|  |  |
| --- | --- |
|  | Software Requirements Specification |
|  |  |
|  | Natalia Palej |

A00279259  
Software Design with Artificial Intelligence for Cloud Computing  
Year 4

Contents

[1 Introduction 2](#_Toc183177859)

[1.1 Purpose 2](#_Toc183177860)

[1.2 Product Scope 2](#_Toc183177861)

[1.3 Intended Audience 2](#_Toc183177862)

[1.4 Intended Use 2](#_Toc183177863)

[1.5 Definitions, Acronyms and Abbreviations 2](#_Toc183177864)

[2 Overall Description 3](#_Toc183177865)

[2.1 Product Perspective 3](#_Toc183177866)

[2.2 Product Functions 3](#_Toc183177867)

[2.3 Constraints 3](#_Toc183177868)

[2.4 Assumptions and Dependencies 3](#_Toc183177869)

[3 System Architecture 4](#_Toc183177870)

[4 System Features and Requirements 4](#_Toc183177871)

[4.1 Functional Requirements 4](#_Toc183177872)

[4.1.1 Overview 4](#_Toc183177873)

[4.1.2 Use Case Diagram 5](#_Toc183177874)

[4.1.3 Detailed Use Cases 5](#_Toc183177875)

[4.2 Non-Functional Requirements 6](#_Toc183177876)

[4.3 External Interface Requirements 6](#_Toc183177877)

[4.4 Design Constraints 6](#_Toc183177878)

[4.5 Acceptance Criteria 7](#_Toc183177879)

[5 Requirements Traceability Matrix 7](#_Toc183177880)

[6 Appendices 7](#_Toc183177881)

[6.1 References 7](#_Toc183177882)

# Introduction

## Purpose

This document specifies the requirements for the Recipe Vision application, a system designed to reduce food waste by identifying raw food ingredients through image recognition and generating recipes based on identified items.

## Product Scope

Recipe Vision is an AI-powered application that detects food items and their quantities using YOLOv8 and suggests recipes through Groq AI API integration. The application aims to provide a sustainable, cost-effective solution for managing food waste and household expenses.

## Intended Audience

The intended audience for this application are individuals and households who aim to reduce food waste by using up the leftover ingredients that they do not know how to combine together. The application is also targeted towards people who seek cost-effective solutions for meal preparation. Users are expected to have basic familiarity with web apps.

## Intended Use

The application is intended to:

* Identify food items and their quantities through image recognition
* Allow users to confirm or edit detected items for accurate recipe generation
* Provide recipe suggestion based on the detected items, where at least one option is healthy
* Promote sustainable consumption by helping users minimize food waste globally
* Raise awareness about food management practices that reduce environmental impact and support cost-effective meal preparations

## Definitions, Acronyms and Abbreviations

**YOLOv8**: You Only Look Once, Version 8 – Object detection model.

**Groq API**: External API for recipe generation.

**Flask:** lightweight, open-source web framework for Python, used to develop web applications

**UI**: User Interface.

**Confidence Threshold**: Minimum detection confidence level (set at 80%)

# Overall Description

## Product Perspective

Recipe Vision joins object detection and recipe generation in a simple, user-friendly application. Most existing applications are text-based, requiring users to type in the ingredients manually, which can be time-consuming. Other solutions, like Samsung’s Family Hub, are very expensive and require purchasing specific appliances, such as smart fridges. Recipe Vision offers more affordable and accessible alternative by allowing users to upload pictures of food items directly, making it easier to manage leftovers and reduce food waste.

## Product Functions

The application will:

* Accept user-uploaded images via a web-based interface
* Detect food items from the image using YOLOv8 custom model, with minimum confidence threshold of 80%
* Allow users to confirm or modify detected items
* Generate recipes through Groq API based on confirmed items
* Display recipe suggestions in user-friendly interface

## Constraints

**Hardware:** The system requires access to hardware with high GPU and memory capacity for training the YOLOv8 model

**Software Dependencies:** The system relies on the Groq API for recipe generation, which could introduce constraints such as:

* **Rate Limits:** The API may limit the number of requests allowed per second or per day, which could affect usability during peak usage times.
* **Latency:** API response times may vary, potentially delaying recipe generation.

**Input:** Clear and high-quality images are required for accurate detection

## Assumptions and Dependencies

* Users will provide clear and good quality images of common food items for detection
* Open-source datasets may not cover all required food items, so additional custom images will need to be collected
* Groq API will remain available and functional through the project duration
* **Fallback Plan:** If the Groq API becomes unavailable, the system will implement an alternative recipe generation solution, such as ChatGPT's API or another open-source recipe generation library
* Users will confirm or modify detected items to improve recipe relevance

# System Architecture

The system is designed to process user-uploaded images, detect food items and generate recipe suggestions. It achieves this by combining a web-based UI, backend for image processing, API integration and advanced AI models like YOLOv8 for object detection. The system integrates with Groq API to generate recipe suggestions based on the detected items.

A diagram of a construction project

Description automatically generated

Figure System Architecture Diagram

**User Interface (UI)** – Flask based web interface where users upload images, view detected food items and access recipe suggestions.

