Software Requirements Specification

For

Botani Scan

Version 1.1

Prepared by
Izma Shafqat (SE-21003)
Aqsa Zaib (SE-21013)
Inshara Iqbal (SE-21018)
Mahnoor Iqbal (SE-21027)
Sahil Krishna (SE-21046)

Internal Advisor: Ms. Sana Fatima

17/11/2023

Table Of Contents

1 INTRODUCTION

1.1 PURPOSE OF REQUIREMENT DOCUMENT

This Software Requirements Specification provides a complete description of all the functions and specifications of the BotaniScan. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate.

The purpose of the BotaniScan software is to provide a powerful and user-friendly tool for the real-time identification of grape,apple and tomato plants by their leaves,the detection of leaf spot disease and the estimation of leaf water levels in the selected plants.This software aims to cater to the needs of both hobbyist gardeners and professional botanists, offering valuable insights about their plants through the utilization of computer vision and machine learning techniques.

1.2 PROJECT SCOPE

1.2.1 Description

BotaniScan is an AI based application that can identify grape,apple and tomato plants by their leaves, detect leaf spot disease, and estimate the leaf water level in these plants using computer vision and machine learning techniques. This project will serve as a valuable tool for both hobbyist gardeners and professional botanists by providing real-time information about these plants.

1.2.2 Benefits

BotaniScan, an AI-based application, offers benefits for both hobbyist gardeners and professional botanists. By using the power of computer vision and machine learning, the application excels in identifying grape, apple, and tomato plants through their leaves. One of the key advantages of BotaniScan lies in its ability to detect leaf spot disease in these plants. By offering real-time information about the presence of leaf spot disease, the application provides tips and precautionary measures to overcome this disease. Furthermore, the application's capability to estimate leaf water levels adds another layer of utility. Additionally, BotaniScan's commitment to user guide and comprehensive documentation ensures easy adoption and understanding for plant care and research.

1.2.3 Corporate Goals

The corporate goals of BotaniScan encompass revolutionizing plant care through advanced technology. Our primary objective is to develop and refine an AI-based application capable of identifying grape, apple, and tomato plants by analyzing their leaves. We aim to empower both hobbyist gardeners and professional botanists by providing a user-friendly tool that not only recognizes these plants but also detects leaf spot disease in these plants and estimates the leaf water level.Through these innovative features, BotaniScan can promote sustainable gardening practices. Ultimately, BotaniScan strives to be the go-to solution for plant enthusiasts and experts alike, fostering a deeper understanding and more effective care of grape, apple, and tomato plants.

1.3 DEFINITIONS, ACRONYMS & ABBREVIATIONS

Kaggle

Kaggle is a well-known online platform for data science competitions, machine learning, and data analytics. It provides datasets, tools, and a community of data scientists and machine learning practitioners to collaborate, compete, and share insights. Kaggle hosts a wide range of data-related challenges, from predictive modeling to computer vision tasks.

OpenCV

OpenCV stands for Open Source Computer Vision.

OpenCV is a library of programming functions mainly for real-time computer vision Originally developed by Intel.It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

YOLO

YOLO stands for You Only Look Once.

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection.

TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

CNN

CNN stands for Convolutional Neural Network.

Convolutional neural network (CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. Vanishing gradients and exploding gradients, seen during backpropagation in earlier neural networks, are prevented by using regularized weights over fewer connections.

Google Colab

Google Colab is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Google Cloud

Google Cloud, offered by Google, is a suite of cloud computing services that provides a series of modular cloud services including computing, data storage, data analytics and machine learning, alongside a set of management tools.[2] It runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, and Google Docs

React JS

ReactJS is a free and open-source front-end JavaScript library[3][4] for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.[2] It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming

RESTful API

RESTful API stands for Representational State Transferful Application Programming Interface

RESTful API is an application programming interface (API or web API) that confirms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

1.4 REFERENCES

- IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
- https://www.scribd.com/doc/9138468/Software-Requirement-Specification-Srs-Midtems
- http://www.processimpact.com/process_assets/srs_template.doc
- http://www.wikipedia.com
- https://opencv.org

