Institute of Technology University of Moratuwa

Cassava Leaf Disease Detection
Software Requirements Specification

Version 1.0

Student Details:

J.A.C.P Jayakodi 21IT0479

H.R. Pramod Dilshan 21IT0463

D.T.Thennakoon 21IT0535

K.H.R.Dilshan 21IT0464

ARM Jesaan 21IT0482

Supervised by:

Mis. M.S. Madubashini

Division of Information Technology

Institute of Technology University of Moratuwa

2024 April

1. Contents

1 Introduction

1.1 Purpose

The purpose of this project is to develop a mobile app using image recognition and machine learning to help farmers accurately detect and manage common cassava diseases. By classifying cassava leaf images, the app enables real-time disease detection, encourages collaboration among users, and promotes widespread adoption with its intuitive interface. Ultimately, it seeks to reduce crop losses, improve food security, and empower farming communities.

1.2 Product Scope

1.2.1 Aim and Objectives

The primary focus of your research project is usually expressed in terms of aims and objectives.

Aim

The creation of a mobile app for farmers to detect and diagnose common cassava diseases accurately represents a breakthrough in agricultural technology. By leveraging image recognition and machine learning, the app provides farmers with an easy-to-use tool to identify symptoms and access guidance on disease management strategies. This innovation has the potential to significantly reduce crop losses, improve food security, and empower farming communities with timely information and interventions.

Objective

- Develop a robust machine learning model capable of accurately classifying cassava leaf images into distinct disease categories, including Cassava Brown Streak Disease (CBSD), Cassava Green Mite (CGM), Cassava Mosaic Disease (CMD), Cassava Bacterial Blight (CBB), and healthy leaves.

- Integrate the developed model into a mobile application to enable real-time detection of cassava diseases, facilitating timely intervention and management practices.

- Enable seamless image and prediction sharing functionality within the mobile app, fostering collaboration among users and facilitating knowledge exchange to enhance disease management efforts.

- Design an intuitive user interface within the mobile app, ensuring ease of use for farmers by simplifying the process of capturing images of cassava leaves and viewing disease predictions, thereby promoting widespread adoption and usability of the application in farming communities.

1.2.2 Project Boundary

Adapting the requirements for a cassava leaf disease detection mobile application would involve considering the specifics of the new domain while maintaining the core principles outlined in the original requirements. Let's break down how these requirements could be translated:

Environment Boundaries:

1. **Accessibility:** The mobile application should be accessible on both Android and iOS devices.

2. **Compatibility:** The app should be compatible with common mobile operating systems like Android and iOS.

3. **Browser Compatibility**: Irrelevant since it's a mobile application.

4. **Software Dependencies**: Specific software dependencies or libraries required for AI model integration should be listed. For example, TensorFlow Lite or Core ML for on-device machine learning inference.

5.**Additional Software for Image Processing or Training**: Specify any additional software or frameworks required for image processing or model training, such as OpenCV or TensorFlow for image manipulation and model training.

6. **Image Upload and Storage**: The app should support the upload and storage of cassava leaf images, preferably in common formats like JPEG.

7. **Internet Connectivity**: The app should operate in an environment with stable internet connectivity for uploading images and retrieving disease identification results. Offline functionality should also be considered.

8. **Accessibility:** The app should be accessible over the internet or a local network, depending on the deployment scenario.

9. **Processing Time**: The AI model should process images within a reasonable time frame suitable for mobile devices.

10. **Limited Diseases**: The identification will be limited to selected cassava leaf diseases. These diseases should be chosen based on relevance to cassava cultivation in the target region.

11. **Simple User Interface**: The user interface of the mobile application should be simple and intuitive, considering potential users' limited technological knowledge.

13. **Regional Focus**: Focus on cassava leaf diseases relevant to the local climate and conditions where cassava is cultivated.

14. **Language**: The language of the user interface should be English, considering its global accessibility.

