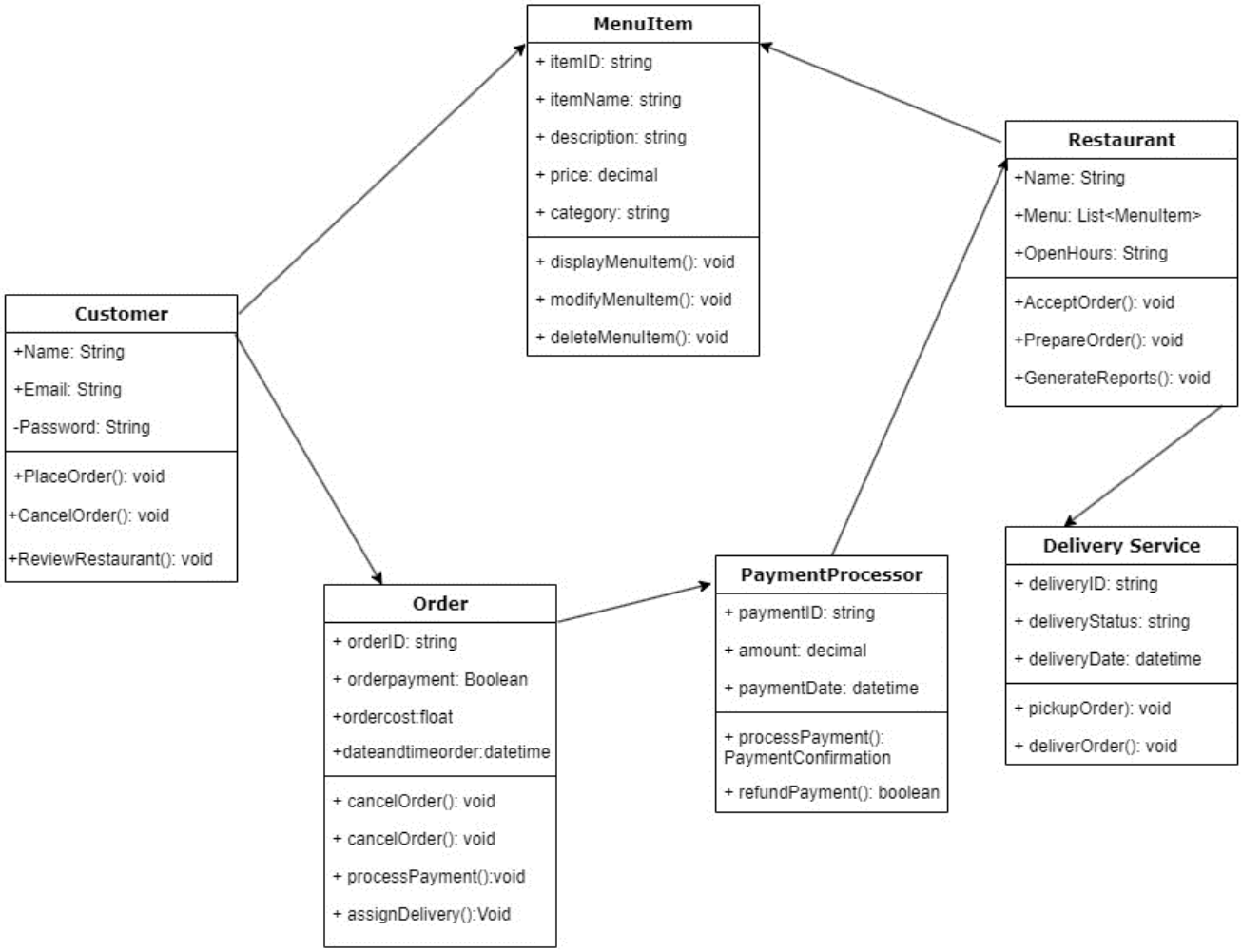
attributes correspond seamlessly to member variables in the codebase, embodying the class's essential data elements.

Class Operations (Methods):

- *Operations are gracefully depicted in the third partition, embodying the services that the class offers.*
- *The return type of a method is thoughtfully denoted after a colon, concluding the method's signature.*
- *Further clarity is achieved by specifying the return type of method parameters following a colon and parameter name. These operations seamlessly map to class methods in the underlying code, providing a bridge between the conceptual and the practical.*

Class Visibility: The visibility of class attributes and operations is elegantly conveyed through symbols:

- *The '+' symbol denotes public attributes or operations, signifying their accessibility to all.*
- *The '-' symbol discreetly marks private attributes or operations, concealing them from external entities.*
- *The '#' symbol, with its protective demeanor, represents protected attributes or operations, granting access to inherited classes while maintaining a level of controlled visibility*


MenuItem
+ itemID: string
+ itemName: string
+ description: string
+ price: decimal
+ category: string
+ displayMenuItem(): void
+ modifyMenuItem(): void
+ deleteMenuItem(): void
Restaurant
+Name: String
+Menu: List<MenuItem>
+OpenHours: String
+AcceptOrder(): void
+PrepareOrder(): void
+GenerateReports(): void
Customer
+Name: String
+Email: String
-Password: String
+PlaceOrder(): void
+CancelOrder(): void
+ReviewRestaurant(): void
Order
+ orderID: string
+ orderpayment: Boolean
+ordercost:float
+dateandtimeorder:datetime
+ cancelOrder(): void
+ cancelOrder(): void
+ processPayment():void
+ assignDelivery():Void
PaymentProcessor
+ paymentID: string
+ amount: decimal
+ paymentDate: datetime
+ processPayment(): PaymentConfirmation
+ refundPayment(): boolean
Delivery Service
+ deliveryID: string
+ deliveryStatus: string
+ deliveryDate: datetime
+ pickupOrder): void
+ deliverOrder(): void