UNIVERSITY PARTNER

# 6CS007 Project and Professionalism

# Artefact Design and Test Plan

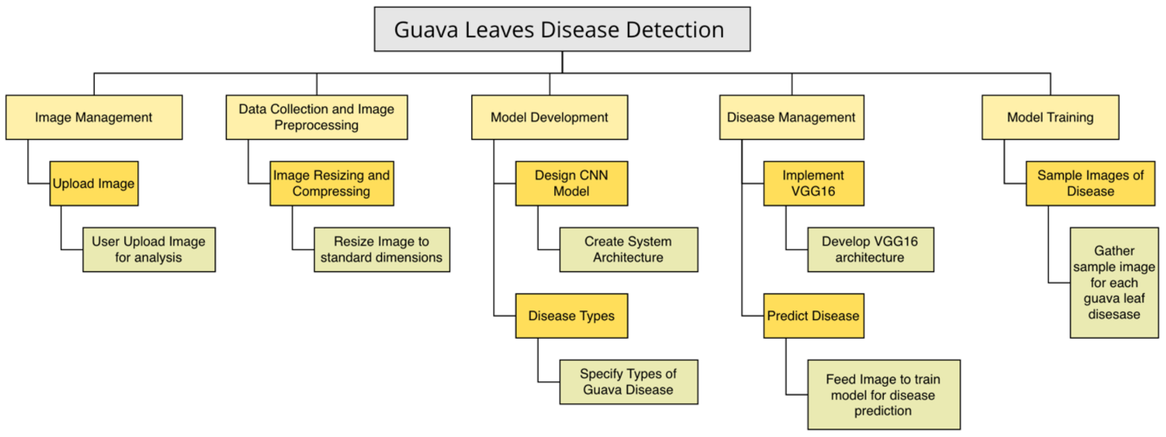
## Guava Leaf Disease Detection

| | |
|---|---|
| University ID | : 2226664 |
| College ID | : NP03CS4S220168 |
| Student Name | : Riden Aryal |
| Group | : L6CG7 |
| Submitted to | : Mr. Ashish Dahal |
| Submitted on | : 11th Feb 2024 |

## Table of Contents

## Table of Figure

## Functional Decomposition Diagram



Guava Leaves Disease Detection
Image Management
Upload Image
User Upload Image for analysis
Data Collection and Image Preprocessing
Image Resizing and Compressing
Resize Image to standard dimensions
Model Development
Design CNN Model
Create System Architecture
Disease Types
Specify Types of Guava Disease
Disease Management
Implement VGG16
Develop VGG16 architecture
Predict Disease
Feed Image to train model for disease prediction
Model Training
Sample Images of Disease
Gather sample image for each guava leaf disesase

This system provides an extensive approach for handling and detecting guava leaf disease. Data augmentation has been applied to the dataset, and users can upload images for analysis. Resizing and compressing images for standardisation is part of the pre-processing step. Convolutional Neural Network (CNN) models are intended to identify different diseases of guava leaves. Accurate disease prediction is achieved using the VGG16 architecture. For each disease, sample images must be gathered during training to ensure a robust and well-trained model. To detecting guava leaf disease, this integrated solution facilitates effective image management, data pre-processing, model development, and disease management.

# 1. The SRS of the Guava Leaf Disease Detection System

## 1.1. Aim

The aim of this document is to outline the functional, non-functional, and usability requirements for the Guava Leaf Disease Detection System. This system aims to provide a dependable and precise method of identifying diseases in guava leaves through the analysis of uploaded images.

## 1.2. Range

Features for managing images, collecting information, building models, managing diseases, and promoting user interaction will all be included in the Guava Leaf Disease Recognition System. CNN models will be used by the system primarily to detect common diseases of guava leaves.

# 2. Functional Requirement

## 2.1. Image Management

### 2.1.1. Upload Image Description:

- **Description:** To detect diseases, users ought to be able to upload pictures of guava leaves.
- **Inputs:** Images of guava leaves in common formats (JPEG, PNG, etc.) are the inputs.
- **Processing:** To facilitate further analysis, the system should accept and pre-process the uploaded images.

### 2.1.2. Data Augmentation

- **Description:** Apply data augmentation techniques to increase the diversity of the dataset.
- **Processing:** The system should be able to manipulate images by rotating and flipping them.

## 2.2. Data Collection and Image Pre-processing

### 2.2.1. Image Resizing and Compressing

- **Description:** Resize photos to a common size that is appropriate for processing.
- **Processing:** The dimensions of the input images should be standardised by the system.

## 2.3. Model Development

### 2.3.1. Design CNN Model

- **Description:** Describe the Convolutional Neural Network (CNN) architecture used in disease detection.
- **Inputs:** Layers, activation functions, and model complexity requirements.
- **Processing:** Use libraries like Keras or TensorFlow to create a CNN model.

### 2.3.2. Disease Type

- **Description:** Name the varieties of Guava leaf diseases that the model can identify.
- **Inputs:** List of illnesses to be included in the classification model is one of the inputs.

## 2.4. Disease Management

### 2.4.1. Implement VGG

- **Description:** Use the VGG16 model to extract features.
- **Processing:** For reliable feature learning, make use of the VGG16 architecture.

### 2.4.2. Predict the Disease

- **Description:** Identify the kind of illness depicted in an image of a guava leaf.

- **Processing:** Provide pre-processed picture data to the model that has been trained to classify diseases.

### 2.5. Training Model

#### 2.5.1. Sample Images of Disease

- **Describe:** Compile a large dataset of labelled photos for training.
- **Inputs:** Various Guava leaf samples, one for each disease.
- **Processing:** Use the gathered dataset to train the CNN model.

## 3. Non-functional Requirements

### 3.1. Performance

- At least 65% of diseases should be predicted by the system.
- A 5-second response time is ideal for illness prediction.

### 3.2. Functionality

- At least four common guava leaf diseases should be detectable by the system.
- It should manage uploads of images in a few widely used formats.

### 3.3. Availability

- The system should be available 24/7 for users to upload and analyse images.

### 3.4. Flexibility

- Based on user feedback, the system should advance the interface and make it more usable.
- For scalability, cloud integration should be possible.

### 3.5. Reliability

- The system should provide consistent and reliable disease predictions.

## 4. Usability Requirements

### 4.1. User Interface

- The user interface should be simple and user-friendly.
- User should have the facility to upload images and view the disease predictions.

### 4.2. Scalability

- The system should be scalable to handle large volume of Guava leaf images.
- It should be supported on a range of hardware platforms.

## 5. System Requirements

A system consists of two components: Hardware and Software, which are discussed below.

### 5.1. Software requirements.

The project's software requirements are as follows. The use of all required software, including data management systems, is outlined in the software requirements. The versions and numbers are specified in the necessary software product. The purpose of each software interface is indicated in relation to the software product.

- Operating systems: Mac, Linux, and Windows.
- Python 3+ is the programming language.
- OpenCV, Keras, TensorFlow, and Tkinter are among the libraries.

### 5.2. Hardware requirements.

Certain hardware specifications are required for the guava leaf disease detection system to function properly. The project is intended to run optimally on hardware with Intel x86 or x64 processors, including Core i5, i7, or higher variants, as well as arm64 architecture for Mac or Linux support.