2 Overall Description

2.1 Product Perspective

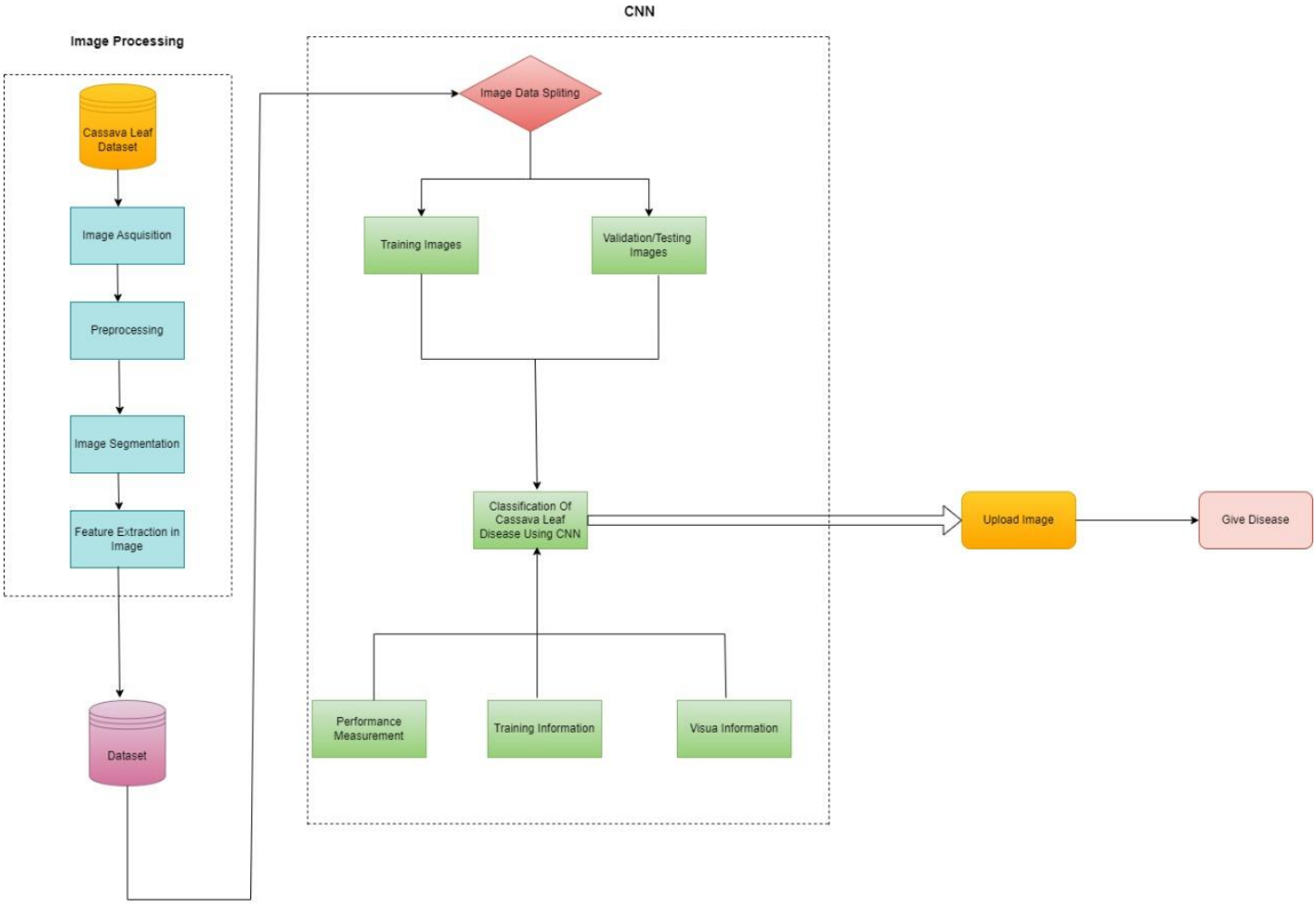

Figure 2.1 High-level Diagram

2.2 User Classes and Characteristics

Farmer/Officer

A Farmer or officer is granted access to a system where an image of a cassava leaf can be uploaded. The purpose of this functionality is to identify any diseases affecting the cassava leaves. After the image is uploaded, the system employs image recognition and analysis techniques to recognize the specific cassava leaf disease. Subsequently, the system presents the name of the identified. This initiative aims to support growers and officials in promptly and accurately detecting cassava leaf diseases.

System Administrator

The system administrator plays a critical role in maintaining and managing the overall functioning of the system. Their responsibilities include updating the dataset with new training data, which involves incorporating new information to improve the system's performance and accuracy in detecting Cassava leaf diseases. Additionally, the system administrator is responsible for host management, ensuring that system infrastructure and servers are properly maintained and operating optimally. Overall, the role of the system administrator is essential to keeping the system up-to-date, secure, and efficient in detecting and addressing issues related to Cassava leaf diseases.

2.3 Operating Environments

The mobile application front-end is crafted using Flutter, facilitating seamless operation across various mobile devices and platforms. Compatible with both iOS and Android, the application ensures a consistent user experience across different smartphones and tablets. Leveraging both Firebase cloud infrastructure and MongoDB, the application's back-end seamlessly integrates with its front-end, enabling efficient data storage, retrieval, and real-time updates. Firebase handles real-time database synchronization and user authentication, while MongoDB provides flexibility and scalability for storing and managing complex data structures. With a focus on responsive design and optimized performance, the mobile application delivers a smooth and intuitive user interface, empowering users to access essential features and functionalities on the go.

2.4 Design and Implementation Constraints

Hardware requirements:

- **Processor (CPU/GPU):**
  A powerful processor is important for handling the tasks of recognizing images of cassava leaves. Graphics Processing Units (GPUs) are often used because they can process lots of information at once.
- **Memory (RAM):**
  Enough memory, like 6GB to 8GB, is needed to handle the data when recognizing cassava leaves. More complex tasks might need even more.
- **Storage:**
  We need storage space on the device to keep all the cassava leaf pictures and data. If we use cloud storage, we can store even more data and results online.
- **Networking:**
  A good internet connection helps when using cloud services to recognize cassava leaves. Faster internet means data moves quicker.
- **Cloud Infrastructure:**
  We can use services from places like Amazon, Google, or Microsoft to help recognize cassava leaves. They offer different setups for different needs.
- **Camera Requirements for Cassava Leaf Recognition:**
  Better cameras capture clearer pictures of cassava leaves. It's good to have a camera with a high resolution, good sensor type like CMOS, and a quality lens. Lighting and environment also affect how well the camera works. A macro camera, like those on some smartphones, can capture close-up details better, which helps in recognizing cassava leaves more accurately.

2.5 Assumptions and Dependencies

- **Training Data Set:**
  We use free datasets from the internet that contain information about diseases that affect cassava leaves.
- **Using Existing AI Models:**We rely on free AI models available online that are known for being accurate in identifying issues with cassava leaves.

3 External Interface Requirements

3.1 User Interfaces

3.1.1 Home page

Figure 3.1.1

Description:

This is the start page of this mobile application .click “Get Start “ button user can start this application.

