Software Requirements Specification

for

Online Food Delivery System

Version 1.0 approved

Prepared by Bit Rebels

B Tech [Computer Science Engg]

12-Sep-2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Bit Rebels	1-9-23	*initial draft*	*1.0 draft 1*
Bit Rebels	15-9-23	*baseline following changes after inspection*	*1.0 approved*

1. Introduction

1.1 Purpose

The purpose of this SRS is to outline both the functional and non-functional requirements of the subject Online food Delivery system. In addition to said requirements, the document also provides a detailed profile of the external interfaces, performance considerations and design constraints imposed on the subsequent implementation. The document should act as a foundation for efficient and well-managed project completion and further serve as an accurate reference in the future.

1.2 Document Conventions

SRS: Software specification system
OFDS : Online Food Delivery System
DBMS : Database Management System
OS : Operating System
COD : Cash On Delivery

It is essential to establish document conventions to ensure clarity and consistency in the documentation. Here are some document conventions that can be followed:

- Font and Styling: Use a consistent font and styling throughout the document. For example, you can use a standard serif or sans-serif font such as Times New Roman or Arial. Use bold or italics for emphasis when necessary, but avoid excessive formatting.

- Headings and Subheadings: Use a hierarchical heading structure to organize the document.

- Highlighting: Consider using highlighting or shading for important notes, warnings, or critical information. For instance, you can use a yellow highlight for critical system constraints or dependencies.

- Priority: Clearly define the priority of requirements. You can use a standardized system such as:

1. High priority (Must-have): Critical features that are essential for the system to function.

2. Medium priority (Should-have): Important features that should be included but are not critical.
3. Low priority (Nice-to-have): Desirable features that would enhance the system but are not essential.

1.3 Intended Audience and Reading Suggestions

The primary audience of this SRS document will be the development team employed to implement the specified Online Food Delivery System. It will not only provide an extensive capacity for project planning and progress assessment but it will further assist with stakeholder interactions. The secondary document audience comprises the stakeholders of the project, that is, restaurateurs and associated staff. To this audience group, this SRS should convey and confirm the required functionality and represent a contractual agreement between the involved parties.

1.4 Product Scope

In current formal dining environments, some form of physical static menu is utilised to convey the available food and beverage choices to customers. Said menus are generally paper based and hence impose restrictions on the textual real estate available and the ability a restaurateur has to update them. The related concepts are encompassed by the general scope of the Online Food Delivery System. It is to the replacement of paper-based menus using an electronic format.

1.5 References

1. Rathore S. Suryadev, Chaudhary Mahik (2018). “Consumer's Perception on Online Food Ordering”. International journal of Management and Business Studies, Volume 8, Issue 4, Oct-Dec 2018, ISSN 2230-9519 (Online) and ISSN: 2231-2463

2. Panse Chetan, Rastogi Shailesh (2019), “Understanding Consumer Behaviour Towards Utilization of Online Food Delivery Platforms”

3. Sethu H.S., Saini Bhavya (2016). “Customer Perception and Satisfaction on Ordering Food via Internet”. Proceedings of the

4. Das Jyotish man (2018). “Consumer Perception Towards ‘Online Food Ordering and Delivery Services’: An Empirical

2. Overall Description

2.1 Product Perspective

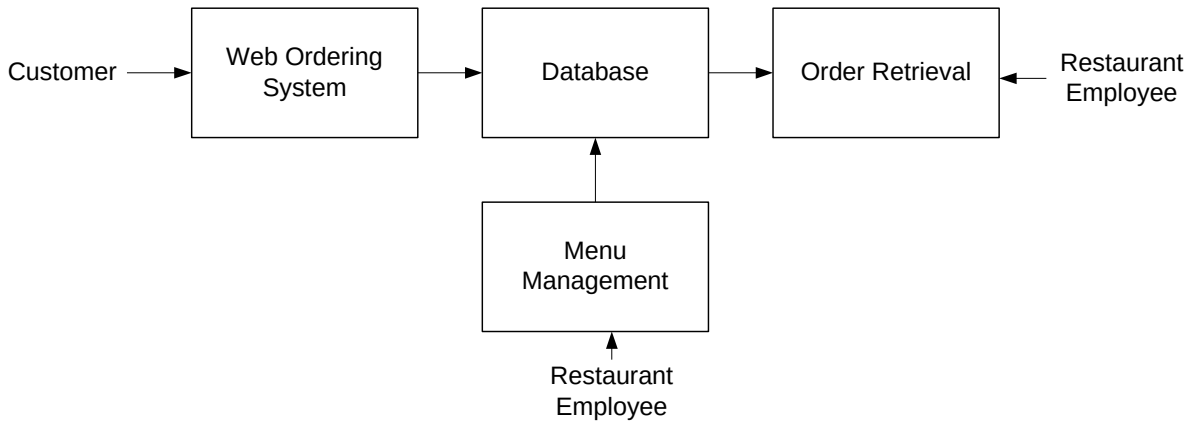
The software described in this SRS is the software for a complete Online Food Ordering System. The system merges various hardware and software elements and further interfaces with external systems. it relies on a number of external interfaces for persistence and unhandled tasks, as well as physically interfacing with humans.

