

---

**SD Food – Team J**

---

**AI-Enabled Online Restaurant Order and  
Delivery System  
Software Requirements Specification**

**Version <1.0>**

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

## Revision History

Date	Version	Description	Author
21/10/25	1.0	Initial draft of the Software Requirements Specification for the AI-Enabled Restaurant Order and Delivery System.	Damilola Babajide, Sri Suvetha Meenaa Subramanian, Sanskriti Khadka, Shreyosee Chowdhury

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

## Table of Contents

1.	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Definitions, Acronyms, and Abbreviations	4
1.4	References	4
1.5	Overview	4
2.	Overall Description	4
2.1	Use-Case Model Survey	5
2.2	Assumptions and Dependencies	5
3.	Specific Requirements	5
3.1	Use-Case Reports	5
3.2	Supplementary Requirements	5
4.	Supporting Information	5

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

# Software Requirements Specification

## 1. Introduction

### 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to describe in detail the functions and requirements of our AI-Enabled Online Restaurant Order and Delivery System. This document serves as a reference for developers, testers, clientele, etc., to understand the scope and behavior of the system.

The system will enable customers to browse the restaurant menus, place online orders, and have their food delivered by assigned delivery employees based on ratings. It includes roles for registered customers, VIP customers, visitors, chefs, delivery staff, and one manager. In addition to this, it also integrates a language model (LLM)-based AI chatbot that provides automated responses to users using a local knowledge base and if needed it turns to a free, open-source LLM.

The SRS ensures all requirements are clearly described before the system is designed and implemented in order to avoid any ambiguities or disparities and to maintain consistency for all readers throughout the development process.

### 1.2 Scope

The AI-Enabled Online Restaurant Order and Delivery System is an interactive platform designed to digitize restaurant operations and enhance customer experience.

The system will:

- Allow visitors to browse menus, ask questions to the chatbot, and register as customers
- Allow registered customers/VIP to login to order food, rate dishes, file complaints or compliments, and join discussion forums
- Allow chefs to independently decide and manage their menus and prepared dishes
- Allow delivery staff to bid for delivery orders and complete deliveries
- Allow a manager to handle customer accounts, manage HR decisions regarding hiring, firing, or receiving bonuses, and to settle complaints.
- Integrate an AI chatbot that answer customer or visitors' questions using a local KB or an external free LLM such as one from Ollama or Hugging Face.

### 1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- LLM: Large Language Model
- VIP Customer: A registered customer who has spent over \$100 or completed at least 3 successful orders without complaints
- KB (Knowledge Base): A local data base storing questions and pairing answers used to respond to user questions for the chatbot
- GUI: Graphical User Interface
- API: Application Programming Interface

### 1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- Ollama Documentation- <https://ollama.com/library>
- Hugging Face – <https://huggingface.co/docs/transformers/index>

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

- Flask Documentation - <https://flask.palletsprojects.com/en/stable/>
- SQLite Database Documentation – <https://www.sqlite.org/docs.html>
- Flet Documentation – <https://flet.dev/docs/>
- Python Documentation - <https://docs.python.org/3/>

## 1.5 Overview

The rest of the document is organized as follows:

- Section 2 describes the **Overall Description** and provides a general overview of the system's functionality, user characteristics, assumptions, and dependencies
- Section 3 defines the **Specific Requirements** of the system in detail. It includes both the functional and nonfunctional requirements necessary for system design and system testing.
- Section 4 provides **Supporting Information** such as appendices, user interface prototypes, and use case storyboards. This section may include some more references to any tools, datasets, or resources used during development.

## 2. Overall Description

The AI-enabled Online Restaurant Order and Delivery System is a multi-user intelligent application that facilitates restaurant operations through digital ordering, automated reputation management, and AI-assisted customer interaction. The system is designed to support three categories of users: employees (chefs, delivery staff, and managers), customers (registered and VIP), and visitors.

The system architecture includes a database for managing orders, menus, user credentials, and feedback, along with an AI module that leverages a local or hosted LLM (such as Ollama or Hugging Face) to respond to customer queries. The graphical user interface (GUI) provides functionalities such as browsing menus, placing orders, tracking deliveries, and interacting with the chatbot. The overall goal of this design is to provide a scalable, efficient, and interactive environment that enhances both customer satisfaction and restaurant workflow.