- The project requires the following hardware: • Intel x86 or x64 processors, including Core i5, i7, or higher versions, arm64 mac, or linux.
- Requires 8GB of RAM, 1TB of hard drive space, and 1GB GPU.
- Compatible operating systems

**Functional Requirements:**

| **Requirement Code** | **Requirement Description** | **Use Case and Activity** |
|---|---|---|
| **FM-01** | Users should be able to upload images of Guava leaves for disease detection. | Upload Image – (ATD02, USD03) |
| **FM-02** | Apply data augmentation techniques to increase the diversity of the dataset. | Data Augmentation – (USD04) |
| **FM-03** | Resize images to a standard dimension suitable for processing. | Image Resizing and Compressing – (ATD02) |
| **FM-04** | Define the architecture of the Convolutional Neural Network (CNN) for disease detection. | Design CNN Model – (USD02, ATD04) |
| **FM-05** | Specify the types of Guava leaf diseases the model will detect. | Type of Diseases – (ATD05) |
| **FM-06** | Implement the VGG16 model for feature extraction. | Implement VGG – (ATD04) |
| **FM-07** | Predict the type of disease present in a Guava leaf image. | Predict the Disease - (ATD05, USD02) |
| **FM-08** | Gather a comprehensive dataset with labelled images for training. | Sample Images for Disease – (USD02) |

**Non-functional Requirements:**

| **Requirement Code** | **Requirement Description** | **Use Case and Activity** |
|---|---|---|
| **NF-01** | The system should predict diseases with at least 65% accuracy. | Predict the Disease – (ATD05, USD02) |
| **NF-02** | Response time for disease prediction should be within 5 seconds. | Predict the Disease – (ATD05, USD02) |
| **NF-03** | The system should support the detection of at least four common Guava leaf diseases. | Type of Diseases –(ATD05) |
| **NF-04** | It should handle image uploads in various common formats. | Upload Image – (ATD02, USD03) |
| **NF-05** | The system should be available 24/7 for users to upload and analyse images. | Overall System – (ATD01, USD01) |
| **NF-06** | Based on user feedback, the system should advance the interface and make it more usable. | Type of Diseases – (ATD05) |
| **NF-07** | Cloud integration should be feasible for scalability. | Overall System – (ATD01, USD01) |
| **NF-08** | The system should provide consistent and reliable disease predictions. | Overall System – (ATD01, USD01) |

**Usability Requirements:**

| **Requirement Code** | **Requirement Description** | **Use Case and Activity** |
|---|---|---|
| **UR-01** | The user interface should be intuitive and user-friendly. | Overall System – (ATD01, USD01) |
| **UR-02** | Users should easily upload images, view disease predictions, and adjust settings. | Upload Image, Predict the Disease – (USD02) |
| **UR-03** | The system should scale to handle large volumes of Guava leaf images. | Overall System – (ATD01, USD01) |
| **UR-04** | It should be deployable on a range of hardware platforms. | Overall System – (ATD01, USD01) |
| **UR-05** | The system should integrate with other software and hardware systems. | Overall System – (ATD01, USD01) |

## 6. System Architecture

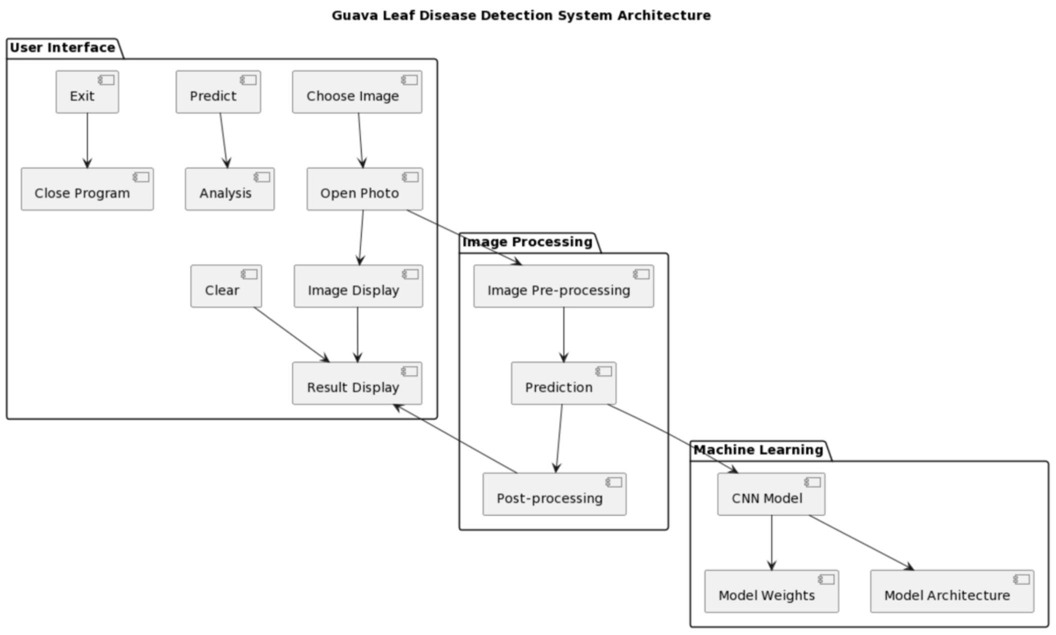


*Figure 1: System Architecture*

The user interface starts with image selection, then displays selected images and analysis results, allowing users to predict, clear, exit, or close the programme easily. Image processing consists of opening images, pre-processing for predictions, and post-processing. In the machine learning module, predictions are made using a Convolutional Neural Network (CNN) model that includes model architecture and weights. This streamlined system architecture prioritises user-friendly interactions, efficient image processing, and a well-defined machine learning pipeline for precise predictions, resulting in a seamless and accessible experience from image selection to insightful analysis.