3.1.2 Sign-in /Sign-up page

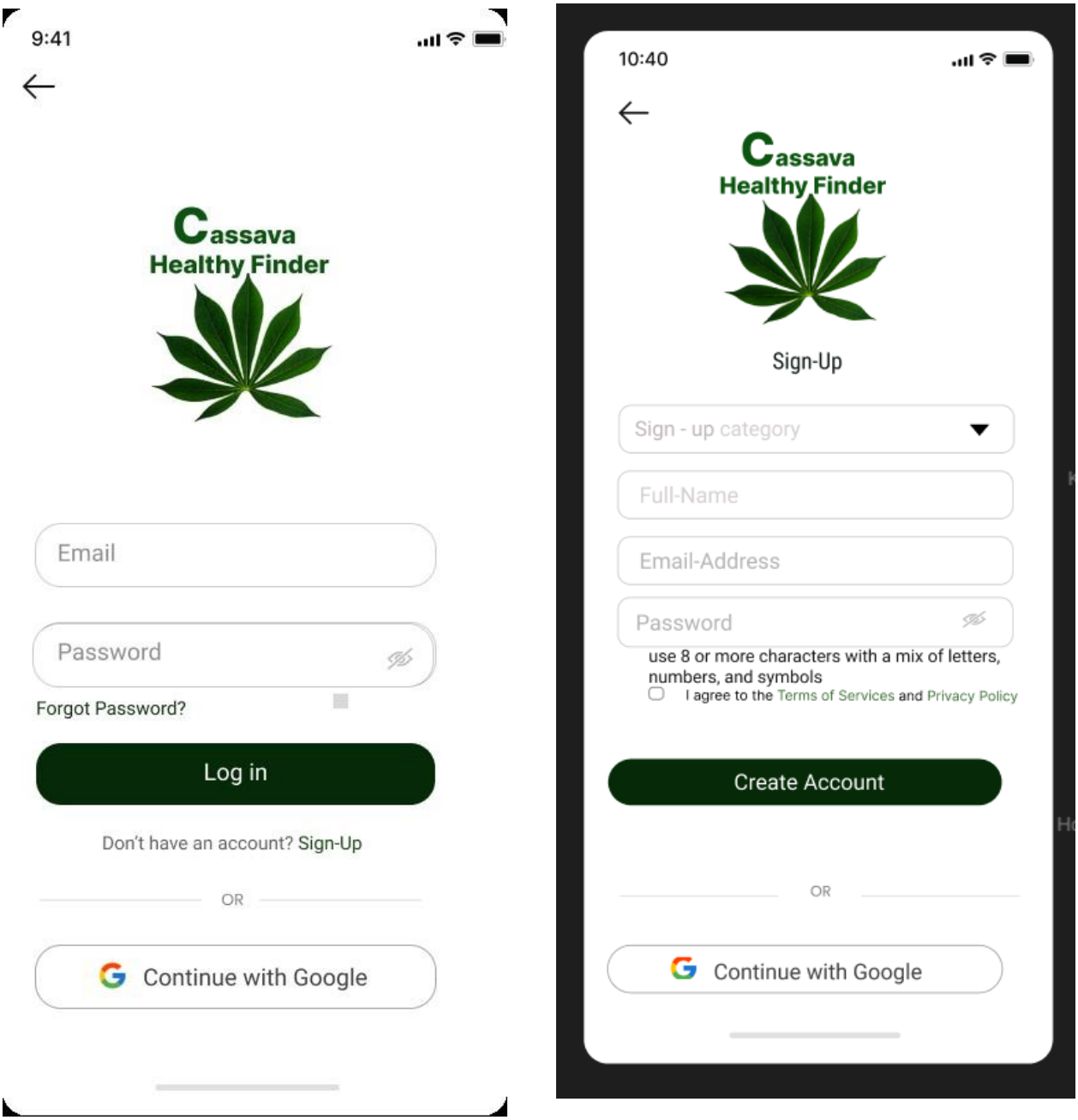

Figure 3.1.2

Description:

This page contains the Log-in .if user have account then, user can fill Email and password after user can log in to app .if user haven't account user should register to app using Sing-up button.

Then, user can select the category (Farmer/Officer), input full name, email and password user can register to app.

3.1.3 Farmer Profile

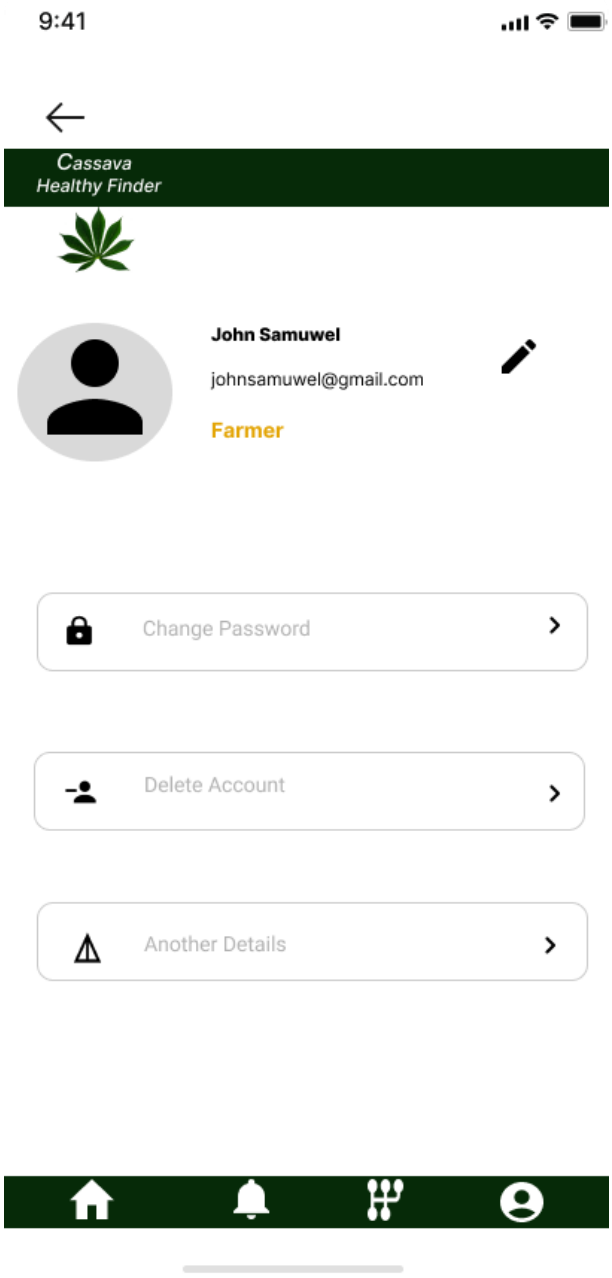

Figure 3.1.3

Description:

This page contains the one of user profile type. It is the Farmer profile interface.

The user can change password, delete account using this.