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

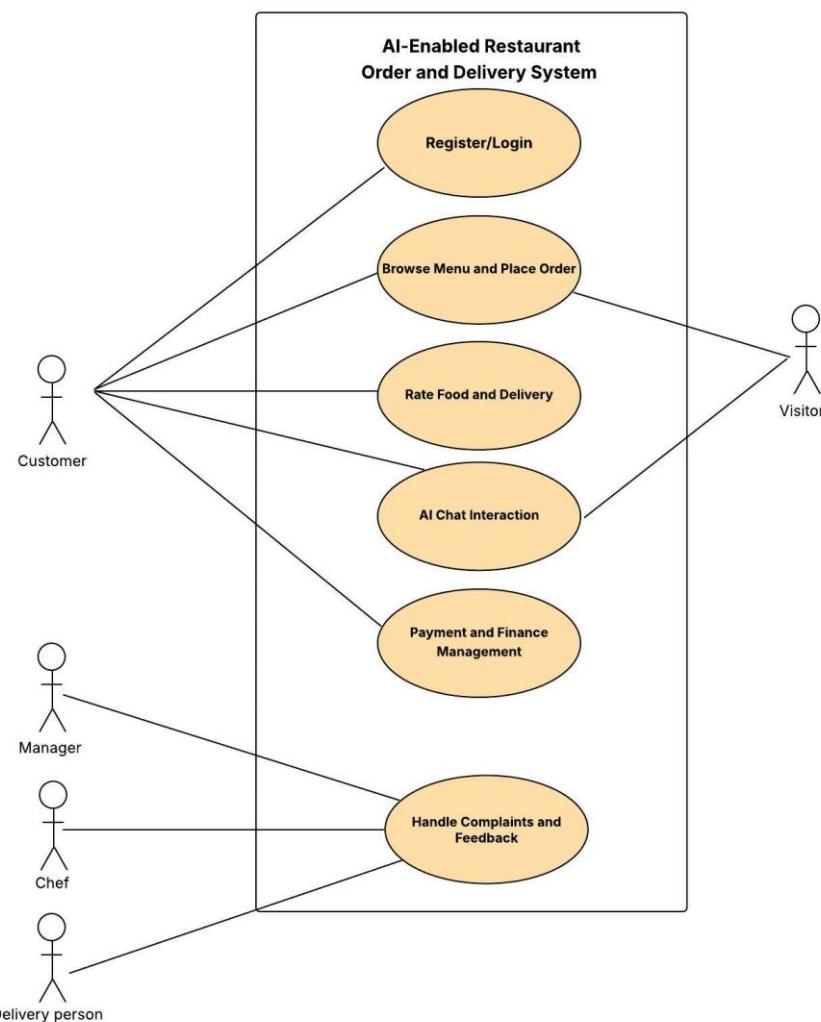
## 2.1 Use-Case Model Survey

### Actors:

- Customer: Registers, orders food, rates dishes, and interacts with the chatbot.
- Manager: Oversees operations, resolves complaints, manages employees.
- Chef: Adds or modifies dishes and receives feedback.
- Delivery Person: Delivers food and updates delivery status.
- Visitor: Views menu and applies for registration.

### Main Use Cases:

- Register/Login
- Browse Menu and Place Order
- Rate Food and Delivery
- Handle Complaints and Feedback
- AI Chat Interaction
- Payment and Finance Management



<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

## 2.2 Assumptions and Dependencies

- The system will operate on Windows 11 and Linux Ubuntu 22.04 environments using Visual Studio Code or equivalent IDEs.
- A local relational database (SQLite) will be used to store user data, menus, orders, and feedback.
- The AI customer support module depends on a large language model (LLM) accessible locally via Ollama or through Hugging Face API; thus, internet connectivity is assumed when using external APIs.
- The system requires Python 3.10 or higher (for backend and AI integration) and relevant libraries such as Flask, sqlite3, and transformers.
- Payment handling assumes integration with standard payment gateways or a simulated wallet system for testing.
- The creative feature (image-based food search or route optimization) depends on external libraries such as YOLO, DINO, or Google Maps APIs.
- It is assumed that users have basic technical literacy, stable network access, and compatible devices (desktop or laptop) to run the GUI.
- The restaurant organization maintains at least one manager, two chefs, and two delivery personnel for system operation.

