Software Requirement Specification (SRS)

Real Time Ingredients Recognition and Recipe Generator

1. Introduction

1.1 Purpose

The rise of digital technology and a growing interest in cooking has led to an increased demand for intelligent systems that can assist with meal preparation. A key challenge for both novice and experienced cooks is the need for a practical way to find recipes that use the ingredients they have on hand. This project aims to solve this problem by developing a system that can not only identify ingredients from a live image but also generate a new, contextually relevant recipe. We will use two state-of-the-art deep learning models: **YOLOv12** for real-time object detection and **BERT** for natural language understanding and generation. The system will leverage a pre-existing CSV file of recipes to provide the foundation for its recipe generation capabilities.

1.2 Scope

The scope of this project is to create a functional prototype that performs the following tasks:

- **Ingredient Recognition**: The system will use a trained YOLOv12 model to detect and classify a predefined list of ingredients from an image or a live video feed. This involves capturing an image, processing it, and outputting a list of identified ingredients.
- **Data Integration**: The list of detected ingredients will be passed to the recipe generation component. This component will use a CSV file as its database.
- **Recipe Generation**: The system will use a fine-tuned BERT model to generate a recipe, including a title, a list of ingredients, and step-by-step instructions. The generated recipe will be based on the ingredients identified by the YOLO model and the data within the CSV file.
- **Real-Time Processing**: The system will be designed to process images and generate recipes in near real-time, providing a seamless user experience.

1.3 Definitions, Acronyms, Abbreviations

- **BERT:** Bidirectional Encoder Representations from Transformers. A transformer-based machine learning model for natural language processing (NLP).
- **CSV:** Comma-Separated Values. A simple file format used to store tabular data.

- **Object Detection:** A computer vision technique that locates and identifies objects in an image or video.
- **Real-time:** A system that responds to input within a very short, specific, and predictable time frame.
- **YOLO:** You Only Look Once. A family of real-time object detection models. YOLOv12 is a recent version of this model.

2. Overall Description

2.1 Product Perspective

This system is a real-time, ingredient-aware recipe generator. It functions as a standalone application or as a feature integrated into a larger platform (e.g., a smart kitchen device or a mobile cooking app). The system's primary role is to bridge the gap between having ingredients and finding a suitable recipe, streamlining the meal preparation process for users. It is designed to be an intelligent, interactive tool that enhances the user's culinary experience.

2.2 Product Functions

- **Real-time Ingredient Recognition:** The system must accurately identify and list ingredients from a live video feed or a static image.
- **Recipe Generation:** Based on the detected ingredients, the system will generate a new, contextually relevant recipe.
- **Recipe Database Access:** The system will use a CSV file as its core recipe database. It will process this data to understand the relationships between ingredients and recipes.
- **User Interface (UI):** A simple, intuitive UI will display the recognized ingredients and the generated recipe. The UI should be responsive and easy to navigate.

2.3 User Classes

- **Home Cooks:** Individuals who frequently cook at home and want a quick way to find recipes based on available ingredients. They value convenience and efficiency.
- **Cooking Enthusiasts:** Users who enjoy experimenting with new recipes and ingredients. They would use the system to discover novel combinations and reduce food waste.
- **Novice Cooks:** People who are new to cooking and require step-by-step guidance. The system would assist them in finding simple recipes they can easily prepare.

- **Smart Kitchen Device Users:** Individuals who have kitchen gadgets (e.g., smart refrigerators, smart displays) and would use this system as an integrated feature.

2.4 Operating Enviroment

The system is designed to operate in a **real-time environment**, processing data from a live camera feed. It requires a device with a **camera and a processor capable of running deep learning models**. This could be a:

- **Personal Computer (PC):** With a webcam and a GPU for accelerated processing.
- **Mobile Device (Smartphone/Tablet):** With an integrated camera. The models may need to be optimized for mobile processors.
- **Smart Appliance:** Such as a smart refrigerator or a smart kitchen hub.

The software will be developed on a **Python-based platform** and will likely use libraries like TensorFlow or PyTorch. It requires access to the **pre-trained YOLOv12 and BERT models** and the **recipe CSV file**.