## 7. Sequence Diagram

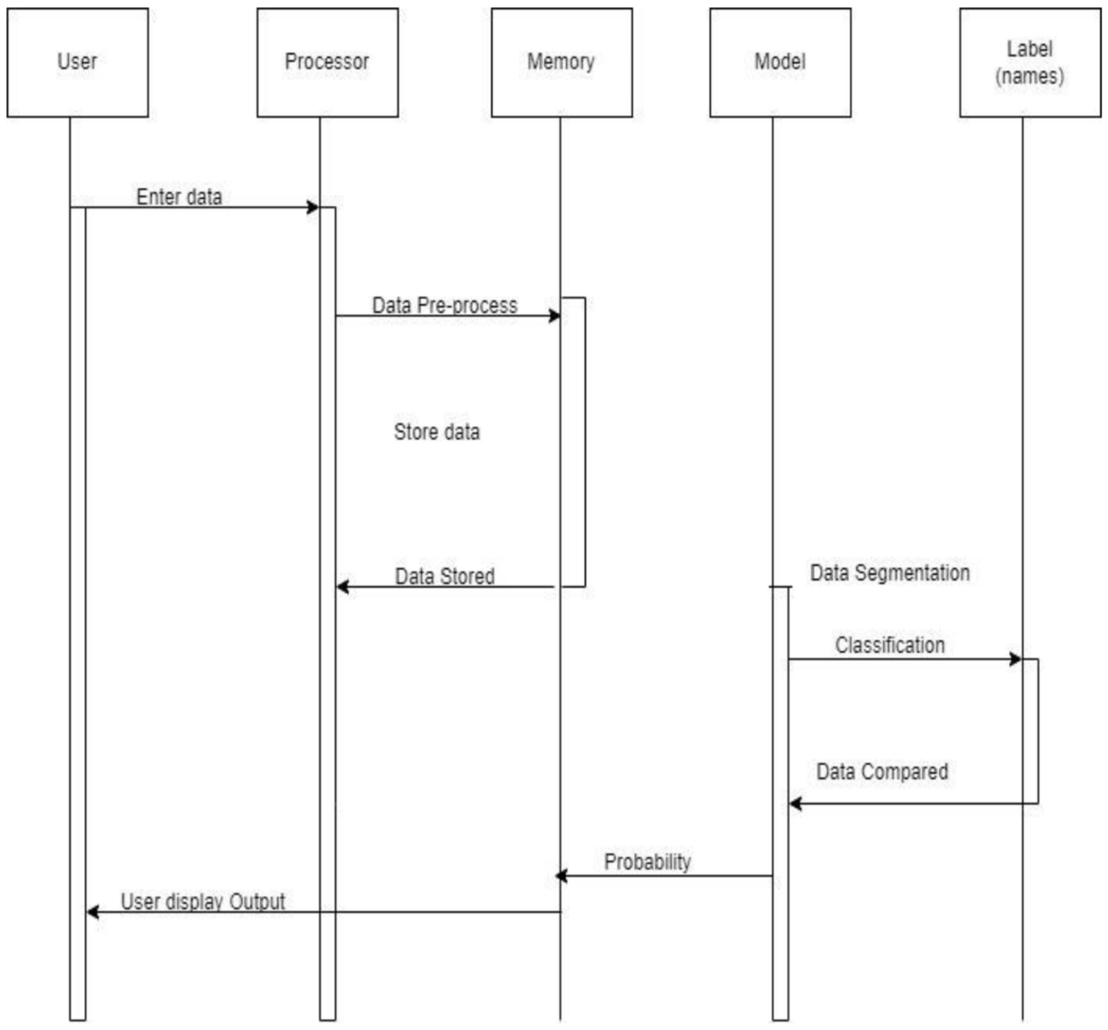


*Figure 2: Sequence Diagram*

As seen in the above figure, the sequence diagram is composed of five blocks: user, processor, memory, model, and labels. The user inputs data from saved picture files, which are then saved in the memory unit after being pre-processed for differentiator parameters. The trained model is loaded, and features are extracted to classify output after files have been pre-processed and stored.

## 8. Activity Diagram.

An activity diagram is a type of behavioural diagram that shows how a system behaves. An activity diagram displays the entire control flow, including all decision-making paths taken during execution. To ensure error-free code execution, we first import all required library packages. When the code is executed, it produces the desired results.