3.1.4 Agriculture Officer Profile

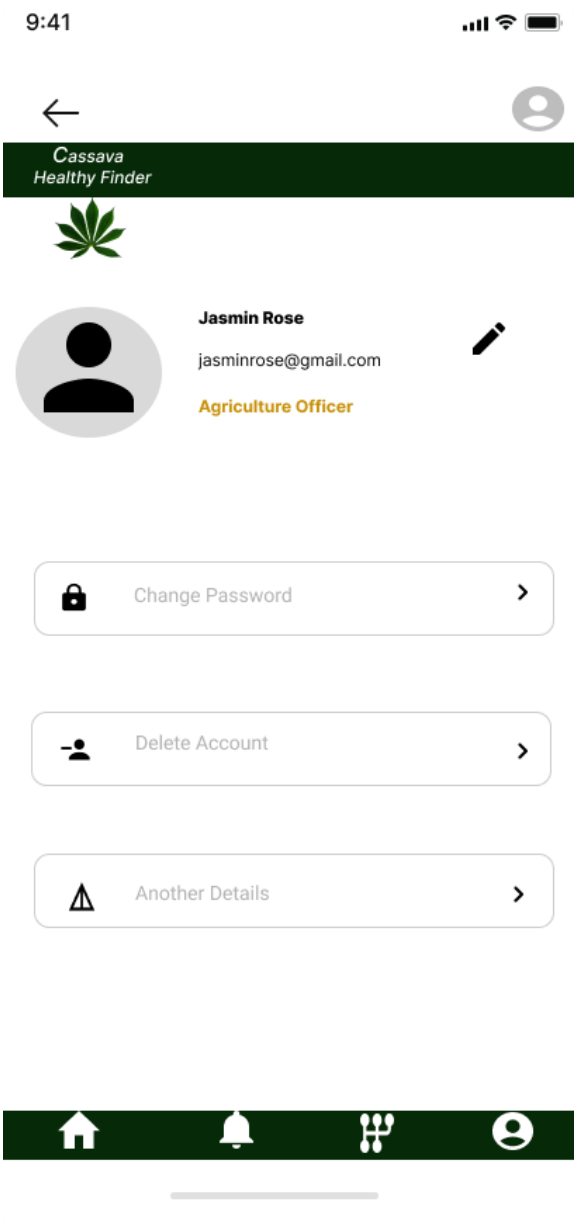

Figure 3.1.4

Description:

This page contains the one of user profile type. It is the Agriculture officer profile interface.

The user can change password, delete account using this.

3.1.5 Detection page

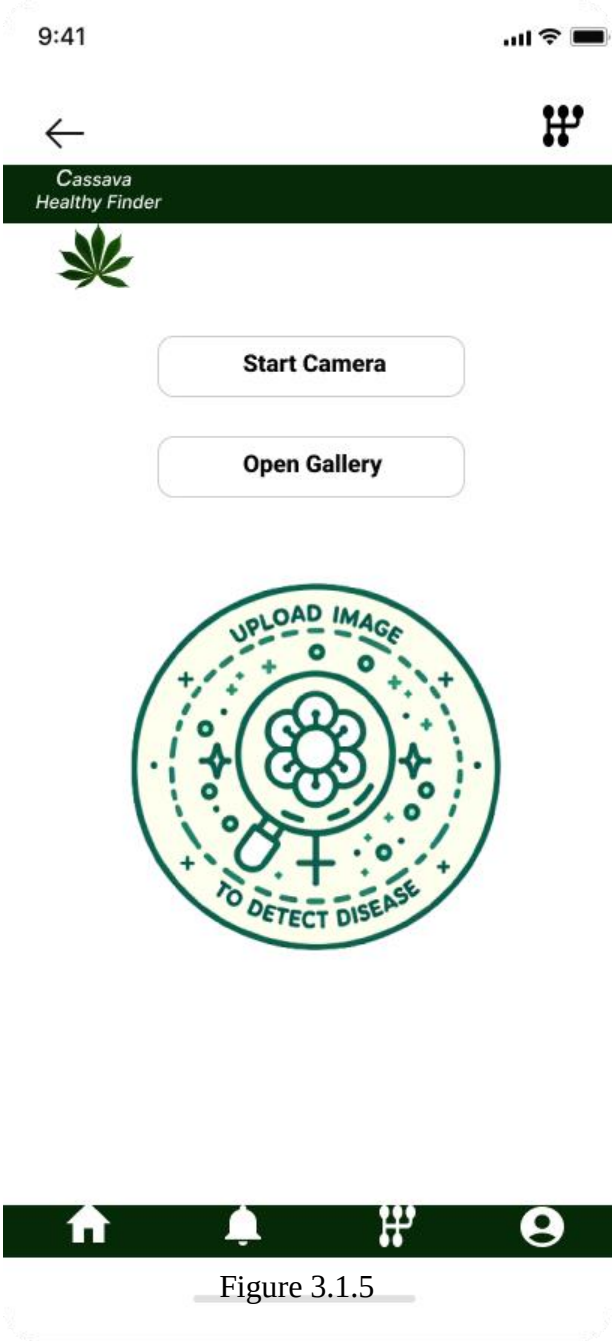

Figure 3.1.5

Description:

This page has only an Upload Image button. When the user clicks on the Upload Image button (Start Camera/Open Gallery) the browser will display a prompt message to the user by asking access to their local storage of the Mobile Phone.