- https://v7labs.com
- https://redhat.com
- https://www.mdpi.com/2077-0472/12/10/1542#:~:text=Grape%20isariopsis%20leaf%20spot%20is,be%20detected%20early%20%5B12%5D.
- https://www.missouribotanicalgarden.org

1.5 OVERVIEW

The rest of this SRS is organized as follows:

- Section 2 and 3 gives an overall description of the software. It gives what level of proficiency is expected of the user, some general constraints while making the software and some assumptions and dependencies that are assumed.
- Section 4 contains most important features presented with detailed description, and requirements. It gives specific requirements which the software is expected to deliver. Functional requirements are given in this section. This section is written primarily for the developers and describes in technical terms the details of the functionality of the product and about safety and performance.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2 GENERAL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

BotaniScan, an AI-based application, enhances plant management for hobbyist gardeners and professional botanists. Through intuitive user interfaces and computer vision techniques, it identifies grape,tomato and apple plants, detect leaf spot disease, and estimates leaf water levels. The system interfaces with the device's camera, external plant datasets, and machine learning models. It operates in the context of plant and garden management, assuming access to a reliable compatibility with various devices and operating systems. By providing real-time information about the mentioned plants, BotaniScan serves as a valuable tool for plant enthusiasts and experts, contributing to more informed and efficient plant care.

2.2 PRODUCT FUNCTION

BotaniScan provides the following functionalities to its users (Hobbyists gardeners, farmers, botanists, and agricultural experts)

2.2.1 Plant Identification:

Users will be using BotaniScan to easily identify grape, apple, and tomato plant's leaves in their gardens, fostering a better understanding of the plants they cultivate.

2.2.2 Disease Detection:

Users will be able to get the result of detected leaf spot disease and tips to maintain the health of their plants.

2.2.3 Water Level Estimation:

Users will be able to get an estimation of leaf water levels that contribute to sustainable gardening.

2.3 USER CLASSES AND CHARACTERISTICS

We will be having different classes for following users.

2.3.1 For Individuals (Hobbyist Gardeners)

BotaniScan is designed for hobbyist gardeners, regardless of age or gender, with varying levels of expertise. These users are motivated by their love for gardening, landscaping, and plants. Their specific needs include grape,tomato and apple plants identification and leaf spot disease detection to maintain their plants health. Usage frequency ranges from occasional to regular.

2.3.2 For Professionals (Botanists and Agricultural Experts)

BotaniScan caters to professionals in botany and agriculture, facilitating research and precise leaf spot disease management. These experts, with advanced degrees, use the app regularly.

2.4 GENERAL CONSTRAINTS

2.4.1 Hardware Limitations

The performance and accuracy of the BotaniScan application may be influenced by the quality of the camera on the user's device. Higher-resolution cameras are likely to provide better leaf image input for analysis.

2.4.2 Machine Learning Model Size

The size of the dataset used for leaf identification and disease detection may affect the response time and performance of our application.

2.5 ASSUMPTIONS AND DEPENDENCIES

2.5.1 Assumptions

- **User-Provided Images:** The project assumes that users will provide clear and well-captured images of plant leaves. The accuracy of identification and disease detection is highly dependent on the quality of the input images.

- **Legal Considerations:** The project assumes that there are no legal issues related to using images of plants(dataset), and there are no copyright violations regarding the dataset.

2.5.2 Dependencies

- **Availability of Quality Datasets:** The project relies on comprehensive datasets of plant images and diseases. A dependency is that such datasets are available, and they need to be regularly updated to ensure the system's accuracy.
- **Machine Learning Models:** The successful implementation of machine learning models for plant identification and disease detection depends on the availability of suitable algorithms and tools. Dependencies include the selection of the right machine-learning frameworks and libraries.
- **Server Infrastructure:** To process and analyze images and deliver real-time results, the application depends on server infrastructure with sufficient computational power and storage capacity.

