Software Requirements Specification
for
AI-Powered Smart Kitchen & Waste Minimizer

Charotar University of Science and Technology

August 19, 2025

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the features, behavior, and requirements for the AI-Powered Smart Kitchen & Waste Minimizer system (hereafter referred to as "the System"). Its purpose is to establish a clear agreement on the product to be built between the stakeholders, project managers, and the development team. This document will serve as the guiding reference for all design, development, and testing activities. The core problem this System addresses is the significant financial loss and environmental impact caused by food waste in the commercial restaurant industry, which stems from inefficient inventory management, inaccurate demand forecasting, and suboptimal menu planning.

1.2 Document Conventions

- This SRS follows the IEEE 830 standard format.
- Functional requirements are labeled as FR-x and Non-functional requirements as NFR-x.
- All priorities are explicitly mentioned for each feature (High, Medium, Low).
- Technical terms are defined in the Glossary section (Appendix A).

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Project Managers**: To define the project's scope and for planning and tracking.
- **Software Developers**: To understand the system's architecture, features, and technical constraints.
- **UI/UX Designers**: To create intuitive and effective user interfaces based on the specified user personas and functional requirements.
- **QA & Testing Teams**: To develop test cases and quality assurance plans.
- **Stakeholders**: To serve as a definitive agreement on the capabilities of the final product.

Suggested reading order: Sections 1 and 2 for general understanding, Sections 3 and 4 for technical details, and Sections 5–6 for quality and operational constraints.

1.4 Product Scope

The project's primary objective is to leverage Artificial Intelligence to provide restaurant operators with a tool that directly reduces food waste and increases profitability.

In Scope:

- A web and mobile-responsive application for system interaction.
- Automated inventory tracking using images from pre-installed cameras.
- Predictive analytics for ingredient demand and potential waste.

- Dynamic menu and recipe suggestions based on inventory status.
- A comprehensive dashboard for visualizing waste data and its financial impact.
- Role-based user access control for managers and kitchen staff.

Out of Scope:

- The sale, installation, or maintenance of camera hardware.
- Direct integration with all third-party Point-of-Sale (POS) systems (though data import/-export will be supported).
- Full financial accounting or payroll management.
- Employee scheduling and human resources functionalities.

1.5 References

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640
2. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
3. FastAPI Documentation. https://fastapi.tiangolo.com/
4. PostgreSQL Documentation. https://www.postgresql.org/docs/

2 Overall Description

2.1 Product Perspective

The System is a new, self-contained software-as-a-service (SaaS) product. It operates independently but is designed to complement a restaurant's existing operational toolkit. It is composed of a cloud-hosted backend, a database, AI/ML microservices, and a web-based frontend. It interfaces with on-premise hardware (IP cameras) to gather visual data from the kitchen environment.

2.2 Product Functions

- **Computer Vision Inventory Management**: Automates inventory counts and spoilage checks using camera images.
- **AI-Powered Prediction**: Forecasts ingredient demand, warns of potential waste, and suggests purchase orders.
- **Intelligent Menu Optimization**: Suggests menus and recipes to utilize surplus stock and analyzes dish profitability.
- **Waste Analysis & Reporting**: Tracks waste events, provides interactive charts, and calculates financial losses.
- **User Management**: Handles user accounts, roles, and access control.

- **Dashboard Views**: Provides role-specific overviews for inventory, predictions, menus, and reports.
- **Real-time Notifications**: Alerts for low stock, nearing expiry, or waste risks.

2.3 User Classes and Characteristics

- **Restaurant Manager**: Increases profitability, reduces waste; comfortable with web apps but needs actionable insights; uses dashboards for planning.
- **Head Chef/Kitchen Staff**: Manages kitchen efficiency; limited technical proficiency; interacts via mobile for quick inventory views and waste logging.

2.4 Operating Environment

- **Platforms**: Web (React/Flutter), mobile-responsive.
- **Databases**: PostgreSQL.
- **Networks**: Cloud-hosted (AWS/GCP).
- **Hardware**: IP cameras with 1080p resolution and RTSP support.