### 8.1. Activity Diagram Overall System (ATD01)

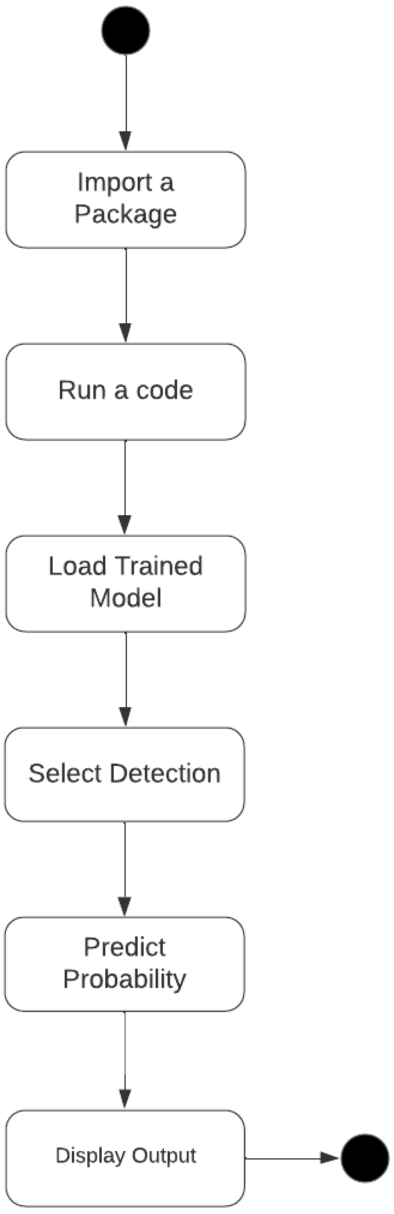


*Figure 3: Activity Diagram - ATD01*

## 8.2. Activity Diagram of Subsystem: Image Management (ATD02)

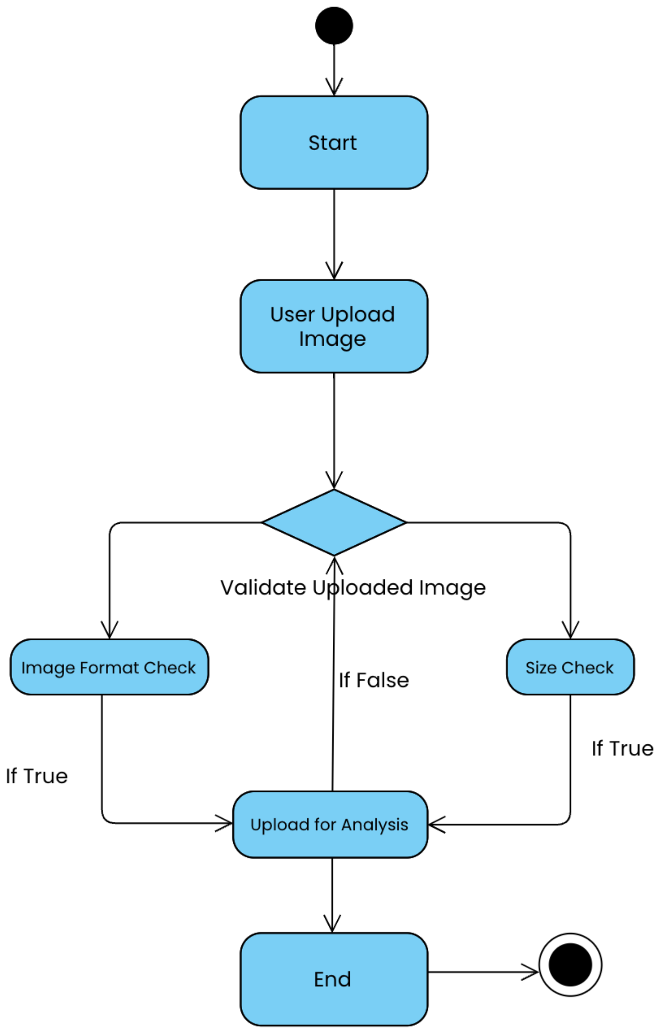


*Figure 4: Activity Diagram - ATD02*

## 8.3. Activity Diagram of Subsystem: Data Collection and Image Pre-Processing (ATD03)

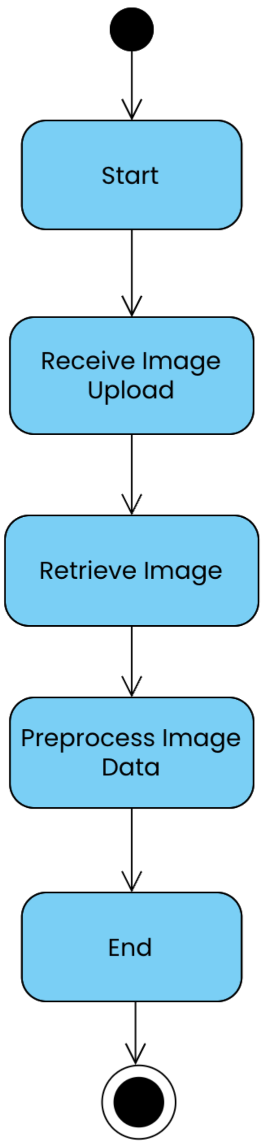


*Figure 5: Activity Diagram - ATD03*

## 8.4. Activity Diagram of Subsystem: Model Development (ATD04)

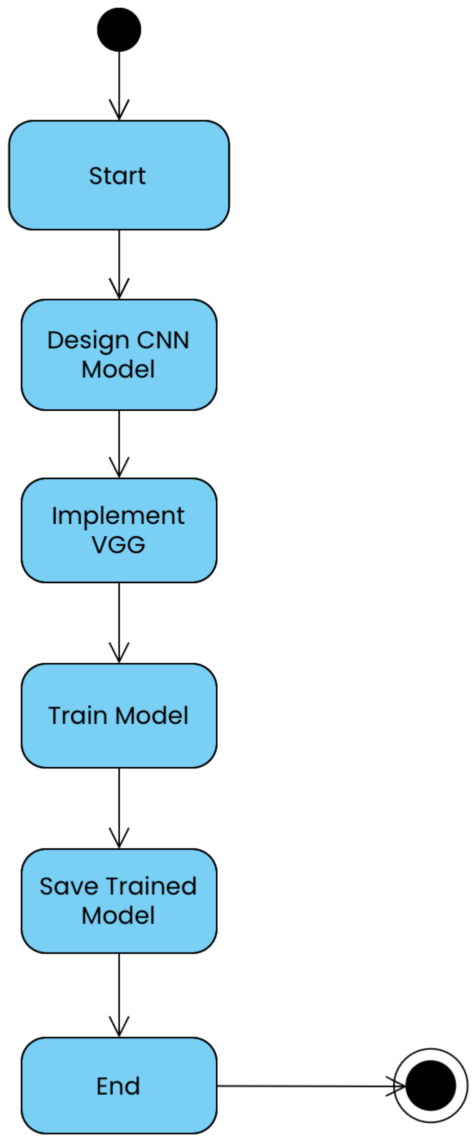


*Figure 6: Activity Diagram - ATD04*

## 8.5. Activity Diagram of Subsystem: Disease Management (ATD05)

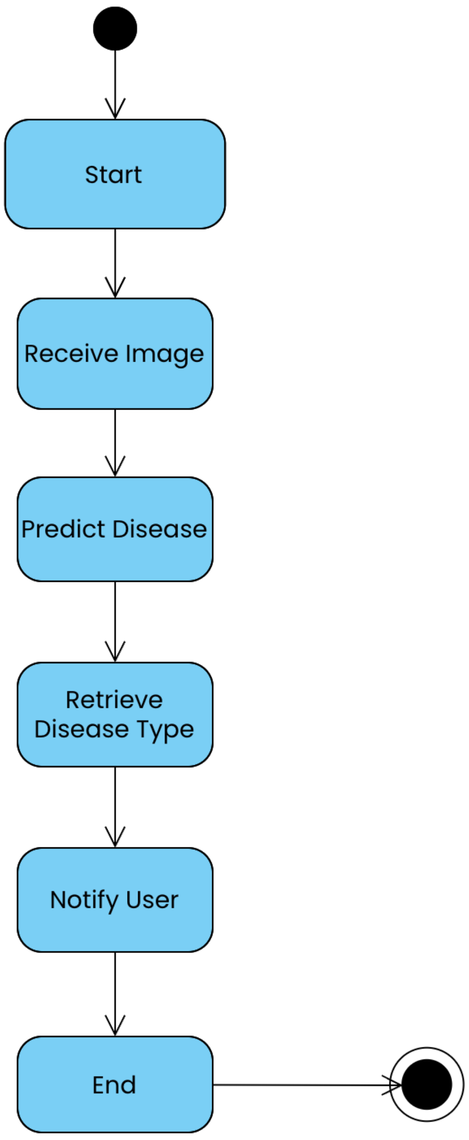


*Figure 7: Activity Diagram - ATD05*

## 9. Dataflow and Workflow Diagram

### 9.1. Dataflow Diagram, Level 0

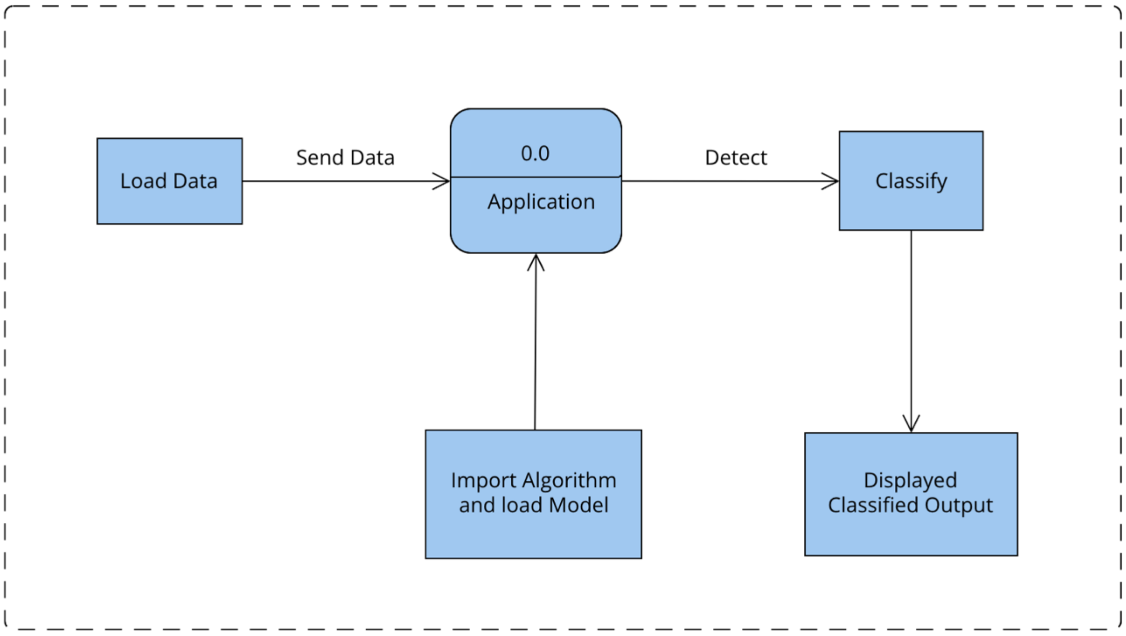


*Figure 8: Dataflow Diagram, Level 0*

The above diagram represents DFD0, which provides an overview of the system. The system is presented as a single high-level process, providing an easy overview. The application loads data from a file and sends it to a classification unit for prediction using a trained model file. The output is then known.