3 SPECIFIC REQUIREMENT

3.1 FUNCTIONAL REQUIREMENT

- **Plant Identification:**
  - The system should analyze images captured by users.
  - The application should identify grape, apple, and tomato plants based on the analysis of their leaves.
- **Disease Detection:**
  - The application should be capable of detecting leaf spot disease in grape, apple, and tomato plants through visual analysis of their leaves.
- **Leaf Water Level Estimation:**
  - BotaniScan should estimate the leaf water level in grape, apple, and tomato plants, offering insights into the hydration status of the plants.
- **Real-Time Information Delivery:**
  - BotaniScan should deliver real-time information about identified plant whether it is apple, grape or tomato plant leaves.
  - The System should deliver real-time information about leaf spot disease whether it exists in plant leaves or the leaf is free from this disease.
  - Thy system should deliver real-time information about estimated leaf water levels to users.
- **Error Handling**
  - The system must detect and handle unclear or distorted images.
  - In such cases, a clear error message should be generated to prompt users to capture a clearer image.
- **User-Friendly Interface:**
  - The application must have an intuitive and user-friendly interface accessible to both hobbyist gardeners and professional botanists.
  - Based on disease detection and water level estimation, the system should provide a blog that guides user about disease, it's cure, precautionary measures and importance and need of water for plants

3.2 NON FUNCTIONAL REQUIREMENT

3.2.1 Product Requirements

- **Accuracy:**
  - The system should provide accurate identification of grape, apple and tomato plants and leaf spot disease with a better level of confidence.

- **Response Time:**
  - The application should have low latency in processing user inputs and returning results, ensuring a smooth and responsive user experience.
- **Scalability:**
  - The system should be able to handle a large number of users and images simultaneously.
- **Usability:**
  - The user interface should be intuitive, easy to navigate, user-friendly and should provide responsive user interface.
- **Reliability:**
  - The system should be robust and reliable, with minimal downtime or technical issues.
- **Compatibility:**
  - The web application should be compatible with various devices, including smartphones, tablets, and desktop computers, and across different operating systems.
- **Documentation:**
  - Comprehensive documentation for users and developers should be available, explaining how to use the application and its underlying technologies.

3.2.2 Domain-Specific Requirements

- **Botanical Dataset:** Provide an extensive dataset of grape,tomato and apple plant leaves , leaf spot disease and watering level information.
- **Botanical Knowledge Base:** Incorporate botanical expertise to improve plant identification accuracy.

3.2.3 External Requirements

- **Ethical Considerations:** The system should be designed and operated ethically, avoiding any harmful consequences related to plant diagnosis. Ethical considerations also include transparent data usage and user consent.

3.3 EXTERNAL INTERFACE REQUIREMENTS

3.3.1 Hardware Interfaces

This section describes the hardware components or devices with which the Plant Disease Detection system needs to interact or be compatible. It includes details about the system's requirements for mobile phones, laptop and desktop computer with minimum 10 mega pixel camera.

3.3.2 Software Interfaces

- **Kaggle** will serve as a resource for accessing plant-related datasets and machine learning competitions. It will help in training and improving the accuracy of the system's plant identification and disease detection models by providing valuable data and benchmarking opportunities.
- **OpenCV** will be utilized to process and analyze plant images. It will enable the system to perform tasks like leaf recognition, disease detection, and image manipulation to enhance the accuracy of plant identification and health assessment.
- **YOLO** will be employed for efficient and real-time object detection in plant images. It will enhance the system's ability to quickly and accurately locate and identify plant leaves and diseases within images and video streams.
- **TensorFlow** will be the core technology for building and training machine learning models, including convolutional neural networks (CNNs), to improve the accuracy of plant identification and disease detection.
- **CNN** built using TensorFlow, will be used for deep learning tasks, such as recognizing plant leaves and detecting diseases, making the system proficient in image-based plant analysis.
- **Google Colab** will provide access to free GPU resources, allowing the system to train machine learning models more efficiently and handle large datasets for plant identification and disease detection.
- **Google Cloud** services can be used for deploying the system, handling scalability, storing data, and running the application in a reliable and cost-effective manner.
- **React JS** will be employed to build a user-friendly and interactive web interface, enabling users to easily input images and receive real-time information about plant identification and disease status.
- **Flask** will serve as the backend framework for the system, handling API requests and responses. It will ensure smooth communication between the frontend and the machine learning models.
- **Python** will be the primary programming language for the system, enabling the integration of various components and libraries for plant identification and disease detection.
- **RESTful API** will be implemented to facilitate communication between the frontend and backend of the system, allowing users to interact with the application and receive plant-related information.

4 INDEX