3.1.6 Detection Result page

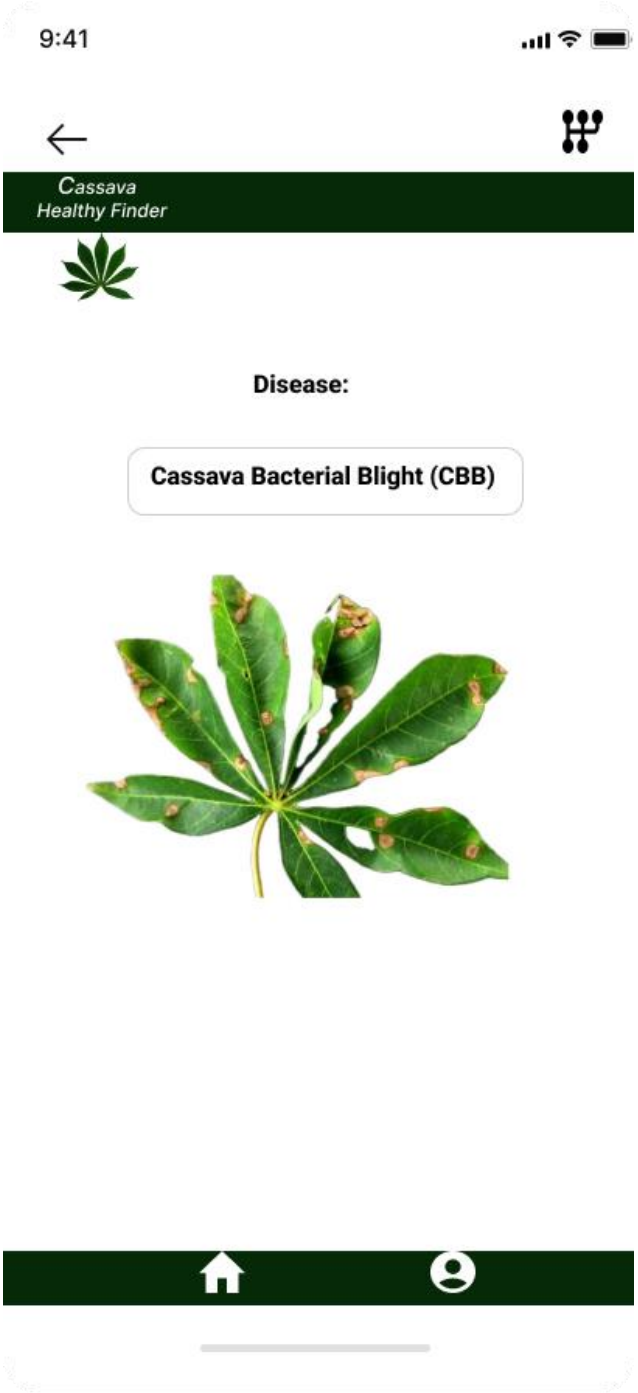

Figure 3.1.6

Description:

The page displays the identified Cassava leaf disease

3.1.7 Guide Lines Screen

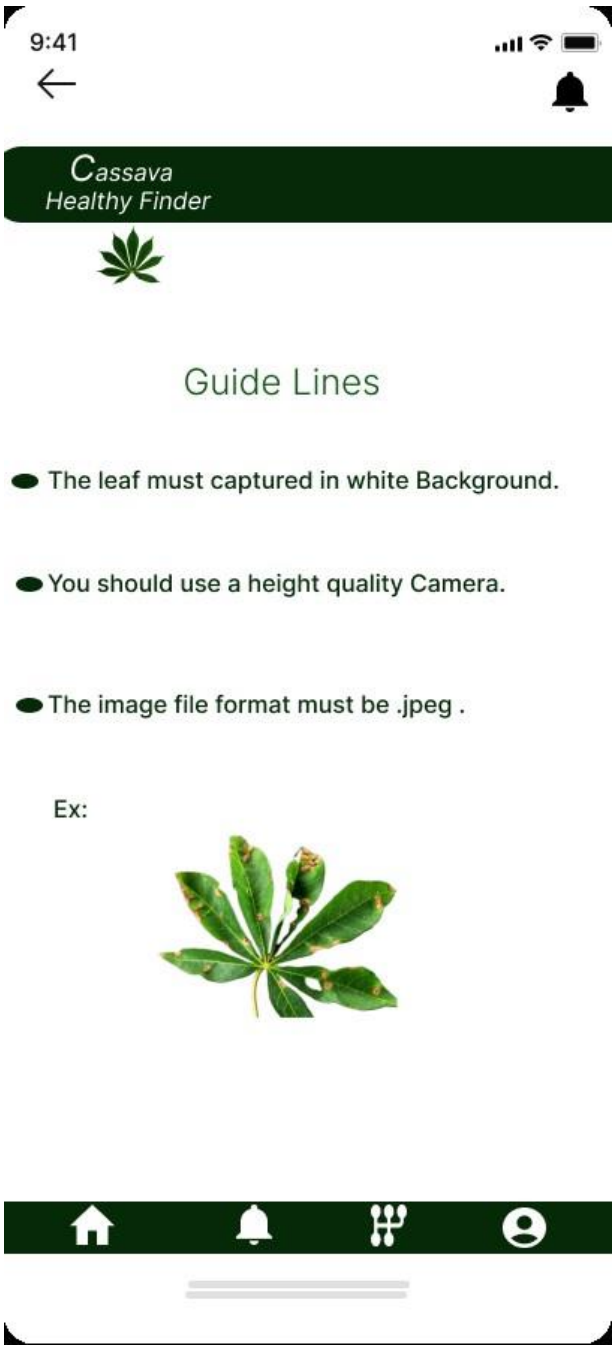

Figure 3.1.7

Description:

This page contains the guidelines for user. Users can see the instructions and a sample image. Users can go to Home screen using the Back to Home button.

3.2 Hardware Interfaces

No hardware interface requirements are applicable.

3.3 Software Interfaces

- When we develop and train AI model to identify Cassava leaf disease it involves several software components, operating systems, tools, libraries, and integrated commercial components.

❖ **Web Application Interface:**

- This interface works as a user-friendly interface of the system. So, users can interact with the AI model and access the features from this interface.
- HTML, CSS, JavaScript, and frameworks like react.js are used as technologies to build the interface.

❖ **AI model:**

- We are using an AI model to identify Cassava leaf disease.
- AI models will develop using deep learning and it uses training dataset.

❖ **Database:**

- We use MongoDB to store data related to the Cassava leaf diseases information.

❖ **Operating System:**

- The operating system mainly focuses on Windows and Mac OS.

❖ **Tools and Libraries:**

- Programming Language - Python
- Frameworks - TensorFlow
- Image recognition and classification - CNN
- Deep learning libraries - PyTorch, Keras
- Cloud based Infrastructure - MongoDB

❖ **Integrated Commercial Components:**

- We rely on cloud services to incorporate various commercial elements into our project. Above, we've outlined how our project connects and interacts. Here, we'll detail the incoming data items and messages, along with the services and channels used for communication and data sharing.

❖ **Data Items and messages:**

- Input data - Users upload the images of diseased Cassava leaves.
- Output data - The AI model provides the disease based on the input Cassava leaves. (like CBSD, CGM, CMD, CBB, and healthy leaves.)

❖ **Services and Communication:**

- User requests are communicated between the web application interface and the backend.
- The web application communicates with trained AI models to identify Cassava leaves diseases.(like CBSD, CGM, CMD, CBB, and healthy leaves.)

❖ **Data sharing:**

- Data will be shared through the database.
- The AI model accesses the training dataset to learn the Cassava leaf images with their relevant diseases.
- The web application interface gets disease information the database to display the users.