## 9.2. Dataflow Diagram, Level 1

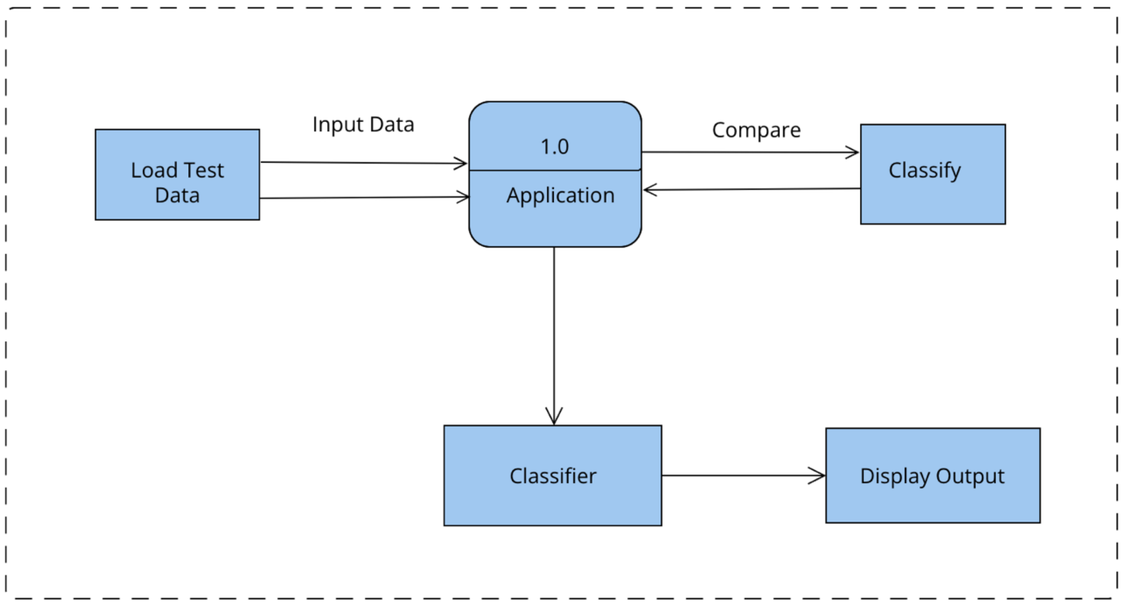


*Figure 9: Dataflow Diagram Level 1*

The following diagram represents DFD1. Level 0 DFD is further categorised as Level 1 DFD. Level 1 DFD depicts basic system modules and data flow between them. The application loads data from a file, which is then sent to a classification unit for prediction. Classes are then classified and labelled.

## 9.3. Workflow Diagram

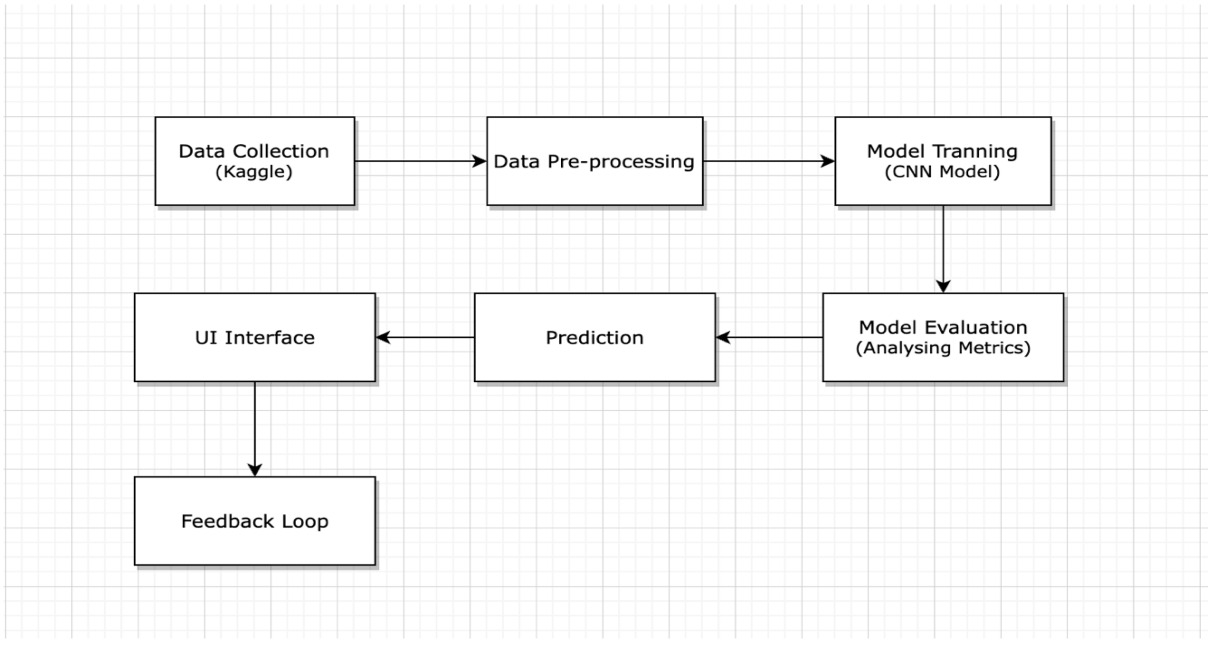


*Figure 10: Workflow Diagram*

## 10. Use case diagram.

The use case consists of a user and a processor. The user provides input to the system, while the processor processes and outputs it. The flow is represented in the diagram above. The first user runs the system, importing and loading code, model, and library packages. The code runs and displays output based on the input data.