2.5 Constraints

- **Performance:** The system must provide results in **near real-time**. Latency for both ingredient detection and recipe generation must be minimal.
- **Accuracy:** The **YOLOv12 model must have high accuracy** in identifying ingredients, and the **BERT model must generate coherent and relevant recipes**.
- **Database Dependency:** The **quality and variety of generated recipes are limited by the contents of the CSV file**. If an ingredient combination is not in the CSV, the system may struggle to generate a relevant recipe.
- **Scalability:** The system's ability to handle a larger number of ingredients and recipes is dependent on the size of the models and the database.
- **Maintenance:** The models will need to be **periodically updated and retrained** to improve accuracy and expand the range of identifiable ingredients.

3. System Features

A real-time ingredient recognition and recipe generation system combines the power of computer vision and natural language processing to deliver a seamless culinary experience. The system's

features can be broken down into three main categories: ingredient recognition, recipe generation, and the overall system capabilities.

3.1 Ingredient Recognition Features

- **Real-time Object Detection:** The system's core feature is its ability to instantly identify ingredients from a live camera feed or an uploaded image. It must accurately detect objects, even when they're partially obscured or in a cluttered environment.
- **High Accuracy:** The YOLOv12 model used for detection is chosen for its state-of-the-art accuracy in identifying a wide range of objects. This is critical for correctly identifying ingredients like different types of vegetables, fruits, and pantry staples.
- **Scalability:** The system should be able to recognize a large and growing number of ingredients. The YOLOv12 model can be fine-tuned on a custom dataset to expand its recognition capabilities beyond common food items.
- **Bounding Box and Confidence Score:** When an ingredient is detected, the system will not only identify it but also draw a bounding box around it and provide a confidence score, indicating the probability of the detection being correct.

3.2 Recipe Generation Features

- **Contextual Understanding**: The BERT model, fine-tuned on a CSV file of recipes, is the brain behind the recipe generation. It understands the semantic relationships between ingredients, allowing it to generate recipes that make sense and are contextually relevant.
- **Coherent and Structured Output**: The generated recipe is more than just a list of ingredients. The BERT model will produce a structured output that includes a **recipe title**, a list of **ingredients**, and clear, step-by-step **cooking instructions**.
- **Recipe Retrieval and Generation**: The system can function in two ways. First, it can retrieve existing recipes from the CSV file that match the detected ingredients. Second, it can use the detected ingredients as a prompt to **generate a new, unique recipe** based on the patterns it learned during training.
- **Adaptability**: The system can be designed to handle ingredient substitutions or recommend similar recipes if a perfect match isn't found in the database.

3.3 Overall System Features

- **Real-time Performance:** The entire system—from capturing the image and detecting ingredients to generating the recipe—is designed for minimal latency, providing a near-instantaneous response.
- **Modular Architecture:** The use of separate YOLO and BERT models allows for a modular design. This means each component can be updated or replaced independently without affecting the entire system. For example, if a more advanced object detection model becomes available, it can be swapped in for YOLOv12.
- **Resource Efficiency:** YOLOv12's optimized architecture and BERT's fine-tuning on a specific task help in reducing computational overhead, making the system more efficient and suitable for deployment on a wider range of hardware, from cloud servers to edge devices.

4. External Interface Requirements

4.1 User Interfaces

The system's user interface will be simple and direct. It will have an input area where users can either use their camera for real-time recognition or upload a photo of their ingredients. After the ingredients are detected, the system will display the list of what it found, and with a click, it will show the generated recipe with a title, ingredients, and clear instructions for cooking.

4.2 Hardware Interfaces

To function properly, the system needs access to specific hardware. A high-quality camera is essential for capturing clear images of ingredients, whether it's a webcam or a smartphone's camera. Because the YOLO and BERT models require a lot of processing power, a powerful GPU is needed to make sure the system works in real-time. A good CPU is also required to manage all the data and the overall application.

4.3 Software Interfaces

The system will run on common operating systems like Windows, macOS, and mobile platforms. Its core functionality is built on deep learning frameworks like TensorFlow or PyTorch, which handle the complex models. It will also include a component to read and process the recipe data stored in a simple CSV file. The system can even be designed with a special API, allowing other applications to connect and use its features.

4.4 Communication Interfaces

For the system to work effectively, especially if the deep learning models are stored on a cloud server, it must have a stable internet connection. This network link allows it to securely send data (like the image of ingredients) to the server and receive the generated recipe back, all using standard internet protocols.

5. Non-Functional Requirements

Non-functional requirements describe the quality and operational characteristics of the system, focusing on how well it works rather than what specific tasks it performs. These are crucial for a successful product because a system that is fast, reliable, and secure is much more valuable than one that simply performs its functions.