2.5 Design and Implementation Constraints

- **C1**: Backend must use Python 3.9+ with FastAPI; frontend must use a modern JavaScript framework (e.g., React).
- **C2**: System requires client-provided IP cameras with 1080p resolution and RTSP support.
- **C3**: All data handling must comply with data protection regulations (e.g., GDPR, CCPA).
- **C4**: AI/ML models must use open-source frameworks (e.g., PyTorch, TensorFlow).
- **C5**: No internet access for package installation; use pre-installed libraries.

2.6 User Documentation

- Comprehensive README.md with installation and configuration instructions.
- In-app guides for camera setup, data uploads, and dashboard usage.
- Developer documentation for extending modules.

2.7 Assumptions and Dependencies

- **A1**: Users have stable internet and consistent camera lighting.
- **A2**: Staff will undergo basic training and provide historical data for predictions.
- **D1**: Depends on cloud hosting services (e.g., AWS) for availability.
- **D2**: Prediction accuracy relies on quality of provided sales data.

3 External Interface Requirements

3.1 User Interfaces

The UI will be a clean, modern web application. Key screens include:

- **Dashboard**: A high-level overview screen showing KPIs: estimated daily waste cost, inventory alerts, top-selling items, and menu suggestions.
- **Inventory Page**: A detailed tabular and visual view of all ingredients, their quantities, status, and expiry dates.
- **Reports Page**: An interactive analytics page with date filters and various charts for waste, sales, and profitability analysis.
- **Menu Engineering Page**: A section to manage menu items, view their costs/profits, and see AI-driven suggestions.
- **Settings Page**: For managing users, cameras, and account information.

3.2 Hardware Interfaces

The system will interface with standard IP cameras. It must support the RTSP (Real-Time Streaming Protocol) for video stream ingestion. The system does not directly control the camera hardware (e.g., pan, tilt, zoom).

3.3 Software Interfaces

- **RESTful API**: Built with FastAPI for client-server communication.
- **Database**: PostgreSQL for data storage.
- **AI/ML Services**: PyTorch/TensorFlow for image processing and predictions.

3.4 Communications Interfaces

- HTTPS/TLS 1.2 or higher for secure communication.
- Email for sending daily risk reports and notifications.

4 System Features

4.1 Computer Vision Inventory Management

4.1.1 Description and Priority

Automates inventory tracking using images from pre-installed cameras to identify ingredients and monitor spoilage. Priority: High.

4.1.2 Stimulus/Response Sequences

- The system triggers a scan based on a schedule (e.g., every 4 hours).
- The system captures a high-resolution image from the target camera stream.

- The image is sent to the AI/ML Inference Service.
- The service identifies ingredients and counts them.
- The service returns the identified items and their counts to the Backend API.
- The backend updates the ingredients table in the database.

Exception Flow:

- **Unrecognized Item**: If an item is detected but not recognized, it is flagged for manual identification by a manager.
- **Blurry Image**: If the image quality is too low for processing, the scan is aborted, and a notification is sent to the manager.

4.1.3 Functional Requirements

- **FR-INV-1** (High): The system shall allow managers to configure camera streams by providing an IP address or RTSP URL.
- **FR-INV-2** (High): The system shall capture images from configured streams on a user-defined schedule (e.g., hourly, daily) or via a manual trigger.
- **FR-INV-3** (High): The image processing model must identify registered ingredients with a precision of at least 90% under optimal lighting conditions.
- **FR-INV-4** (Medium): The system shall provide a UI where managers can review scanned images and correct any misidentified items or incorrect counts. This feedback shall be used to improve the model over time.
- **FR-INV-5** (High): The spoilage detection model shall classify produce into 'Fresh', 'Nearing Expiry', and 'Spoiled' categories with an accuracy of at least 85%.

4.2 AI-Powered Prediction

4.2.1 Description and Priority

Provides predictive analytics for ingredient demand and potential waste, including purchase order suggestions. Priority: High.