### 10.1. Use case diagram of Overall System (USD01)

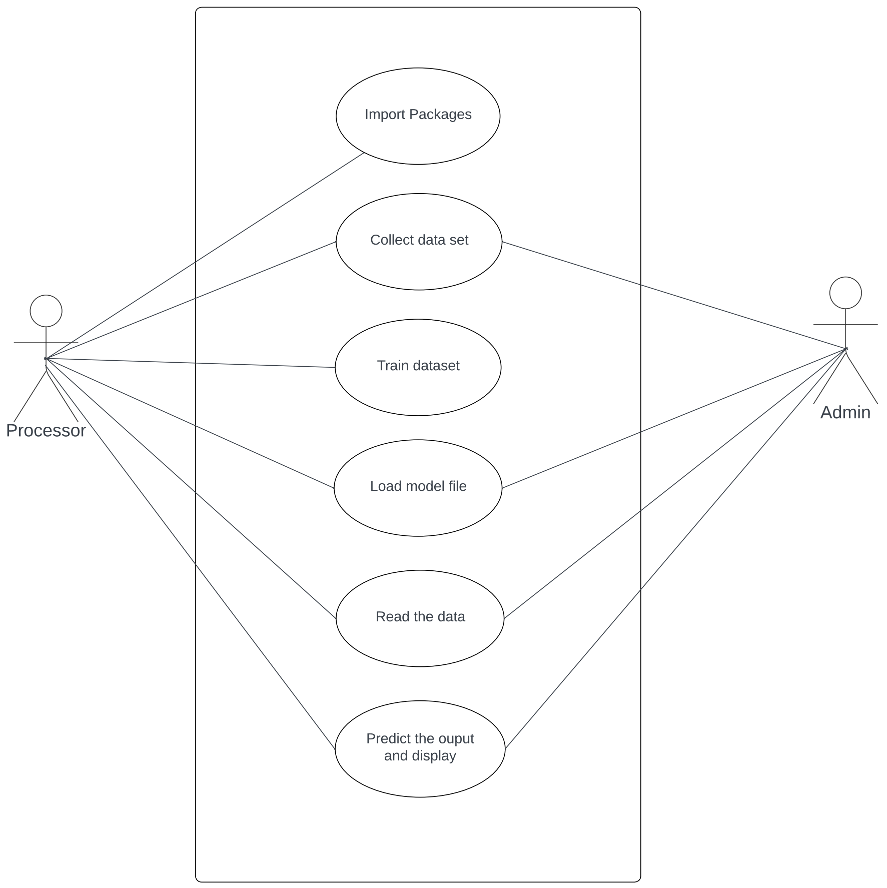


*Figure 11: Use case Diagram - USD01*

## 10.2. Use case diagram of Subsystem: Model Development (USD02)

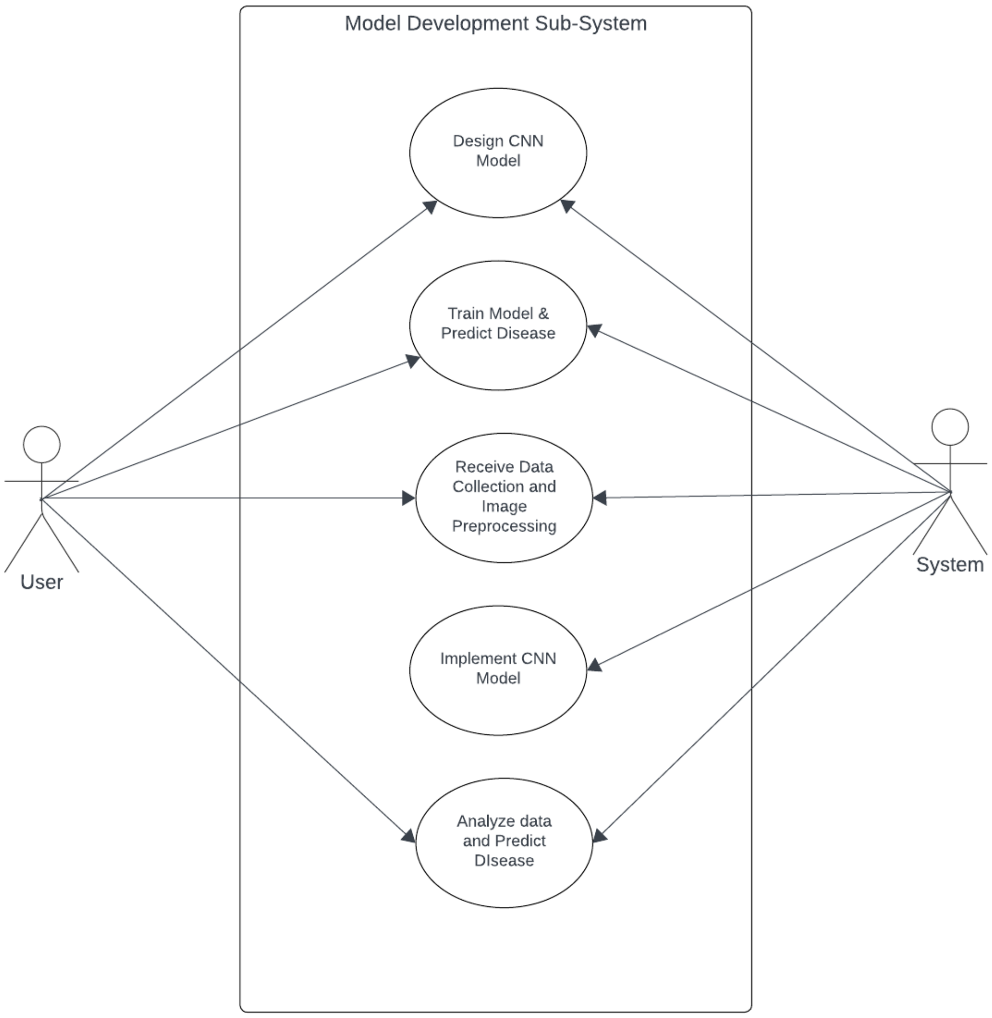

Model Development Sub-System
Design CNN Model
Train Model & Predict Disease
Receive Data Collection and Image Preprocessing
Implement CNN Model
Analyze data and Predict DIsease
User
System