2.2 Product Functions

The Online Food Delivery System interfaces with an existing payment system, including a cash register and software accessible credit system, in order to quickly and easily handle customer billing. The payment system should be operable such that it can return information to the OFDS system as to whether payment was successful or failed.

2.3 User Classes and Characteristics

There are three separate user interfaces used by the OFDS software, each related to an interfaced physical hardware device. These three user interfaces are the Surface Computer UI,Tablet UI and Display UI.



2.4 Operating Environment

The Surface Computer UI is the interface used by Online customers. This interface uses the surface computer paradigm - users interact with the system by dragging 'objects' around on the flatscreen touch-sensitive display.
The Tablet UI is designed to run on a small, wireless-enabled touch-screen tablet PC, to be used by waiters to accommodate customer needs.
The Display UI provides kitchen staff with simple functionality related to ordered items.

2.5 Design and Implementation Constraints

The OFDS should be written in an object-oriented language with strong GUI links and a simple, accessible network API. The primary candidate tool chains are Java/Swing, C++/Qt and Python/Qt.
The system must provide a capacity for parallel operation and system design should not introduce scalability issues with regard to the number of surface computers, tablets or displays connected at any one time
The system must be reliable enough to run crash and glitch free more or less indefinitely, or facilitate error recovery strong enough such that glitches are never revealed to its end-users.

2.6 User Documentation

The end-users of the OFDS fall into three primary categories, unskilled, partly skilled and highly skilled.

2.7 Assumptions and Dependencies

The SRS assumes that none of the constituent system components will be implemented as embedded applications. It is further assumed that tablet PCs of sufficient processing capability and battery life will be utilised.

3. External Interface Requirements

3.1 User Interfaces

a. Sample Screens: Provide sample screen images showcasing the main user interfaces, including the customer's order placement screen and the restaurant owner's menu management screen.

b. Layout Constraints: Ensure that screen layouts are responsive and adapt to various screen sizes and orientations, catering to both desktop and mobile users.

c. GUI Standards: The user interface should follow a responsive and user-friendly design, adhering to common GUI standards. It should provide a consistent look and feel across all screens.

d. Standard Buttons and Functions: Include standard buttons like "Order Now," "Add to Cart," "Checkout," and "View Order History." Common functions such as "Search," "Filter," and "Profile Management" should also be present.
e. Error Handling: Design clear and user-friendly error messages for scenarios like payment failures or order issues.

f. Keyboard Shortcuts: Provide keyboard shortcuts for power users to navigate the interface efficiently.

g. Feedback Handling: Define a process for handling user feedback and complaints, including a dedicated support interface or contact information.

3.2 Hardware Interfaces

a. Device Types: The software should be compatible with various device types, including smartphones, tablets, laptops, and desktop computers. It must support both iOS and Android platforms for mobile devices.

b. Data and Control Interactions: The software interacts with hardware components for input and output. For example, it receives GPS data from mobile devices to determine the user's location and sends orders to receipt printers in restaurants for order confirmation.

d. Communication Protocols: The system uses standard internet protocols such as HTTP/HTTPS for web-based interactions, GPS protocols for location tracking, and Bluetooth for connecting with peripherals like receipt printers.

3.3 Software Interfaces

A. Database Interface:

Database Management System (DBMS): The online food delivery system interacts with a specific DBMS, such as MySQL, PostgreSQL, or MongoDB, for data storage and retrieval.

