Software Requirements Specification (SRS)
for
VOC- Sensor Based Food Spoilage Detection System

Version 2.0

Group-1

Submitted to:	Team members:	
Coordinator:	Alan Vincent	23RR040
Dr. Neenu Daniel	Anase Benny	23RR202
Guide:	Janpaul Benny	23RR015
Mr. Andrews Jose	Jose James	23RR041

Contents

Revision History

Name	Date	Reason for Change	Version
V1	09-01-26	Initial draft	1.0
V2	13-01-26	Correction of diagrams	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements for the **VOC-Sensor Based Food Spoilage Detection System**. This system aims to detect food spoilage using sensor-based volatile organic compound (VOC) data and food images, processed through cloud-based machine learning models. The results are presented to users through an **Android-based mobile application**. This document serves as a reference for development, evaluation, and future enhancement of the system.

1.2 Document Conventions

This document follows IEEE Software Requirements Specification guidelines. The following conventions and abbreviations are used:

- VOC – Volatile Organic Compounds.
- ML – Machine Learning.
- RF – Random Forest.
- CNN – Convolutional Neural Network.
- UI – User Interface.
- DB – Database.
- GPIO – General Purpose Input/Output.
- CSI – Camera Serial Interface.

1.3 Intended Audience and Reading Suggestions

This document is intended for academic evaluators, faculty mentors, and student developers involved in the project. Readers are encouraged to review the overall system description before examining system features and nonfunctional requirements.

1.4 Product Scope

The VOC-Sensor Based Food Spoilage Detection System provides an intelligent and cost-effective solution for identifying food freshness. It is intended for household users, small food storage facilities, warehouse workers, and food inspection personnel as a **preliminary screening tool**, not as a replacement for laboratory-grade testing.

1.5 References

- IEEE Std 830-1998 – Software Requirements Specification.
- IEEE Journals on Electronic Nose Systems for Food Quality Assessment.
- Kaggle and UCI Machine Learning Repository – Gas Sensor Datasets.

2. Overall Description

2.1 Product Perspective

This system is a distributed IoT and cloud-based solution consisting of:

- An edge device (Raspberry Pi Zero 2) connected to sensors and a camera.
- A cloud backend hosting machine learning models and database services.
- An Android mobile application for user interaction and result visualization.

The system processes sensor and image data in the cloud and returns spoilage classification results to the mobile application.

2.2 Product Functions

The primary function of the VOC-Sensor Based Food Spoilage Detection System is to assist users in determining the freshness of food items using sensor data and image-based analysis. The system is designed to collect input data from users and hardware components, process the data using cloud-based machine learning models, and present meaningful results through a mobile application.

Inputs collected by the system:

- Food image captured using a camera module connected to the edge device.
- VOC sensor readings indicating gas concentration levels emitted by food items.
- User-initiated inspection request through the mobile application.

Processing performed by the system:

- Preprocessing of image and sensor data.
- Food type identification using a CNN-based machine learning model.
- Spoilage level classification using a Random Forest classifier.

Outputs provided to the user:

- Identified food category.
- Spoilage or freshness level (e.g., fresh, moderate, spoiled).
- Inspection timestamp and summary.
- Historical inspection records accessible through the mobile application.
- Detailed summary and report file generation.

2.3 User Classes and Characteristics

2.3.1

General User

Includes household consumers, warehouse workers, and inspectors. Users interact through a mobile application and require minimal technical expertise.

2.3.2

Administrator

Administrators manage cloud services, databases, and machine learning model updates. This role is limited to authorized personnel.

2.4 Operating Environment

2.4.1 Hardware

Uses Raspberry Pi Zero 2 as the main controller device with MQ-series gas sensors, a camera module to provide input for various tasks performed by the system.

2.4.2 Backend Software

The backend application is handled mostly on cloud server that handles all the ML based processing and result generation. Also deals with history and logs of users.

2.4.3 Mobile Platform

The project model is set to work on a mobile application set to work on an android device. The user is able have an individual profile with access to all logs of all the scans performed.

2.5 Design and Implementation Constraints

2.5.1 Network and Connectivity Constraints

Continuous internet connectivity is required for cloud processing.

2.5.2 Sensor Constraints

Sensor accuracy may vary based on environmental conditions.

2.5.3 Platform Limitation

Mobile application features are limited to Android platform for this mini-project.

2.6 User Documentation

2.6.1 Application

The application provides user documentation in the form of a user manual and tutorials to help new users understand the system. Step-by-step instructions are provided for using the dashboard, along with clear explanations of features. This documentation serves as both a learning guide and a reference for users

2.6.2 Device

The device is accompanied with usage manual, safety guidelines, and troubleshooting instructions. It details the limitations and the extent of usage of the analysis generated.

2.7 Assumptions and Dependencies

2.7.1 Sensors

Sensors are properly calibrated before operation.

2.7.2 Database

Cloud services and APIs remain operational.

2.7.3 Connectivity

Users have access to Android smartphones with internet connectivity.