*Figure 12: Use case Diagram - USD02*

## 10.3. Use case diagram of Subsystem: Image Management (USD03)

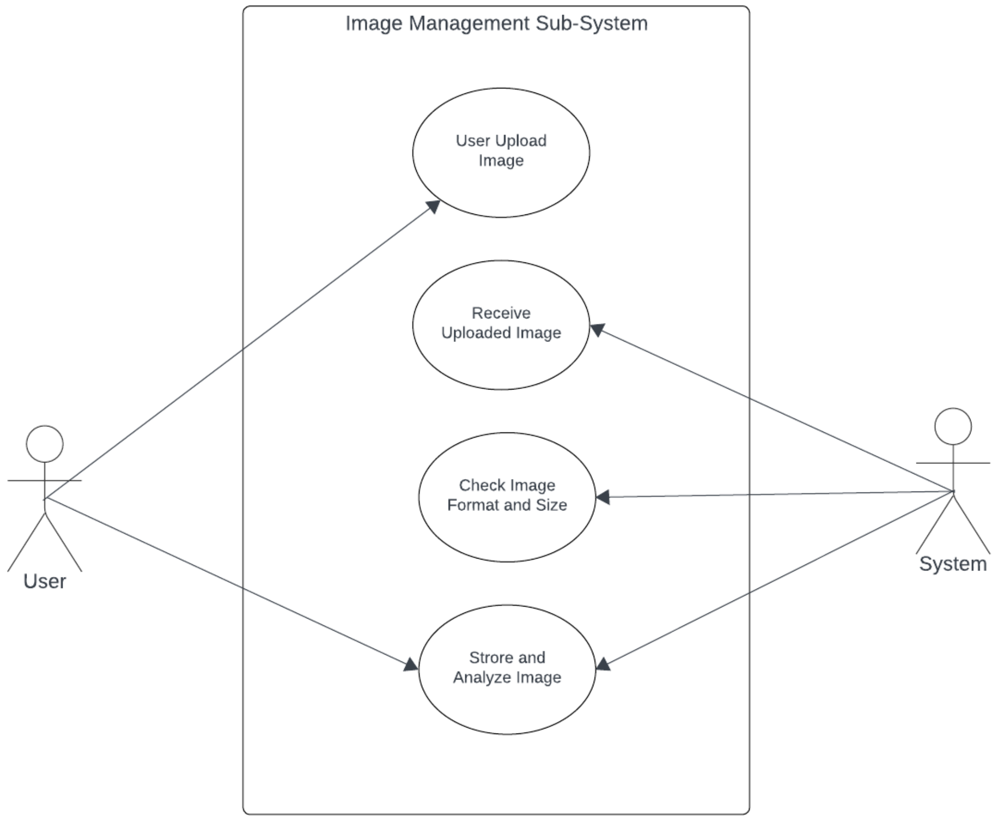


*Figure 13: Use case Diagram - USD03*

## 10.4. Use case diagram of Subsystem: Data Collection and Image Pre-processing (USD04)

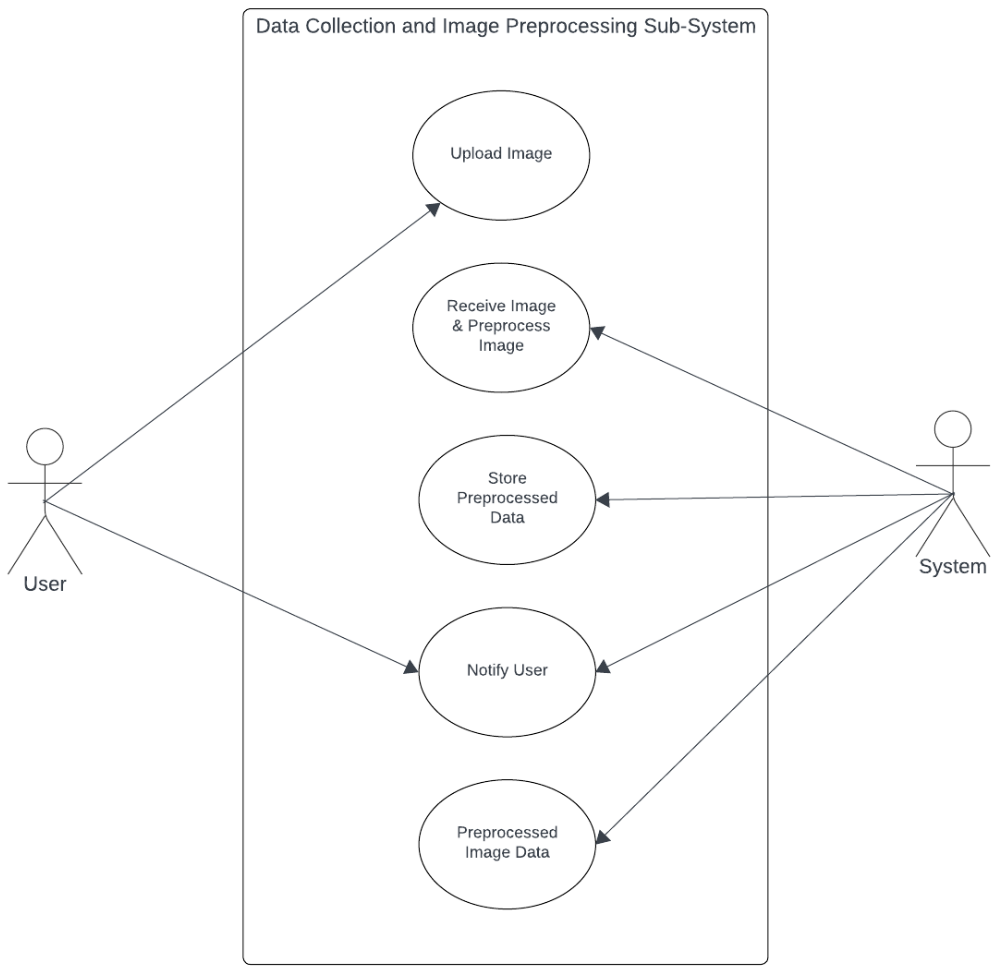

Data Collection and Image Preprocessing Sub-System
Upload Image
Receive Image & Preprocess Image
Store Preprocessed Data
Notify User
Preprocessed Image Data
User
System

*Figure 14: Use case Diagram - USD04*

## 11. Wireframe

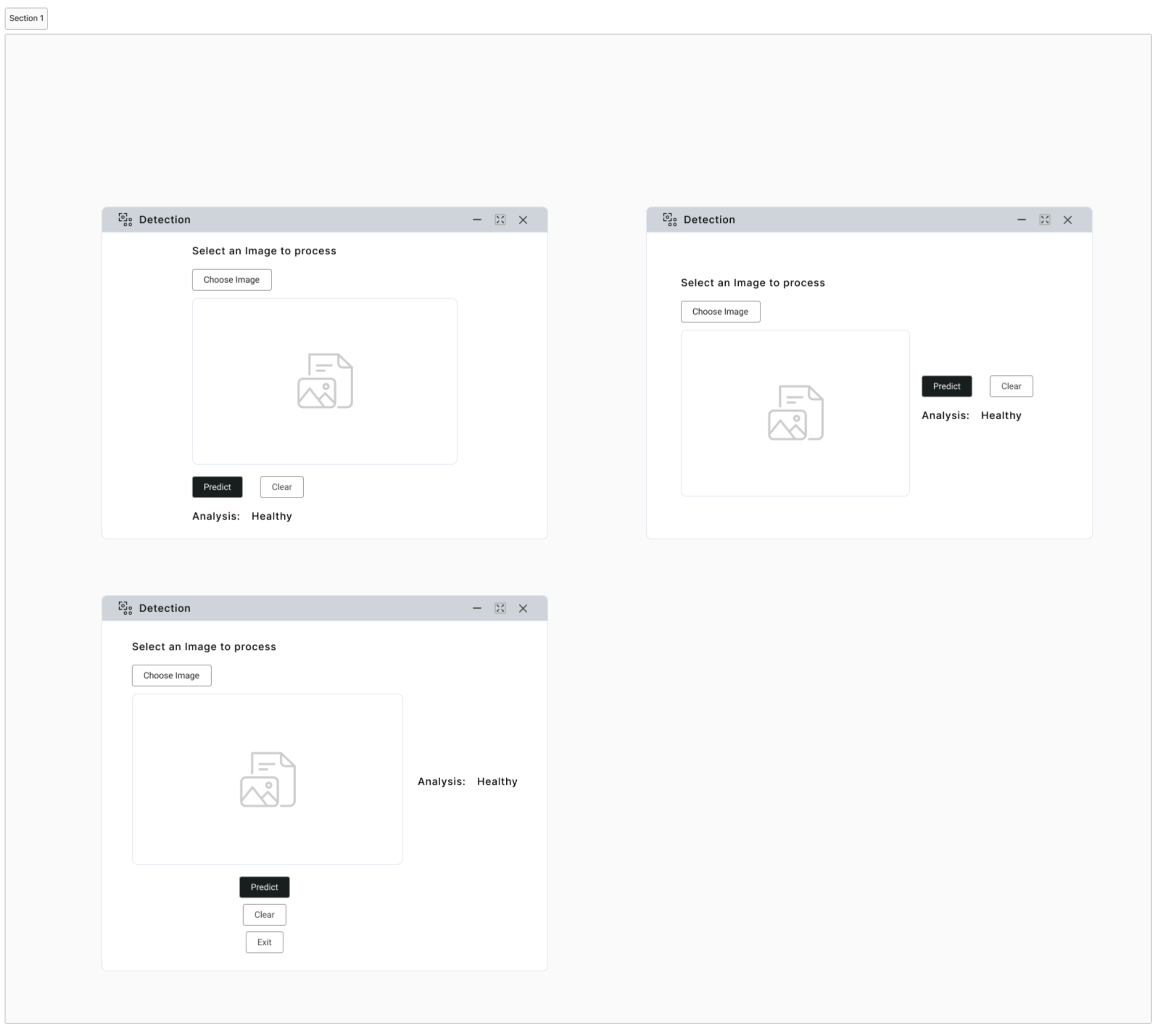


*Figure 15: Wireframe*

The designed wireframe focuses on simplicity for farmers who use the guava leaf disease detection system. The user interface is purposefully simple, ensuring a seamless experience for farmers. The primary focus is on ease of use, allowing farmers to easily import leaf images and quickly identify any diseases present. This streamlined approach addresses the specific needs of farmers by highlighting a simple process for image input and disease visualisation, thus enhancing accessibility and usability in the agricultural context.