## 3. Specific Requirements

This subsection describes each major use case identified in the use-case model survey. Each use case is documented with its preconditions, main flow, alternative flows, postconditions, and associated functional requirements.

### 3.1 Use-Case Reports

This subsection describes each major use case identified in the use-case model survey. Each use case is documented with its preconditions, main flow, alternative flows, postconditions, and associated functional requirements.

#### UC-01: Register/Login

**Description:** Visitors register and users log in to access the system.

**Main Flow:**

- Visitors submit the registration form. Manager approves the request. Account is created.
- User logs in with credentials. System redirects to the appropriate dashboard based on role.
- Returning customers see personalized content; new users see popular dishes.
- Warnings are displayed if present.

**Requirements:**

- FR-01.1: Unique usernames and encrypted passwords.
- FR-01.2: Role-based dashboards with personalized recommendations.
- FR-01.3: Display warnings prominently.
- FR-01.4: Manager approval required for registrations.
- FR-01.5: Auto VIP upgrade after \$100 spent or 3 successful orders (no complaints).

#### UC-02: Browse Menu and Place Order

**Description:** Users browse dishes, and customers place orders.

**Main Flow:**

- Users view dishes with images, prices, ratings, and chef information.

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

2. They can search or filter by category, price, rating, or chef.
3. Customer selects items. System checks wallet balance and applies a 5% VIP discount if applicable.
4. Order is confirmed. System updates spending total and checks for VIP upgrade.
5. Delivery staff are notified to bid. VIPs receive one free delivery for every three paid orders.

**Requirements:**

- FR-02.1: Personalized menu with search and filtering options.
- FR-02.2: Validate wallet balance and issue warning for insufficient funds.
- FR-02.3: Automatically apply VIP discount (5%) and free delivery.
- FR-02.4: Track spending and trigger auto VIP upgrade.
- FR-02.5: Restrict access to VIP-exclusive dishes.

**UC-03: Rate Food and Delivery**

1. Description: Customers rate food and delivery separately (1–5 stars).
2. **Main Flow:**
  3. Customer opens completed order and rates food quality and delivery quality separately.
  4. Optional feedback can be added.
  5. VIP ratings count double.
  6. System updates averages and flags employees for bonuses or demotions based on trends.

**Requirements:**

- FR-03.1: Separate rating options (1–5 stars); VIP ratings count double.
- FR-03.2: Real-time update of average scores; prevent duplicate ratings.
- FR-03.3: Automatically flag HR actions based on rating trends.

**UC-04: Handle Complaints and Feedback**

**Description:** Users file complaints or compliments; the manager reviews and resolves them.

**Main Flow:**

1. User submits a complaint or compliment. Manager reviews the case.
2. Manager decides to uphold, dismiss, or find partial merit.
3. If upheld: The subject receives a warning. VIP complaints count as two warnings.
4. If dismissed: The complainant receives one warning for a meritless complaint.
5. Compliments can cancel one complaint. Three compliments trigger a bonus review.
6. Customers with 3 warnings are blacklisted. VIPs with 2 warnings are downgraded.
7. Employees with 3 upheld complaints are demoted; a second demotion leads to termination.
8. Subjects can dispute once before the manager makes a final decision.

**Requirements:**

- FR-04.1: Validate user interactions and apply VIP double weighting.
- FR-04.2: Auto-enforce penalties and allow one dispute per case.
- FR-04.3: Track all complaints and notify involved users.
- FR-04.4: Allow 1:1 complaint cancellation with compliments.
- FR-04.5: Maintain a permanent blacklist for removed customers.

**UC-05: AI Chat Interaction**

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

**Description:** Users ask questions through the AI chatbot.

**Main Flow:**

1. User enters a question in the chat box.
2. System searches the local knowledge base (KB) first.
3. If an answer is found, it is labeled “From Restaurant Knowledge Base,” and the user rates it (0–5).
4. If rated 0, the answer is flagged for manager review. Manager may remove it and ban the author from contributing.
5. If no KB answer exists, the system uses a free LLM (Ollama or Hugging Face) and labels the response “AI Assistant.”
6. All chat interactions are logged.