Data Items In: Data items coming into the system include customer profiles, restaurant information, menus, order details, user reviews, and transaction records.

Data Items Out: Data items leaving the system encompass order confirmations, delivery status updates, customer ratings, and payment records.

Purpose: The database interface manages the storage and retrieval of critical data required for order processing, user management, and reporting.

B. Operating System Interface:

Operating System (OS): The software runs on specific operating systems, such as Windows Server, Linux, or macOS.

Data Sharing: Data shared with the operating system includes file I/O operations for logging, handling system resources, and managing processes.

Purpose: The operating system interface ensures the compatibility and resource management of the online food delivery system.

C. Operating System Interface:

Operating System (OS): The software runs on specific operating systems, such as Windows Server, Linux, or macOS.

Data Sharing: Data shared with the operating system includes file I/O operations for logging, handling system resources, and managing processes.

Purpose: The operating system interface ensures the compatibility and resource management of the online food delivery system.

D. Operating System Interface:

Operating System (OS): The software runs on specific operating systems, such as Windows Server, Linux, or macOS.

Data Sharing: Data shared with the operating system includes file I/O operations for logging, handling system resources, and managing processes.

Purpose: The operating system interface ensures the compatibility and resource management of the online food delivery system.

3.4 Communications Interfaces

A. Email Communication:

Requirement: The system sends email notifications to users for order confirmations, updates, and promotions.

Message Format: HTML or plain text emails with order details and links for tracking orders.

Security: Use secure SMTP protocols to send emails and ensure data privacy.

B. Web Browser Communication:

Requirement: Users access the system through web browsers on various devices.

Communication Protocol: HTTP/HTTPS for data transfer between web clients and servers.

Security: Implement SSL/TLS encryption for secure data transmission.

C. Network Server Communication Protocols:

Requirement: The system relies on server-client communication for real-time order tracking and updates.

Communication Protocol: WebSocket or MQTT for efficient real-time communication.

Data Transfer Rates: Low latency for real-time updates and high throughput for order processing.

D. Electronic Forms:

Requirement: Users input delivery details, payment information, and order preferences through electronic forms.

Form Data Format: JSON or XML for structured data exchange.

Validation: Implement data validation and error handling for form submissions.

E. APIs and Web Services:

Requirement: Integration with external services like payment gateways, Google Maps, and social media platforms.

API Protocols: Use RESTful APIs and specific protocols provided by third-party services.

Security: Implement API keys and authentication mechanisms as required by external services.

4. System Features

4.1 User Registration and Authentication

4.1.1 Description and Priority

This feature allows users to create an account and log in to the online food delivery system. It is of High priority as it forms the foundation for user interaction with the platform.

4.1.2 Stimulus/Response Sequences

User navigates to the registration page.

User provides necessary information (name, email, password, etc.).

System validates the information and creates a user account.

User logs in using their credentials.

4.1.3 Functional Requirements

REQ-1: The system shall provide a user registration form with fields for name, email, password, and contact information.

REQ-2: The system shall validate the email format to ensure it is in a valid format.

REQ-3: The system shall enforce password requirements (e.g., minimum length, special characters).

REQ-4: The system shall store user account information securely.

REQ-5: The system shall provide a login form with fields for email and password.

REQ-6: The system shall verify user credentials during the login process.

REQ-7: The system shall securely handle authentication tokens and sessions.

REQ-8: The system shall allow users to reset their password through a secure process.

REQ-9: The system shall provide error messages for invalid login attempts.

4.2 Browse Restaurants and Menus

4.2.1 Description and Priority

This feature enables users to view a list of available restaurants and their menus. It is of High priority as it is a core functionality of the platform.

4.2.2 Stimulus/Response Sequences

User navigates to the "Browse Restaurants" section.

System fetches and displays a list of restaurants.

User selects a restaurant to view its menu.

System retrieves and displays the menu for the selected restaurant.

4.2.3 Functional Requirements

REQ-10: The system shall provide a page to browse available restaurants.

REQ-11: The system shall display a list of restaurants with their names, cuisines, and average ratings.

REQ-12: The system shall allow users to filter restaurants by cuisine, ratings, and location.

REQ-13: The system shall display a menu for each restaurant, showing items, prices, and descriptions.

REQ-14: The system shall allow users to add items to a cart for ordering.