## 12. Testing

Testing is a crucial method employed to ensure the absence of errors in a software product and assess its adherence to expected requirements. It involves the manual or automated execution of software/system components using various tools to identify mistakes, gaps, or missing requirements compared to the specified ones. Some individuals prefer using the terms "White Box" and "Black Box" to characterize software testing. In essence, software testing verifies the Application Under Test (AUT). Detecting and resolving bugs or errors early in the development process is the primary objective of testing, ensuring the final product's dependability, safety, and high performance. Thoroughly tested software brings about cost savings, efficiency, and customer satisfaction.

Several advantages are associated with testing:

- **Cost-effectiveness:** Timely testing of IT projects leads to long-term financial savings, as addressing bugs in the early stages is more economical.

- **Security:** Testing is crucial for ensuring the reliability of software products, helping identify and address risks and issues in advance.

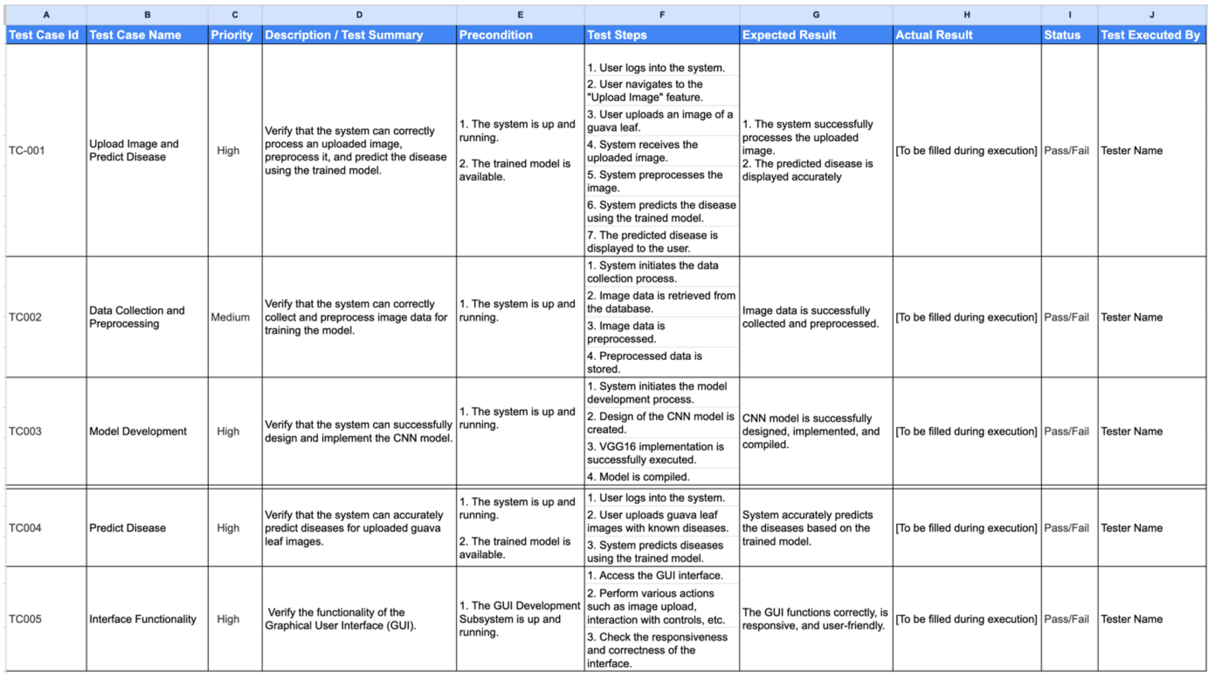
- **Product quality:** Testing is a prerequisite for any software product, guaranteeing that clients receive a high-quality and reliable product.

- **Customer Satisfaction:** The goal of a product is to ensure customer happiness, and testing, especially in terms of User Interface/User Experience (UI/UX), contributes to the best user experience.

Testing is broadly categorized into three types:

- Functional Testing
- Non-Functional Testing or Performance Testing
- Maintenance (Regression and Maintenance) Testing

## 12.1. Test Cases



| A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|
| Test Case Id | Test Case Name | Priority | Description / Test Summary | Precondition | Test Steps | Expected Result | Actual Result | Status | Test Executed By |
| TC-001 | Upload Image and Predict Disease | High | Verify that the system can correctly process an uploaded image, preprocess it, and predict the disease using the trained model. | 1. The system is up and running.<br>2. The trained model is available. | 1. User logs into the system.<br>2. User navigates to the "Upload Image" feature.<br>3. User uploads an image of a guava leaf.<br>4. System receives the uploaded image.<br>5. System preprocesses the image.<br>6. System predicts the disease using the trained model.<br>7. The predicted disease is displayed to the user. | 1. The system successfully processes the uploaded image.<br>2. The predicted disease is displayed accurately | [To be filled during execution] | Pass/Fail | Tester Name |
| TC002 | Data Collection and Preprocessing | Medium | Verify that the system can correctly collect and preprocess image data for training the model. | 1. The system is up and running. | 1. System initiates the data collection process.<br>2. Image data is retrieved from the database.<br>3. Image data is preprocessed.<br>4. Preprocessed data is stored. | Image data is successfully collected and preprocessed. | [To be filled during execution] | Pass/Fail | Tester Name |
| TC003 | Model Development | High | Verify that the system can successfully design and implement the CNN model. | 1. The system is up and running. | 1. System initiates the model development process.<br>2. Design of the CNN model is created.<br>3. VGG16 implementation is successfully executed.<br>4. Model is compiled. | CNN model is successfully designed, implemented, and compiled. | [To be filled during execution] | Pass/Fail | Tester Name |
| TC004 | Predict Disease | High | Verify that the system can accurately predict diseases for uploaded guava leaf images. | 1. The system is up and running.<br>2. The trained model is available. | 1. User logs into the system.<br>2. User uploads guava leaf images with known diseases.<br>3. System predicts diseases using the trained model. | System accurately predicts the diseases based on the trained model. | [To be filled during execution] | Pass/Fail | Tester Name |
| TC005 | Interface Functionality | High | Verify the functionality of the Graphical User Interface (GUI). | 1. The GUI Development Subsystem is up and running. | 1. Access the GUI interface.<br>2. Perform various actions such as image upload, interaction with controls, etc.<br>3. Check the responsiveness and correctness of the interface. | The GUI functions correctly, is responsive, and user-friendly. | [To be filled during execution] | Pass/Fail | Tester Name |

*Figure 16: Test Cases*

**Test Case Link:**

https://docs.google.com/spreadsheets/d/1YOGTsAGmmotMNVwjdn0xV1X-fu5-Uhi3ZIloqxVDlh0/edit?usp=sharing