3.4 Communications Interfaces

Web Browser Communication:

Protocol: HTTP (Hypertext Transfer Protocol)

The connection between web browsers and cloud databases makes it easy to manage data and ensures smooth operation for web applications. It allows for efficient storage, retrieval, and manipulation of data.

4 System Designs

4.1 Use case Diagram

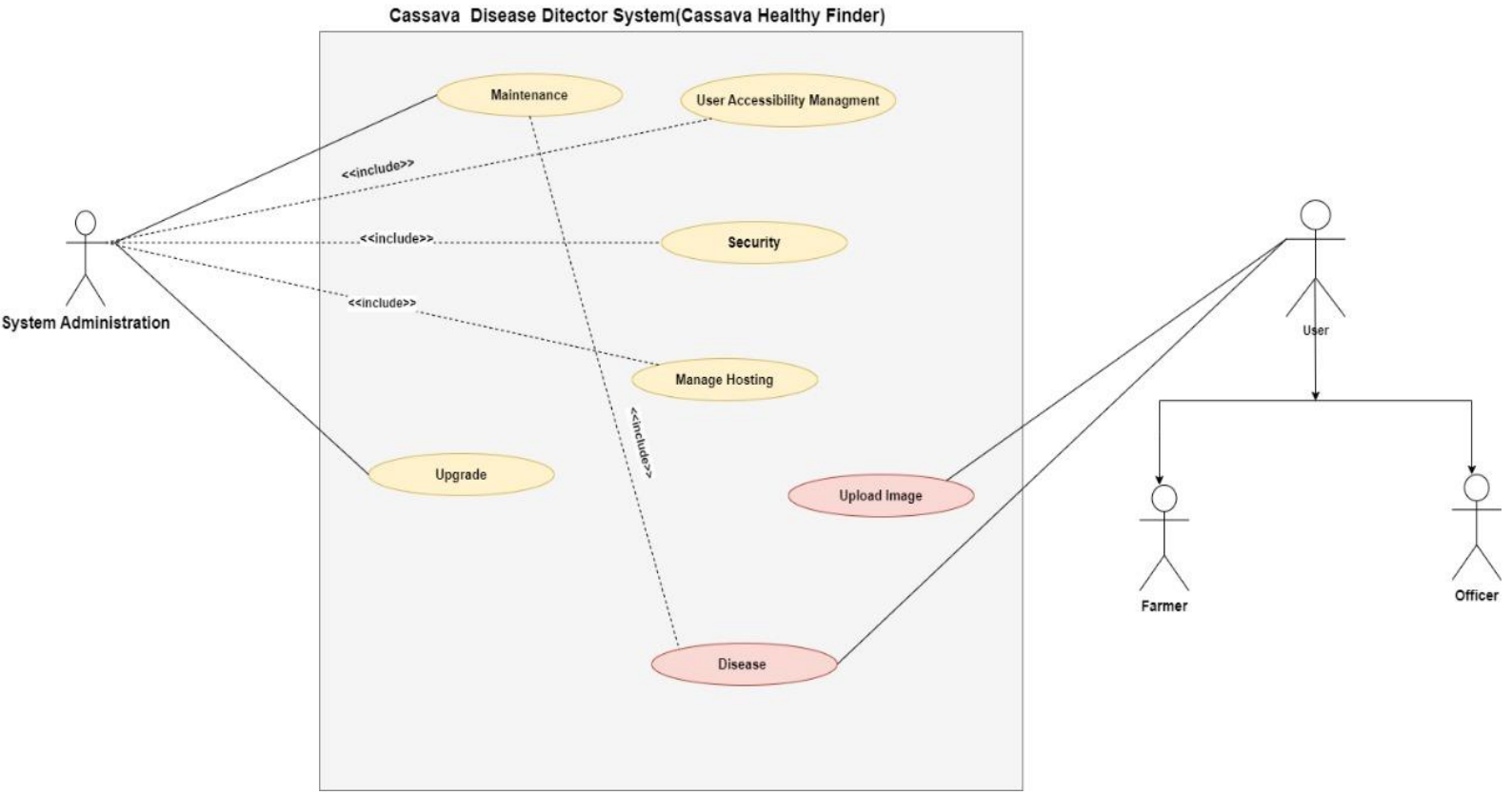

Figure 4.1 Use Case Diagram

4.1.1 Use case Description

Table 1.Use Case - Database Maintain

Use Case Name:	Maintenance	
Priority:	High	
Primary system Actor:	System Administrator	
Other Actors:	-	
Description:	In managing system maintenance, it oversees user accessibility, security measures, hosting infrastructure, and backup & restoration protocols. Furthermore, the system is designed to identify health issues and provide appropriate remedies to users. Whenever a remedy is applied, the corresponding template view is updated within the database maintenance process. Additionally, the system continuously trains on new data, integrating and storing valuable insights within its database	
Pre-Condition:	The system administrator must log in to the database.	
Trigger:	When a new data training.	
Typical course events	**Actor Action**	**System Response**
	● System Administrator login to the database and test security.	● System identified that it has bugs or some errors.

	● System Administrator saves the new data to the database.	● System must train a new data
Alternative Path:	-	
Conclusion:	System trained new data.	
Post Condition:	Log out from the database.	
Implement Constraints and Specification:	● At once, the system administrator can do one path in the maintenance. ● Cannot give earlier results from the database to the users.	
Assumption:	For major updates or changes in the system database or functionalities, the system administrator may need to coordinate with end users.	