REQ-15: The system shall update the cart in real-time as items are added or removed.

REQ-16: The system shall allow users to customize their orders (e.g., specify toppings, special requests).

REQ-17: The system shall calculate and display the total order cost.

4.3 Real-Time GPS Tracking

Real-time GPS tracking is one of the essential features of food delivery apps that enable customers to track the location of their food in real-time. The purpose of GPS is to offer two-way

tracking as well as functioning. It helps to recognize the user's location to deliver the food. And once the area is confirmed, users can easily track the progress and movement of the delivery personnel.

4.4 Easy Payment Options

You can incorporate all the payment gateways or mobile wallet app services available in the market, such as Google Pay, PayPal, Amazon Pay, iOS Wallet, Stripe, Credit/Debit Card, Online Banking, and Cash On Delivery (COD) alternatives. You can likewise offer promo or voucher code usage from a similar page.

4.5 Reviews & Ratings

Every business cares for its customers as they're the ones who ultimately buy products or services. Hence, it's essential to know whether an on-demand app meets their needs or not. Every online food delivery application must have a review and rating feature. It would help your customers rate and review various restaurants with their listed dishes on the application.

4.6 Order scheduling and pickup

A person is working in the office, and their wife rings them up, saying she wants Chinese for dinner. Eating out is not an option because all the restaurants are packed. So, they check out a food ordering app and scroll for Chinese cuisine from the menu. After scrolling and a call with the wife, they order Kung Pao Chicken, Mongolian Beef, and Hunan Shrimp

4.7 Order History

Suppose you ordered a pizza from Scarr's that you enjoyed a lot. A few weeks later, you want to order the same pizza again. What would you do? Search for the name of the restaurant in the food app. You can, but what if you forget its name? We all do in the day-to-day hustle and bustle. That's why the order history feature is very efficient. Users can review their previous orders and place their favorite orders again.

4.8 In-app Messages

In-app messages are the alerts that appear on users' screens while the app is open. The idea is to deliver targeted and context-sensitive information.

The in-app messages feature is helpful in scenarios when communication between restaurants and customers becomes essential. For example, sometimes customers want to clarify specifics with the administration or inform the restaurant that they delivered a wrong order by mistake.

4.9 Contactless delivery

Customers have become cautious about the human touch post the COVID pandemic, and the fear of infection scares them often. Hence, more and more of them now prefer a contactless food ordering experience.

Seeing this, many famous online food delivery applications like Doordash and Pizzahut have started offering contactless delivery experiences. They eliminate direct contact by ensuring that the delivery person knocks or rings the customer's doorbell and then steps back 6-feet. This way, the customers can ensure their safety while enjoying the best food delivered to their doorstep.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should be able to process a high volume of orders efficiently.
This could include system speed, the quantity of orders it can process at once, and the ability to handle peak periods of activity.

5.2 Safety Requirements

The system should be available and working when required, with as little downtime as possible.
This could include requirements for the system's ability to handle failures or unforeseen events, as well as the utilisation of backup systems and processes to assure service continuity.

5.3 Security Requirements

The system should prevent unauthorised access or misuse of sensitive information, such as consumer payment and personal information.
This could include regulations for the use of encryption, secure servers, and other data integrity safeguards. The system should include a disaster recovery plan to ensure business continuity in case of unexpected outages or failures.

5.4 Software Quality Attributes

With a clear and well-documented codebase and a solid testing and deployment procedure, the system should be simple to upgrade and maintain over time.
This could include requirements for using version control, automated testing, and other tools and processes to keep the system reliable and up to date.

5.5 Business Rules

1. All requests made from a mobile app or UI will go to different services via the API gateway. API gateway will take care of load balancing and routing requests to services. This will authenticate and authorize the user and send back the token ID. This token is used for further communication

2. Different services like, user registration and management service, order service, payment service will use transactional databases. We will use the Amazon Aurora relational database. This is a highly scalable database service to manage users and concurrent orders etc.