---

3. External Interface Requirements

3.1 User Interfaces

The system provides an Android-based mobile interface with the following characteristics:

- Simple and intuitive layout.
- Display of food type and spoilage level.
- Visual indicators for freshness classification.
- Access to previous inspection records.

3.2 Hardware Interfaces

- GPIO interface for VOC sensors.
- CSI/USB interface for camera module.
- Wi-Fi module for internet communication.

3.3 Software Interfaces

- Android Mobile Application – User interaction.
- Cloud Database (e.g., Firebase) – Storage of inspection data.
- Cloud ML Services – Food identification and spoilage classification.

3.4 Communications Interfaces

- Data transmission between the mobile application, edge device, and cloud backend shall be performed over HTTP, with support for HTTPS for secure communication.

---

4. System Features

Now with all of the functions and benefits, the following are the key features we are planning to implement on our food spoilage detection system.

4.1 Image-Based Food Identification

Upon an inspection request, the edge device (Raspberry Pi Zero 2) shall capture at least one image of the food item using the connected camera module. The system shall analyze captured food images using a CNN-based machine learning model to identify the type of food and display it to the user.

4.2 Spoilage Classification

The system shall allow the user to view a list of previous inspection records in the mobile application. VOC sensor data shall be processed using a Random Forest classifier to determine the level of food spoilage.

4.3 Result Visualization

The mobile application shall present the inspection result on a dedicated result screen containing food type, spoilage level, and inspection timestamp. The system shall use visual indicators (e.g., color codes or icons) to represent freshness categories.

4.4 Record Storage and Traceability

The system shall store each inspection result in the cloud database, including at least: food type, spoilage level, timestamp, and device or user identifier. The system shall allow the user to view a list of previous inspection records in the mobile application, ordered newest to oldest, and further details upon request.

---

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Mobile application screens shall load within 3 seconds.

5.2 Security Requirements

- Secure communication using HTTPS.
- Restricted access to cloud databases.

5.3 Software Quality Attributes

- **Reliability:** Consistent classification results.
- **Usability:** Simple mobile interface.
- **Maintainability:** Modular backend design.
- **Scalability:** Ability to extend to additional food types in the future.

6. Other Requirements

6.1 Privacy Policy

The system shall follow basic data privacy principles while handling user data and inspection records. The mobile application shall clearly inform users about what data is collected, including food inspection results and timestamps, and how this data is stored and used. Personal user information shall not be shared with third parties.

6.2 Collection of User and System Data

The system collects limited user-related data such as inspection requests and result history. Sensor data and image data are collected solely for the purpose of spoilage detection and system improvement. Data collection practices shall comply with applicable academic and institutional guidelines.

6.3 Legal and Ethical Considerations

The system is intended as a preliminary food freshness assessment tool and shall not be used as a certified food safety or medical diagnostic system. Ethical considerations include transparent communication of system limitations and responsible use of machine learning outputs.

6.4 Copyrighted Content

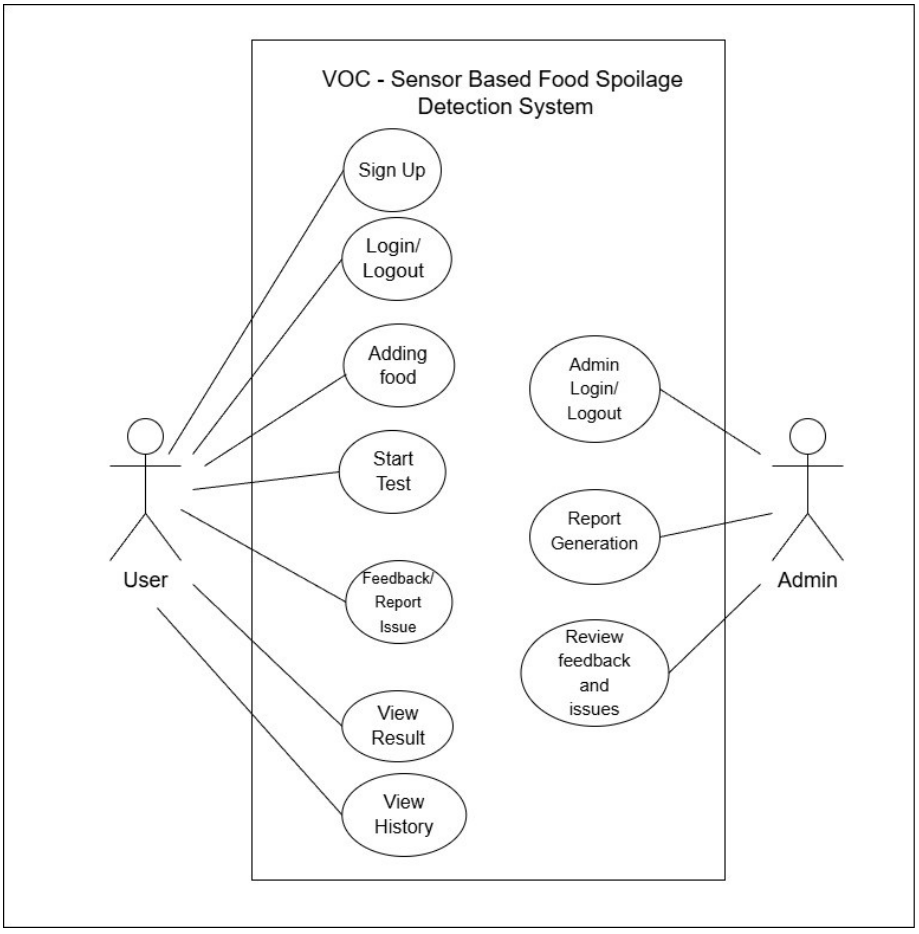
All software components, documentation, datasets, and application interfaces developed as part of this project shall be protected from unauthorized reproduction. Appropriate copyright notices will be included where applicable.