Table 2.Use Case - Upload Image

Use Case Name:	Upload Image
Priority:	High
Primary system Actor:	● Farmer ● Officer
Other Actors:	System Administrator

Description:	Image uploading serves as a critical pathway for farmers and officers within this system. By uploading images of diseased casava leaves, they gain insights into specific diseases.	
Pre-Condition:	The farmer or officer must upload relevant diseased casava leaves.	
Trigger:	The farmers or officers want to identify diseases.	
Typical course events	**Actor Action**	**System Response**
	● Upload an image.	● Identify diseases.
	● Users can view the disease name for uploaded images.	● Display a new template for the Output Disease.
Alternative Path:	-	
Conclusion:	Users can identify casava leaves disease and get appropriate remedies.	
Post Condition:	Users can view the output results.	
Implement Constraints and Specification:	● Farmer or officer can upload one image simultaneously. ● They can't view earlier remedies for previously upload diseases from the system.	
Assumption:	-	

Table 3.Use Case – System Update

Use Case Name:	System Update	
Priority:	High	
Primary system Actor:	System Administrator	
Other Actors:	-	
Description:	System Update is an action of the system administrator. The system should be system Updated with future system features.	
Pre-Condition:	New system features must add	
Trigger:	The system administrator wants to make updates & new adjustments.	
Typical course events	**Actor Action**	**System Response**
	● System Update	● Should be system updated for new system features.
Alternative Path:	**-**	
Conclusion:	The system will be graded.	

Post Condition:	The system provides new feature actions to users.
Implement Constraints and Specification:	The previous user actions and accessibilities cannot be changed during the upgrade.
Assumption:	-

5 System Features

5.1 User Authentication

5.1.1 Description and Priority

User authentication is a critical feature in the development of a mobile app for assisting farmers with cassava disease management. It ensures that only authorized users can access the app, enhancing data security and user privacy. Prioritizing robust authentication mechanisms is essential for building trust and maintaining the integrity of the application.

5.1.2 Stimulus/Response Sequences

When a user wants to use a protected part of the app, the system asks for a username and password. Then, the user types in their username and password. The system checks if the username and password match what it has stored. If they do, the system lets the user in. But if the username and password are wrong, the system doesn't let them in and asks the user to try again or do something else to prove who they are. This process explains how user authentication works to keep things safe and only let the right people in.

5.1.3 Functional Requirements

REQ-1: The system shall prompt the user to enter their username and password.

REQ-2: The system shall verify the entered username and password against stored credentials.

REQ-3: The system shall provide a mechanism for users to reset their password in case of forgotten credentials.

5.2 Image Capture

5.2.1 Description and Priority

The Image Capture feature is important for the app because it lets users take pictures of cassava leaves using their phone's camera. This makes it easy and quick for users to collect important information needed to identify and manage diseases. By allowing

users to take pictures directly in the app, it makes things simpler and helps users get the information they need faster. This feature is a big help for farmers dealing with cassava diseases.

5.2.2 Stimulus/Response Sequences

In the image capture process, users start by tapping the option to take a picture in the app. This gets the device's camera ready to take a photo. Users then aim the camera at what they want to take a picture of, like cassava leaves, and press the button to snap the photo. After that, they can check if the picture looks good or if they need to take it again. If they're happy with the picture, they confirm it, and the app saves it for later. This straightforward process shows how users can easily take and manage pictures using the app.

5.2.3 Functional Requirements

REQ-1: The system shall provide a button or option within the app interface to initiate the image capture process.

REQ-2: The system shall allow users to preview and adjust the captured finalizing the capture.

5.3 Sharing

5.3.1 Description and Priority

The sharing feature allows users to easily share images and predictions on social media or messaging platforms. This enhances collaboration among users and facilitates knowledge exchange within the farming community. Sharing is considered a high priority as it promotes community engagement and collective efforts in disease management.

5.3.2 Stimulus/Response Sequences

The sharing feature lets users easily send images and predictions to friends or social media. This is important because it helps farmers work together and learn from each other about managing cassava diseases. When users want to share something, they just pick where they want to send it, like a messaging app or social media. Then, the app opens up that place and lets them send what they want to share. This makes it simple for farmers to help each other out and improve how they deal with cassava diseases.

5.3.3 Functional Requirements

REQ-1: The system shall provide a button or option within the app interface to initiate the image capture process.

REQ-2: The system shall allow users to preview and adjust the captured image before finalizing the capture.

5.4 Integration with Agricultural Officers

5.4.1 Description and Priority

Integration with agricultural officers allows users to directly contact agricultural experts for assistance. This feature is considered high priority as it provides users with access to professional guidance and support, enhancing the effectiveness of disease management efforts.

5.4.2 Stimulus/Response Sequences

Users can directly reach out to agricultural officers through the application by selecting the contact option. Upon selection, the app prompts users to pick a communication method, like email or phone call. Once the user chooses their preferred method, the app opens it with the agricultural officers' contact details already filled in, simplifying the process of getting in touch with experts for assistance.

5.3.3 Functional Requirements

REQ-1: The application shall provide an option for users to contact agricultural officers.

REQ-2: The application shall support multiple communication methods, including email and phone calls.

REQ-3: Users shall be able to easily access the contact details of agricultural officers within the application.

5.5 Notification System

5.5.1 Description and Priority

The notification system sends alerts for updates, disease prevention tips, and reminders for plant care. This feature is of high priority as it ensures timely dissemination of important information to users, aiding in effective disease management and plant care.

5.5.2 Stimulus/Response Sequences

When the system receives new updates, disease prevention tips, or plant care reminders, it promptly responds by sending notifications to users' devices. This ensures that users stay informed about important information related to disease management and plant care, enhancing their ability to take timely actions and maintain the health of their crops.

5.5.3 Functional Requirements

REQ-1: The system shall support sending notifications for updates.

REQ-2: The system shall send notifications for disease prevention tips.

REQ-3: The system shall send reminders for plant care.

5.5 Notification System

5.5.1 Description and Priority

The notification system sends alerts for updates, disease prevention tips, and reminders for plant care. This feature is of high priority as it ensures timely dissemination of important information to users, aiding in effective disease management and plant care.

5.5.2 Stimulus/Response Sequences

When the system receives new updates, disease prevention tips, or plant care reminders, it promptly responds by sending notifications to users' devices. This ensures that users stay informed about important information related to disease management and plant care, enhancing their ability to take timely actions and maintain the health of their crops.

5.5.3 Functional Requirements

REQ-1: The system shall support sending notifications for updates.

REQ-2: The system shall send notifications for disease prevention tips.

REQ-3: The system shall send reminders for plant care.

5.6 Feedback Mechanism

5.6.1 Description and Priority

The feedback mechanism allows users to provide input on prediction accuracy, aiming to improve the model's performance. This feature holds high priority as it facilitates continuous refinement of the model based on real-world feedback from users, ultimately enhancing its effectiveness in disease detection and management.