4.2.2 Stimulus/Response Sequences

- User uploads historical sales data.
- System processes data with AI model.
- System generates 14-day demand forecasts.
- System sends daily risk reports via email.

4.2.3 Functional Requirements

- **FR-PRED-1** (High): The system shall provide an interface for uploading historical sales data in CSV format.

- **FR-PRED-2** (High): The demand forecasting model shall generate daily predictions for the next 14 days, including confidence intervals for each prediction.
- **FR-PRED-3** (Medium): The system shall generate a "Daily Risk Report" email, listing the top 5 ingredients most likely to be wasted that day.
- **FR-PRED-4** (High): The purchase order suggestion algorithm must account for current stock, predicted demand, supplier lead times, and a configurable safety stock level.

4.3 Intelligent Menu Optimization

4.3.1 Description and Priority

Suggests menus and recipes based on inventory to minimize waste and maximize profitability. Priority: High.

4.3.2 Stimulus/Response Sequences

- User views current inventory.
- System suggests menu items based on stock.
- System updates dish costs in real-time.

4.3.3 Functional Requirements

- **FR-MENU-1** (High): The menu suggestion feature must allow filtering based on cuisine type (e.g., Italian, Mexican) and dish type (e.g., appetizer, main course).
- **FR-MENU-2** (High): The system shall calculate dish costs in real-time. If the cost of an ingredient changes, the cost of all associated menu items must be updated automatically.
- **FR-MENU-3** (Medium): The system shall display a profitability matrix, plotting menu items on a graph of popularity vs. profitability (Stars, Puzzles, Plowhorses, Dogs).

4.4 Waste Analysis & Reporting

4.4.1 Description and Priority

Tracks and analyzes waste events, providing visual dashboards and financial impact reports. Priority: High.

4.4.2 Stimulus/Response Sequences

- Staff logs waste manually or via camera.
- System compiles data into charts.
- User filters and views reports.

4.4.3 Functional Requirements

- **FR-WASTE-1** (High): The system shall provide a simple interface for kitchen staff to manually log waste events not captured by the camera (e.g., a spilled container).

- **FR-WASTE-2** (High): The dashboard shall feature interactive charts allowing users to filter waste data by date range, ingredient, reason, and kitchen station.
- **FR-WASTE-3** (Medium): The financial loss report must be convertible to different currencies based on user settings.

4.5 User Management

4.5.1 Description and Priority

Manages user accounts and role-based access for restaurant staff. Priority: High.

4.5.2 Stimulus/Response Sequences

- Manager creates or modifies user accounts.
- System enforces authentication policies.
- User resets password if forgotten.

4.5.3 Functional Requirements

- **FR-USR-1** (High): The system shall allow a user with the 'Manager' role to invite, create, modify, and deactivate user accounts for their restaurant.
- **FR-USR-2** (High): The system shall enforce a strong password policy (minimum length, character types).
- **FR-USR-3** (Medium): The system shall provide a "forgot password" functionality that sends a secure reset link to the user's registered email.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- **NFR-1** (High): The system shall support up to 50 concurrent users per restaurant branch without significant degradation in performance.
- **NFR-2** (High): Median API response time for database CRUD operations shall be less than 200ms.
- **NFR-3** (High): AI model inference for a single image must complete in under 5 seconds.

5.2 Safety Requirements

- **NFR-4** (High): The system shall comply with data protection regulations (e.g., GDPR, CCPA) to prevent data leaks and ensure user privacy.

5.3 Security Requirements

- **NFR-5** (High): All communication will be encrypted using HTTPS/TLS 1.2 or higher.
- **NFR-6** (High): User passwords will be hashed using a modern algorithm (e.g., Argon2, bcrypt).

- **NFR-7** (High): The system will be protected against common web vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and CSRF, by leveraging FastAPI's built-in protections and following security best practices.
- **NFR-8** (High): Data at rest (in the database) will be encrypted.