5.1 Performance Requirements

This is about speed and efficiency. The system needs to be fast enough to be useful in a kitchen environment. When a user points the camera at ingredients, the system should identify them and generate a recipe almost instantly, without any noticeable lag. This is particularly important for real-time applications where a delay would make the system frustrating to use.

5.2 Reliability

This refers to the system's ability to consistently perform as expected under various conditions. It should not crash or produce nonsensical results if an ingredient isn't recognized or if the lighting is poor. The generated recipes must be logical and consistent, and the system should provide helpful error messages or alternative suggestions when it encounters a problem.

5.3 Usability

The system should be easy and intuitive for anyone to use, regardless of their technical skill or cooking experience. The user interface must be clean and simple, with clear instructions and easy-to-able-to-read text. The output should be well-organized, so the user can quickly find the title, ingredients, and steps without confusion.

5.4 Scalability

This requirement ensures the system can grow and handle more data in the future. As the recipe database grows with more ingredients and recipe combinations, the system should still be able to perform quickly. The architecture must be designed to accommodate these increases without needing a complete overhaul.

5.5 Security

If the system connects to the internet to access models or data, it must do so securely. Any data transmitted, such as images or user queries, needs to be protected from unauthorized access. The system must also have safeguards to prevent malicious attacks or data corruption.

6. System Models

System models are visual representations that help to understand, analyze, and design a system. They are crucial in software engineering to communicate complex ideas clearly among developers, stakeholders, and clients. Each type of diagram focuses on a different aspect of the system.

6.1 Use Case Diagram

A use case diagram shows the interactions between users (actors) and the system. It illustrates the different ways a user can interact with the system to achieve a goal. It's a high-level view that outlines the system's functional requirements from the user's perspective. For our project, a use case diagram would show a "User" actor interacting with use cases like "Recognize Ingredients" and "Generate Recipe".

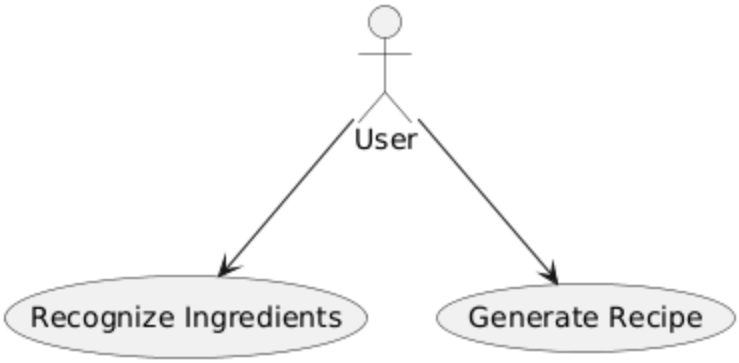

Fig 6.1.1 Use Case Diagram

6.2 Data Flow Diagram (DFD)

A **data flow diagram** visually traces the flow of data through a system. It shows where data comes from, where it goes, and how it is transformed in the process. DFDs do not show control flow (e.g., decisions or loops), but rather the **data's journey**. A DFD for our system would show data flowing from the "Camera" to the "YOLOv12 Model," then to the "BERT Model," and finally to the "User Interface" as a recipe.