5.6.2 Stimulus/Response Sequences

When users come across a prediction generated by the model within the application, they are prompted by the application to offer feedback on the accuracy of the prediction. This feedback loop allows users to contribute their insights, helping to refine and enhance the model's performance over time.

5.6.3 Functional Requirements

REQ-1: The application shall include an option for users to provide feedback on predictions.

REQ-2: Users shall be able to rate the accuracy of predictions using a designated feedback mechanism.

REQ-3: The application shall collect and analyze user feedback to identify areas for model improvement and refinement.

5.6 Disease Prediction

5.6.1 Description and Priority

The Disease Prediction feature displays predicted disease categories along with their corresponding confidence scores. This functionality is of high priority as it provides users with valuable information for identifying and managing cassava diseases effectively.

5.6.2 Stimulus/Response Sequences

When users submit an image of cassava leaves for analysis, the application processes the image and produces predictions for potential disease categories, accompanied by confidence scores. This streamlined process enables users to swiftly obtain valuable insights into the health status of their cassava plants, aiding in proactive disease management efforts.

5.6.3 Functional Requirements

REQ-1: The application shall analyze submitted images of cassava leaves to predict disease categories.

REQ-2: Predicted disease categories shall be displayed to users along with corresponding confidence scores.

REQ-3: The application shall provide an interface for users to view and interpret the predicted disease categories and confidence scores effectively.

6 Other Nonfunctional Requirements

6.1 Performance Requirements

- **Image Upload Speed:** Users should be able to quickly upload images of Cassava leaves, with the system swiftly analyzing them**.**

- **Disease Detection Time:**The AI model should rapidly identify diseases in Cassava leaves, aiming for almost instant results to reduce wait times.

- **Scalability:** The system needs to handle increased user traffic and image uploads without slowing down.

- **Concurrent User Support:**The Mobile App should manage multiple users uploading images and getting diagnostic results simultaneously, without any performance issues.

- **Compatibility:** The Mobile App work smoothly on different browsers and devices, ensuring users can access it from anywhere.

- **Error Handling:** The system should effectively deal with any errors during image uploading or disease detection, providing clear error messages to help users understand and resolve issues.

6.2 Safety Requirements

- **Keeping Data Safe and Private:**
  Make sure any pictures or information users share are kept private and safe. Take steps to prevent unauthorized access or sharing of this data.

- **Safe Data Sending:**
  Use secure methods (like HTTPS) to send data between the user's device and the website server, so it can't be intercepted by unauthorized parties.

- **Controlling Who Can Access:**
  Have strong ways to check if someone is who they say they are before letting them upload pictures or see medical info. This makes sure only the right people can do these things.

- **Protecting the System and Data:**
  Use good practices to keep the website, databases, and other parts of the system safe. This includes keeping everything up-to-date, fixing any security issues quickly, and controlling who can get into the system and change things.

6.3 Software Quality Attributes

- **Adaptability:**
  The system should be able to handle different types and looks of cassava leaf images. It needs to work well with various lighting, angles, and image qualities to accurately spot diseases no matter how the pictures look.
- **Availability:**
  Users should be able to use the system whenever they need to without any issues. It should always be up and running when they want to access it.
- **Accuracy:**
  The system's disease detection results and the advice it gives should be right on point and trustworthy. We can check this by comparing its findings with known diseases and solutions, using a dataset we know is reliable or asking experts.
- **Flexibility:**
  The system should be easily upgradable and open to improvements. This means we can add new disease-spotting techniques or extra features based on what users need or new research.

- **Interoperability:**

  The webpage should fit smoothly with other systems or platforms. It should be able to share data and work well with different kinds of image files, making it easy to connect with other tech tools or databases.
- **Maintainability:**

  The code, structure, and instructions for the Mobile App should be well-organized and easy to keep up. This makes it simpler for developers to make changes or fix problems.
- **Portability:**

  The Mobile App should work smoothly on different web browsers and devices. No matter what you're using, it should look and work the same, giving everyone a consistent experience.
- **Testability:**

  The Mobile App should be built in a way that makes it easy to check how well it's working. This includes having good testing tools, datasets, and ways to fix any issues that pop up.
- **Usability:**

  The Mobile App should be easy and pleasant to use. It should be simple to understand, with clear instructions and helpful messages if something goes wrong.

7 References

[1] Convolutional Neural Networks in Detection of Plant Leaf Diseases: A Review https://doi.org/10.3390/agriculture12081192

[2] Hassan, Sk & Maji, Arnab. (2022). Plant Disease Identification Using a Novel Convolutional Neural Network. IEEE Access. PP. 1-1. 10.1109/ACCESS.2022.3141371. https://www.researchgate.net/publication/357663719_Plant_Disease_Identification_Using_a_Nov el_Convolutional_Neural_Network

[3] Analysis on Cassava leaf disease prediction using pre-trained models https://ieeexplore.ieee.org/document/9984351

[4] Cassava Syndrome Scan a Pioneering Deep Learning System for Accurate Cassava Leaf Disease Classification

https://www.researchgate.net/publication/378793845_Cassava_Syndrome_Scan_a_Pioneering_Deep_Learning_System_for_Accurate_Cassava_Leaf_Disease_Classification

[5] Attention-Based Approach for Cassava Leaf Disease Classification in Agriculture

https://www.researchgate.net/publication/373089637_Attention-Based_Approach_for_Cassava_Leaf_Disease_Classification_in_Agriculture

Websites:

[1] Convolutional Neural Network

https://www.geeksforgeeks.org/introduction-convolution-neural-network

[2] End To End Machine Learning Project With Deployment Using Flask

https://youtu.be/BUVeiXJoXHQ

[3]Leaf Disease Detection Flask App — with source code

https://medium.com/mlearning-ai/leaf-disease-detection-flask-app-with-source-code-70e46f3f59b7

Books:

[1] Digital Image Processing - 2007 Author: Rafael C. Gonzalez, Richard E. Woods

[2] Artificial Neural Systems - 1992 Author: Jacek M. Zurada

[3] Artificial Intelligence - 2007 Author: Stuart J. Russell, Peter Norvig

8 Appendix

8.1 Figures :

8.2 Tables :