3. Information about different restaurants, their menu, price, offers, etc will be stored in JSON document storage in ElasticSearch. We can use a multi-node cluster here. Whenever a customer searches for a menu/cuisines it will be fetched from elastic search. Elastic search provides fast scalable search options

4. Once the user selects the dishes and quantity from the restaurant. He will go to the checkout option and then do payment. Different payment gateways and payment options are integration with the system and upon successful payments, the order is successfully placed

5. Once the order is placed all the information is sent to the central message Queue like Kafka. The order processing unit reads the order info and then notifies the selected restaurant about the order. At the same time, it searches for available delivery partners to nearby locations to pick up the order. It also gets the information like preparation time from the restaurant and estimated pickup time from the delivery partner based on his location and other details. it will select the best available delivery partner and he is notified about order and restaurant details

6. The user gets push notification about the order. The order processing and tracking service will work together and the user can track their order status, live location of the delivery person, etc

7. Delivery person pickup order and deliver to customers. Customer is real-time notified with ETA for the order

6.Other Requirements

1. Usability: The system should be simple to use for both customers and restaurant employees, with a clear and intuitive interface and simple navigation.
   This could include criteria for the system's layout and design, the use of clear and simple language, and the provision of assistance and support.

2. Scalability – it refers to the system's ability to accommodate increases in the number of users or orders without deteriorating performance.
   This could include the capacity to add more servers or other hardware as needed to accommodate rising demand.

Appendix A: Data Dictionary and Data Model

delivery instruction	=	patron name patron + phone number + meal date delivery + location + delivery time window
delivery location	=	* building and room to which an ordered meal is to be delivered *
delivery time window	=	* 15-minute range during which an ordered meal is to be delivered; must begin and end on quarter-hour intervals *
employee ID	=	* company ID number of the employee who placed a meal order; 6character numeric string *
food item description	=	* text description of a food item on a menu; maximum 100 characters *
food item price	=	* pre-tax cost of a single unit of a menu food item, in dollars and cents *
meal date	=	* the date the meal is to be delivered or picked up; format MM/DD/YYYY; default = current date if the current time is before the order cutoff time, else the next day; may not be prior to the current date *
meal order	=	meal order number + order date meal + date + 1:m{ordered food item} + delivery instruction + meal order status
meal order number	=	* a unique, sequential integer that the system assigns to each accepted meal order; initial value is 1 *
meal order status	=	[incomplete | accepted | prepared | pending delivery | delivered | canceled] * see state-transition diagram in Appendix B *

meal payment	=	payment amount + payment method + (payroll deduction transaction number)
menu	=	menu date + 1:m{menu food item} + 0:1{special}
menu date	=	* the date for which a specific menu of food items is available; format MM/DD/YYYY *
menu food item	=	food item description + food item price
order cutoff time	=	* the time of day before which all orders for that date must be placed *
order date	=	* the date on which a patron placed a meal order; format MM/DD/YYYY *
ordered food item	=	menu food item + quantity ordered
patron	=	patron name + employee ID patron + phone number + patron location + patron e-mail
patron e-mail	=	* e-mail address of the employee who placed a meal order; 50 character alphanumeric *
patron location	=	* building and room numbers of the employee who placed a meal order; 50 character alphanumeric *
patron name	=	* name of the employee who placed a meal order; 30 character alphanumeric *
patron phone number	=	* telephone number of the employee who placed a meal order; format AAA-EEE-NNNN xXXXX for area code, exchange, number, and extension *
payment amount	=	* total price of an order in dollars and cents, calculated per BR-12 *
payment method	=	[payroll deduction | cash] * others to be added beginning with release
payroll deduction transaction number	=	* 8-digit sequential integer number that the Payroll System assigns to each payroll deduction transaction that it accepts *

quantity ordered	=	* the number of units of each food item that the Patron is ordering; default = 1; maximum = quantity presently in inventory *
special	=	special description
	+	special price
		* the Menu Manager may define one or more special meals for each menu, with a particular combination of food items at a reduced price *
special description	=	* text description of a daily special meal; maximum 100 characters *
special price	=	* cost of a single unit of a daily special meal, in dollars and cents *