**Backend** – Manages communication between the UI, YOLOv8 model and Groq API, handling preprocessing and results.

**Custom YOLOv8 Model** – Detects and classifies food items in uploaded images.

**Groq AI API** – Generates recipe suggestions based on detected food items, including one healthy option and sends them to the backend.

# System Features and Requirements

## Functional Requirements

### Overview

**FR1:** The system must allow users to upload images via a Flask-based user interface (UI).

**FR2:** The system must detect food items in the uploaded images using the YOLOv8 model with a confidence threshold of 80%.

**FR3:** The system must enable users to confirm or edit detected food items through the UI.

**FR4:** The system must send the confirmed food items to the Groq API for recipe generation.

**FR5:** The system must display recipe suggestions in the UI, ensuring that at least one recipe includes a healthy option.

### Use Case Diagram

A diagram of a system

Description automatically generated

Figure Use Case Diagram

### Detailed Use Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Actor** | **Description** | **Pre-condition** | **Post-condition** |
| Upload Image | User | User uploads the image of raw food products | User opens the browser or app | Ingredients are detected and flagged for confirmation |
| Confirm Detected Items | User | User confirms or adds missing items in an input box | Detected ingredients are displayed to user | Recipes are generated and displayed |
| Generate Recipes | AI API | AI API generates recipes based on confirmed ingredients | Ingredients are confirmed by the user | Recipes are returned to the system for display |
| View Recipe Details | User | User views detailed steps and ingredients needed for recipes | Recipes are generated by AI API | Recipe details are displayed to the user |
| Save Recipe | User | User saves recipes | Recipe is displayed to the user | Recipe is saved and stored in user's device |

## Non-Functional Requirements

**NFR1 Usability -** The system shall provide an intuitive and responsive UI accessible to users with basic web application knowledge.

**NFR2 Reliability -** The system shall ensure consistent functionality, including image processing and recipe generation.

**NFR3 Performance -** The system shall process images and generate recipe suggestions in a timely manner to ensure a smooth and efficient user experience.

**NFR4 Security -** The system shall comply with GDPR regulations and uploaded images shall not be stored or shared with third parties.

**NFR5 Documentation** - The system development process shall be documented comprehensively, including final year thesis and project outcomes.

**NFR6 Presentation** - The project shall include a professional poster summarizing the problem, methods, results and conclusions.

## External Interface Requirements

**User Interface:** A web-based interface developed in Flask, which allows users to upload images, view detected food items, edit or confirm detections and display recipe suggestions.

**Hardware Interfaces:** System requires access to machine with high performance GPU for training the custom YOLOv8 model and handling intensive tasks.

**Software Interfaces:** Integration with YOLOv8 (via PyTorch) for object detection and Groq API for recipe generation.

**Communication Interfaces:** Secure API calls between the system and Groq API.

## Design Constraints

**DC1:** Must be developed using open-source tools (e.g., PyTorch, Flask).

**DC2:** Deployment limited to college-provided resources for training.

**DC3:** The system is dependent on the availability and reliability of the Groq API for recipe generation.

## Acceptance Criteria

**AC1:** The dataset must be fully annotated and validated.

**AC2:** Custom YOLOv8 model must be trained and capable of detecting food items with at least 80% confidence

**AC3:** The system must generate and display recipe suggestions, including at least one healthy option, based on confirmed food items.

**AC4:** The UI must allow user to upload images, confirm or edit detected items and view recipes without errors.

**AC5:** All core functionalities (image upload, detection, confirmation, recipe generation) must pass functional and non-functional testing.

**AC6**: Comprehensive documentation, including final year thesis and project outcomes must be completed.

**AC7**: Professional project poster summarizing the problem, methods, results and conclusions must be designed.

# Requirements Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Related Functionality** | **Linked AC** |
| FR1 | Image Upload | Flask-based user interface | AC1, AC4, AC5 |
| FR2 | Object Detection | Custom-trained YOLOv8 model | AC2, AC5 |
| FR3 | Manual Confirmation | Component for manual adjustment | AC4, AC5 |
| FR4 | Recipe Generation | Backend integration with Groq API | AC3, AC5 |
| FR5 | Recipe Display | Frontend display logic | AC3, AC4, AC5 |
| NFR1 | Usability | Frontend usability design | AC4 |
| NFR2 | Reliability | Robust backend and model integration | AC3, AC5 |
| NFR3 | Performance | Backend and efficient API calls | AC3, AC5 |
| NFR4 | Security | Secure backend design | AC4 |
| NFR5 | Documentation | Final year thesis and outcomes | AC6 |
| NFR6 | Presentation | Professional project poster | AC7 |

# Appendices

## References

1. Lecture Slides on *Proposal and SRS*​, *Project Planning* and *Risk Management*. Available at: <https://moodle.midlands.tus.ie/course/view.php?id=18762>
2. Ultralytics YOLOv8 Documentation. Accessed October 12, 2024. Available at: <https://docs.ultralytics.com/>
3. Groq AI API Documentation, 2024. Accessed October 12, 2024. Available at: <https://console.groq.com/docs/overview>
4. Flask Official Documentation. Accessed October 12, 2024. Available at: [https://github.com/ultralytics/ultralytics/blob/main/docs/en/models/YOLOv8.md](https://github.com/ultralytics/ultralytics/blob/main/docs/en/models/yolov8.md)