Appendix A: Glossary

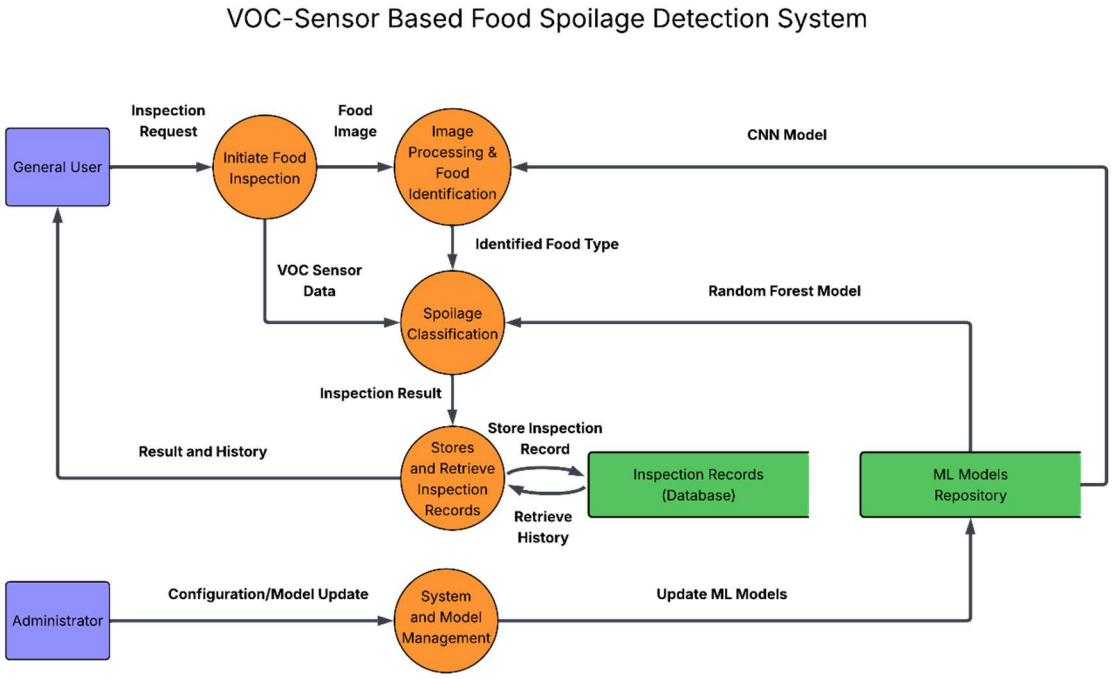
- **Android Application**: The mobile client running on Android devices through which users initiate inspections and view results.
- **Cloud Backend**: The remote server environment that hosts machine learning models, APIs, and the database used for processing and storing inspection data.
- **Cloud Database**: A managed online database service (e.g., Firebase, Supabase) used to store inspection records and related data.
- **CNN:** Convolutional Neural Network for image classification.
- **CSI (Camera Serial Interface)**: A dedicated interface on the Raspberry Pi used to connect camera modules for image capture.
- **Edge Device**: The Raspberry Pi–based hardware unit that connects to sensors and camera, collects data, and sends it to the cloud backend.

- **Food Category**: The class label assigned by the CNN model to identify the type of food (e.g., meat, fish, fruit, vegetable).
- **GPIO (General Purpose Input/Output)**: Programmable pins on the Raspberry Pi Zero 2 used to interface with external hardware such as VOC gas sensors.
- **Inspection Record**: A stored entry representing a single food inspection, including food type, spoilage level, timestamp, and related data.
- **MQ-Series Sensors**: A family of gas sensors used to detect VOC levels emitted by food during spoilage.
- **Random Forest:** An ensemble-based machine learning classifier.
- **Raspberry Pi Zero 2**: A compact single-board computer used as the edge device to interface with sensors and camera.
- **Spoilage Level**: The freshness status assigned by the Random Forest model based on sensor readings (e.g., fresh, moderate, spoiled).
- **VOC (Volatile Organic Compounds):** Gases emitted during food spoilage.

Appendix B: Analysis Models



- Use Case Diagram – Food Spoilage Detection System.



- Data Flow Diagram – Sensor and Image Processing Pipeline.

Appendix C: To Be Determined List

1. Final sensor selection and calibration method.
2. Cloud deployment configuration.
3. Spoilage level threshold definitions.
4. Communication protocol for edge device.