Appendix B: Analysis Models

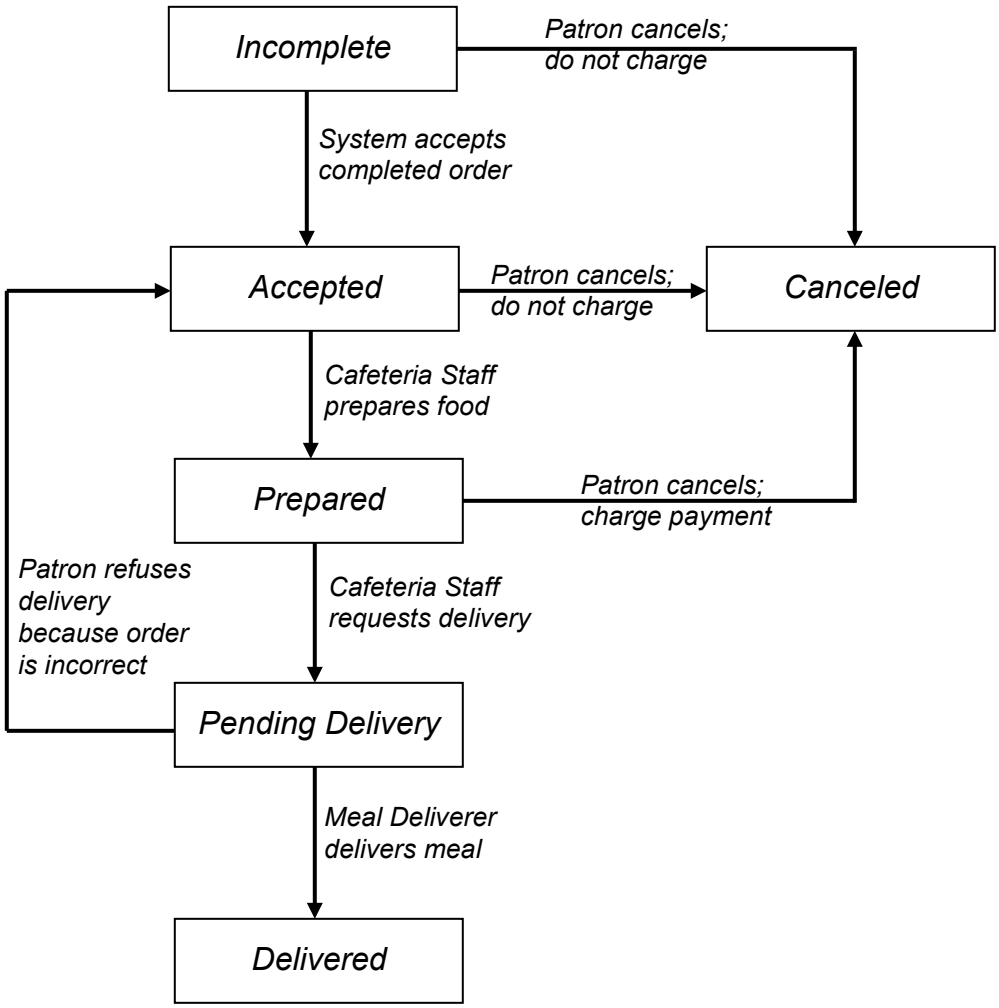


Figure *State-transition diagram for food delivery.*

Appendix C: To Be Determined List

1. TBD: Define the specific user authentication method (e.g., username/password, biometric, two-factor authentication).
2. TBD: Determine the acceptable payment gateways and integration options.
3. TBD: Specify the exact geographic regions for service coverage and delivery zones.
4. TBD: Identify the supported mobile devices and operating systems for the customer mobile application.
5. TBD: Determine the precise hardware and software requirements for the server infrastructure.
6. TBD: Specify the loyalty program details, including rewards and redemption criteria.
7. TBD: Define the process for handling customer feedback and complaints.
8. TBD: Identify the legal and compliance requirements related to food delivery services.
9. TBD: Determine the data retention and privacy policies for user data.
10. TBD: Specify the exact roles and permissions for different types of users (e.g., customers, drivers, restaurant owners).
11. TBD: Define the algorithm for order assignment to delivery drivers.
12. TBD: Specify the communication channels and notifications for order status updates.
13. TBD: Determine the process for handling and resolving order disputes.
14. TBD: Identify the third-party services or APIs required for map integration and location-based services.
15. TBD: Define the procedure for handling menu updates and changes by restaurants.
16. TBD: Specify the customer support hours and contact methods.
17. TBD: Determine the error-handling and recovery mechanisms for system failures.
18. TBD: Identify the criteria and process for onboarding new restaurants onto the platform.
19. TBD: Specify the exact process for calculating delivery fees and charges.
20. TBD: Define the quality assurance and testing procedures for the system.