5.4 Software Quality Attributes

- **NFR-9** (High): Reliability: The system shall have a target uptime of 99.8%.
- **NFR-10** (High): Usability: Designed with a mobile-first approach; new staff can learn basic operations in under 15 minutes.
- **NFR-11** (High): Maintainability: Backend code adheres to PEP 8, documented with comments/docstrings, and containerized with Docker.
- **NFR-12** (High): Testability: CI/CD pipeline with automated tests achieving at least 75% code coverage.

5.5 Business Rules

- **BR-1**: An ingredient is considered "Nearing Expiry" if its expiry date is within a manager-configurable number of days from the current date (default: 3 days).
- **BR-2**: Low stock alerts are triggered when an ingredient's quantity falls below its 3-day predicted consumption level plus a 20% safety buffer.

6 Other Requirements

- **Data Model**: Entity-Relationship Diagram (ERD) with relationships: a restaurant has many users, ingredients, and menu items; menu items link to ingredients via recipe components; sales history and waste logs link to menu items and ingredients.
- **Database Schema**: Detailed CREATE TABLE statements for all entities with constraints and field descriptions.
- **Testing Strategy**: Includes unit tests, integration tests, end-to-end (E2E) tests, performance tests, and User Acceptance Testing (UAT).
- **API Example (Waste Logging)**:
  - **Endpoint**: POST /api/v1/waste/log/image
  - **Description**: Submits an image from a waste bin camera for analysis.
  - **Authentication**: Required (Manager or Staff role).
  - **Request Body**: multipart/form-data containing the image file.
  - **Success Response (202 Accepted)**:

```
{
  "status": "Processing initiated",
  "taskId": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8"
}
```

- **Error Response (400 Bad Request)**:

```
{
  "error": "Invalid image format. Please use JPEG or PNG."
}
```

Appendix A: Glossary

- **AI**: Artificial Intelligence
- **ML**: Machine Learning
- **CV**: Computer Vision
- **SRS**: Software Requirements Specification
- **API**: Application Programming Interface
- **UI**: User Interface
- **YOLO**: You Only Look Once; a real-time object detection model.
- **LSTM**: Long Short-Term Memory; a type of recurrent neural network.
- **NLG**: Natural Language Generation
- **POS**: Point of Sale
- **REST**: Representational State Transfer
- **JSON**: JavaScript Object Notation
- **PostgreSQL**: An open-source object-relational database system.
- **FastAPI**: A modern, fast web framework for building APIs with Python.
- **UAT**: User Acceptance Testing

Appendix B: Analysis Models

- **Use Case Diagram**: Actors include Restaurant Manager and Head Chef/Kitchen Staff. Use cases include Configure Camera Streams, Upload Sales Data, View Menu Suggestions, Log Waste, and Manage Users.
- **Use Case Description (Example: Inventory Scan)**:
  - **Preconditions**: Cameras are online and correctly configured. Ingredient profiles exist in the system.
  - **Postconditions**: The quantity of identified ingredients is updated in the database. A log of the scan is created.
  - **Main Flow**:
    1. The system triggers a scan based on a schedule (e.g., every 4 hours).
    2. The system captures a high-resolution image from the target camera stream.

3. The image is sent to the AI/ML Inference Service.
4. The service identifies ingredients and counts them.
5. The service returns the identified items and their counts to the Backend API.
6. The backend updates the ingredients table in the database.

– **Exception Flow**:

* **3a. Unrecognized Item**: If an item is detected but not recognized, it is flagged for manual identification by a manager.
* **3b. Blurry Image**: If the image quality is too low for processing, the scan is aborted, and a notification is sent to the manager.

- **Entity-Relationship Diagram (ERD)**:
  – A restaurant has many users.
  – A restaurant has many ingredients and menu items.
  – A menu item is made up of many ingredients (via a recipe components join table).
  – Sales history and waste logs are linked to menu items and ingredients respectively.

Appendix C: To Be Determined List

- Detailed database schema (CREATE TABLE statements).
- Specific third-party API integrations for data import/export.
- Additional AI model configurations for future enhancements.