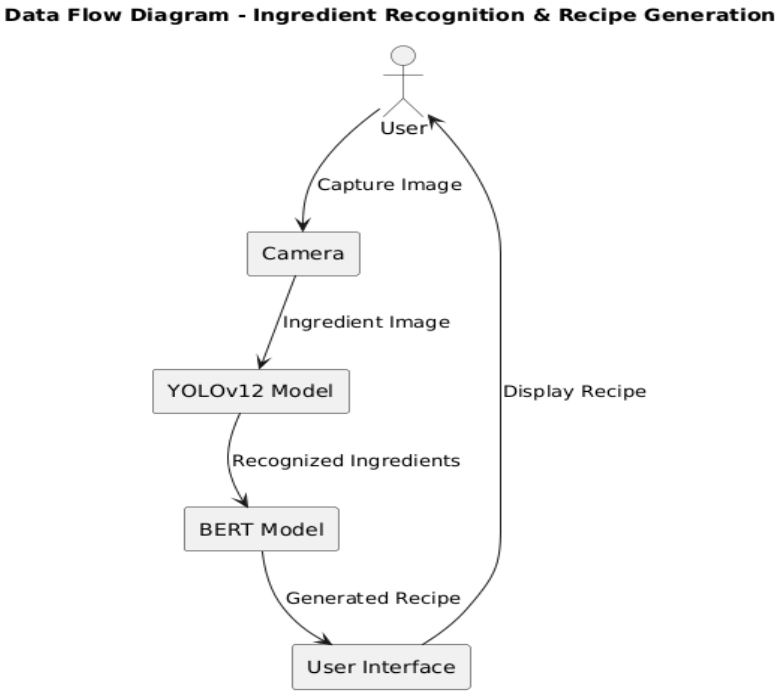

Fig 6.2.1 Data Flow Diagram

6.3 Class Diagram

A **class diagram** represents the static structure of a system by showing its classes, their attributes, methods, and the relationships between them. It is a fundamental part of **object-oriented design**. In our project, a class diagram would define classes like "Ingredient," "Recipe," "YOLOModel," and "BERTModel," detailing their properties and how they relate to each other (e.g., a "Recipe" class would have an association with multiple "Ingredient" classes).

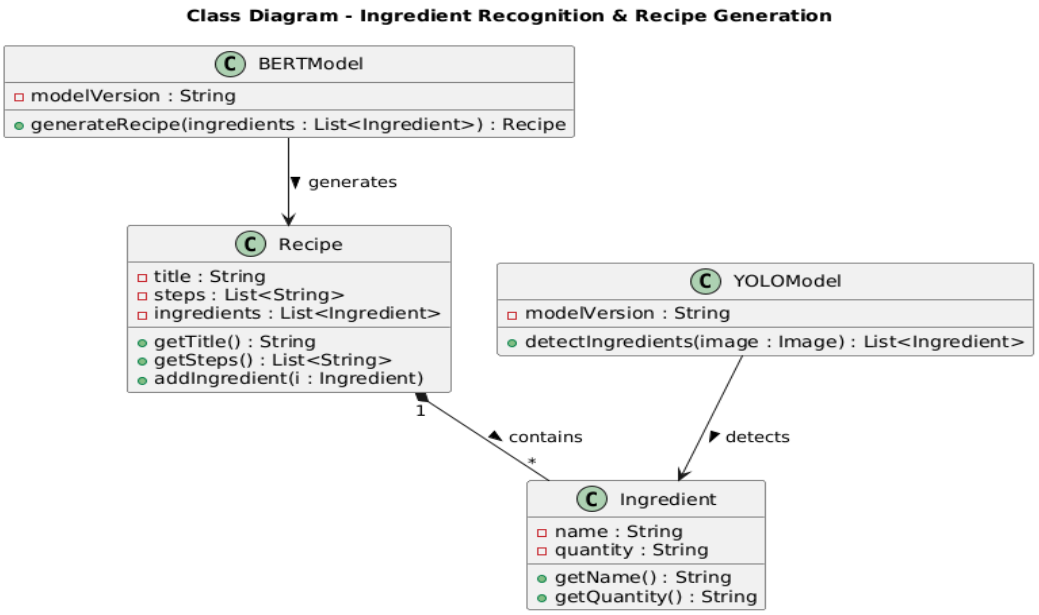

Fig 6.3.1 Class Diagram

6.4 Sequence Diagram

A sequence diagram shows the interactions between objects in a time-ordered sequence. It is used to model the logic of a specific scenario or use case. The vertical lines represent the lifespan of objects, while horizontal arrows show the messages (method calls) exchanged between them. For our system, a sequence diagram would detail the steps for generating a recipe: the user's request, the YOLO model's processing, the data transfer to the BERT model, and the final display of the recipe.