**Requirements:**

- FR-05.1: Search the KB before using LLM; show the source of the answer.
- FR-05.2: Collect ratings and flag poorly rated answers.
- FR-05.3: Restrict authors who post inappropriate content.
- FR-05.4: Use a free, open-source LLM and handle errors gracefully.

### **UC-06: Payment and Finance Management**

**Description:** The system manages deposits, payments, refunds, and delivery bidding.

**Main Flow:**

1. Customer adds funds to their wallet; system updates the balance.
2. During order payment, the system deducts the amount and applies discounts automatically.
3. If balance is insufficient, order is rejected and one warning is issued.
4. When an account is closed, the manager refunds remaining balance and blacklists kicked-out customers.
5. For delivery bidding, drivers submit bids, and the manager assigns the order, usually to the lowest bidder.
6. If a higher bid is selected, the manager must write a short justification.

**Requirements:**

- FR-06.1: Track deposits, deductions, and refunds.
- FR-06.2: Issue warnings for failed payments; auto-apply discounts.
- FR-06.3: Log all transactions and account closures.
- FR-06.4: Support delivery bidding; require justification for non-lowest bids.

## **3.2 Supplementary Requirements**

This subsection captures additional requirements that are not fully addressed in the use cases, including non-functional requirements, design constraints, and system-wide functional requirements.

### **Performance**

The system responds within 2 seconds for most actions, with menus loading in under 1 second. The chatbot replies instantly from the knowledge base and within 5 seconds when using the LLM. It can support over 50 users at once without lag.

### **Security**

Passwords are encrypted with bcrypt and must have at least 8 characters with letters and numbers. Role-based access controls limit user permissions, and blacklisted users cannot register again.

### **Usability**

The interface is simple, intuitive, and consistent across pages. Clear error messages guide users, and the Confidential ©<Company Name> Page 9

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

system runs smoothly on both Windows 11 and Ubuntu 22.04.

### Reliability

The system maintains 95% uptime, performs daily backups, and handles errors gracefully. If the external LLM fails, the chatbot automatically switches to the local knowledge base.

### Technical and Design Constraints

Developed with Python 3.10+, Flask, SQLite, and Flet. It uses a free local LLM from Ollama or Hugging Face and is desktop-based rather than web-based. Each dish includes an image and uses a 1–5 star rating scale. The system requires two chefs, two delivery people, and one manager.

### Data Requirements

All transactions, user history, and chatbot interactions are stored permanently. Deleted dishes are archived, and all complaint and compliment records remain accessible to the manager.

## 4. Supporting Information

### Index

1. Use-Case Storyboards
2. User Interface Prototypes
3. Database Overview
4. References and Notes

#### 1. Use-Case Storyboards

- Register/Login: Visitor registers → Manager approves → User logs in → Redirected to dashboard.
- Browse Menu & Order: Customer views menu → Filters/searches dishes → Adds to cart → Pays → Delivery assigned.
- AI Chat Interaction: User asks question → System checks knowledge base → If not found, uses LLM → User rates response.

#### 2. User Interface Prototypes

- Login Page: Username, password, role selection, buttons for Register/Login.
- Customer Dashboard: Menu, My Orders, Wallet, Chatbot, Feedback.
- Chef Dashboard: My Dishes, Add Dish, Ratings, Chatbot.
- Delivery Dashboard: Available Orders, My Deliveries, Earnings.
- Manager Dashboard: Users, Complaints, HR, Finances, Reports.

#### 3. Database Overview

Table	Description
Users	Stores usernames, passwords, and roles.
Menu	Contains dish info (name, price, chef, rating).
Orders	Tracks all orders and delivery details.
Ratings	Stores food and delivery ratings.
Complaints	Logs user complaints and manager actions. (optional)
Wallet	Tracks user balances and transactions. (optional)
ChatLogs	Records chatbot questions and responses.

<Project Name>	Version: <1.0>
Software Requirements Specification	Date: <dd/mmm/yy>
<document identifier>	

#### 4. References and Notes

- Python 3.10+, Flask, SQLite, Flet, Hugging Face, Ollama
- Runs on Windows 11 or Ubuntu 22.04
- Requires Internet for external AI use
- Appendices are part of the system requirements