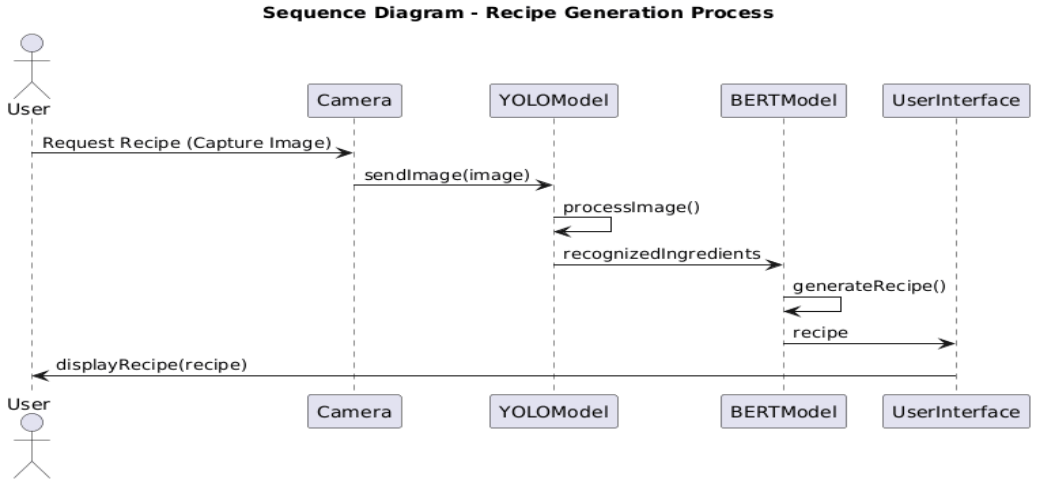

Fig 6.4.1 Sequence Diagram

6.5 ER Diagram (Entity-Relationship Diagram)

An **ER diagram** models the logical structure of a database. It shows the entities (data objects), their attributes (properties), and the relationships between them. It is essential for **database design**. For this project, an ER diagram would show entities like "Ingredient" and "Recipe," with relationships such as "A recipe contains one or more ingredients."

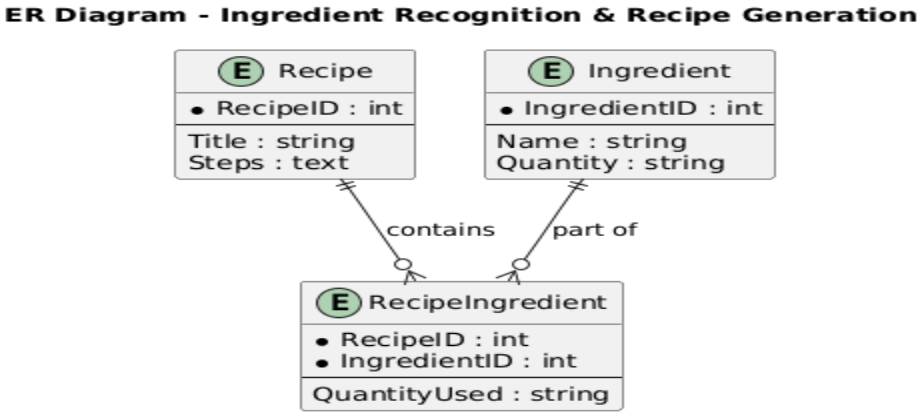

Fig 6.5.1 ER Diagram

6.6 Activity Diagram

An **activity diagram** illustrates the workflow or flow of control in a system. It's similar to a flowchart but can also show parallel or concurrent activities. It's used to model the business process or the logic of an operation. An activity diagram for our system would detail the steps involved in the entire process, from a user taking a picture to the final recipe being displayed, including decision points and parallel actions.

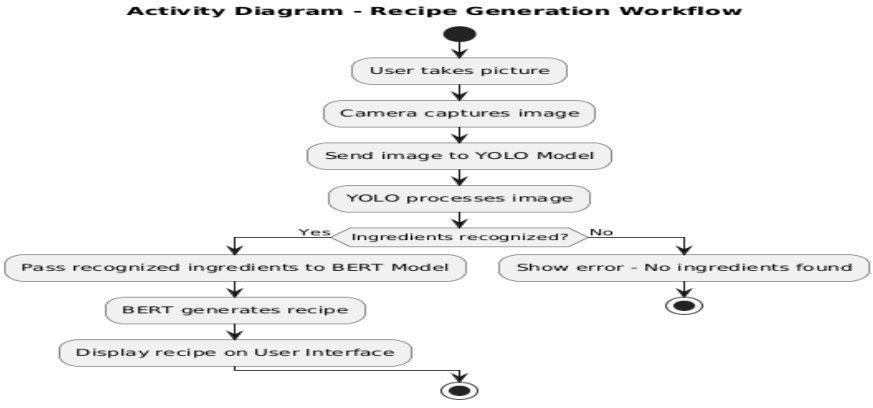

Fig 6.6.1 Activity Diagram