**The Islamia University of Bahawalpur**

**Department of Software Engineering**

**Faculty of computing**

****

**Software Design Document**

**(SDD DOCUMENT)**

**For**

**< Detection of Cotton Crop Diseases >**

Version 1.0

***By***

**Irsa Rafique**

**S22BSEEN1M01125**

**Section: Morning M1**

**Session Spring 2022 – 2026**

***Supervisor***

**Mr. Daler Ali Awan**

Version: 1.0

Detection of Cotton Crop Diseases

Software Design Document

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
|  |  | Irsa Rafique |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Mr. Daler Ali Awan | Supervisor |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. INTRODUCTION 4

1.1 Purpose 4

1.2 Scope 4

1.3 Overview 5

1.4 Reference Material 5

1.5 Definitions and Acronyms 6

2. SYSTEM OVERVIEW 6

3. SYSTEM ARCHITECTURE 8

3.1 Architectural Design 8

3.2 Decomposition Description 9

3.3 Design Rationale 18

4. DATA DESIGN 19

4.1 Data Description 19

4.2 Data Dictionary 22

5. COMPONENT DESIGN 23

6. HUMAN INTERFACE DESIGN 24

6.1 Overview of User Interface 24

6.2 Screen Images 26

6.3 Screen Objects and Actions 27

7. REQUIREMENTS MATRIX 28

# 1. INTRODUCTION

## 1.1 Purpose

This software design document (SDD) describes the architecture and system design of a mobile app for detecting bacterial blight, curl virus, and Fusarium wilt diseases of cotton crops using computer vision techniques. The purpose of this SDD is to provide a comprehensive overview of the software design, including its system architecture, user interface design, data model, algorithms, and other technical details. The intended audience of this SDD includes the project developer student. The SDD serves as a blueprint to build the app and as a reference for project person to understand the app's design decisions and technical aspects.

## 1.2 Scope

The scope of this software project is to develop a mobile app for detecting three common diseases of cotton crops that were used as information sources, app will use computer vision techniques to analyze images of cotton leaves and identify any signs of disease. The goals of this project are to provide a simple and efficient way for farmers to monitor the health of their cotton crops, improve crop yields and reduce losses due to disease, and reduce the use of pesticides by allowing farmers to target treatment only where it is needed.

The objectives of this project include:

* Designing a user-friendly mobile app interface that allows users to easily upload images of cotton leaves and receive accurate disease detection results.
* Using or developing a robust computer vision algorithm that can accurately detect and classify the three targeted diseases based on visual symptoms in the images.
* Integrating the Ai trained model with the mobile app to enable real-time analysis of images and disease detection.
* Implementing a data model and database to store and manage user data, image data, and disease detection results.
* Testing the app thoroughly to ensure its accuracy, reliability, and usability.

The benefits of this project include:

* Helping farmers to identify diseases early and take appropriate action to prevent further spread, leading to improved crop yields and reduced losses.
* Reducing the use of pesticides by allowing farmers to target treatment only where it is needed, resulting in cost savings and environmental benefits.
* Providing a simple and user-friendly tool for farmers to monitor the health of their crops, without requiring specialized training or equipment.
* Demonstrating the potential of computer vision and mobile app technologies to improve agricultural practices and support sustainable farming.

## 1.3 Overview

This software design document (SDD) provides a detailed description of the design and architecture of a mobile app for detecting the diseases of cotton crop using computer vision techniques. The SDD is organized into several sections, each covering a specific aspect of the software design.

Section 2 describes the system architecture of the mobile app, including the components, interfaces, and interactions between them. Section 3 covers the user interface design, including the layout, navigation, and functionality of the app's screens and features.

Section 4 details the data model and database design, including the entities, attributes, relationships, and data storage and retrieval methods used in the app. Section 5 discusses the algorithms and techniques used in the comp high-level system for disease detection, including image processing, feature extraction, and machine learning.

Section 6 outlines the testing and quality assurance procedures used to ensure the accuracy, reliability, and usability of the app. Finally, Section 7 summarizes the key points of the SDD and provides recommendations for future enhancements and improvements to the software design.

Overall, this SDD provides a comprehensive overview of the design and architecture of the mobile app, enabling developer to understand the technical details and make informed decisions about the development and implementation of the software.

## 1.4 Reference Material

* + **IEEE** **Std 830TM-1998 (R2009**) – IEEE Standard for Information Technology – Systems Design – Software Design Descriptions  
    "Detection and classification of cotton plant diseases using computer vision techniques" by El-Dahshan et al.
  + "Automatic detection and classification of cotton diseases using image processing techniques" by Patel et al.
  + "Real-time cotton disease recognition using deep learning techniques" by Saad et al.
  + "Computer Vision: Algorithms and Applications" by Richard Szeliski
  + "Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods
  + "Pattern Recognition and Machine Learning" by Christopher M. Bishop
  + "Plant Disease Detection Using Computer Vision Techniques: A Review" (<https://www.researchgate.net/publication/331002725_Plant_Disease_Detection_Using_Computer_Vision_Techniques_A_Review>)
  + "A Guide to Convolutional Neural Networks for Computer Vision" (<https://pathmind.com/wiki/convolutional-network>)

## 1.5 Definitions and Acronyms

Definitions:

1. Cotton Crop Diseases: Various infections or abnormalities that affect cotton plants, such as Bacterial Blight, Curl Virus, and Fusarium Wilt.
2. Computer Vision: A field of study that focuses on enabling computers to interpret and understand visual information from digital images or videos.
3. AI (Artificial Intelligence): The simulation of human intelligence processes by machines, particularly in tasks involving learning, reasoning, and problem-solving.
4. Trained Model: A machine learning model that has undergone training using labeled data to make predictions or classifications on new, unseen data.
5. Server: A remote computer or system that hosts the AI trained model and provides the prediction services to the Android app.
6. RESTful API: A software architectural style that allows communication between different systems over the internet using standard HTTP protocols, typically based on Representational State Transfer (REST) principles.
7. SDD (Software Design Document): A document that describes the software design, architecture, components, and interactions of a software system.

Acronyms:

1. AI: Artificial Intelligence
2. API: Application Programming Interface
3. HTTP: Hypertext Transfer Protocol
4. HTTPS: Hypertext Transfer Protocol Secure
5. JSON: JavaScript Object Notation
6. SDD: Software Design Document
7. UI: User Interface
8. UX: User Experience

# 2. SYSTEM OVERVIEW

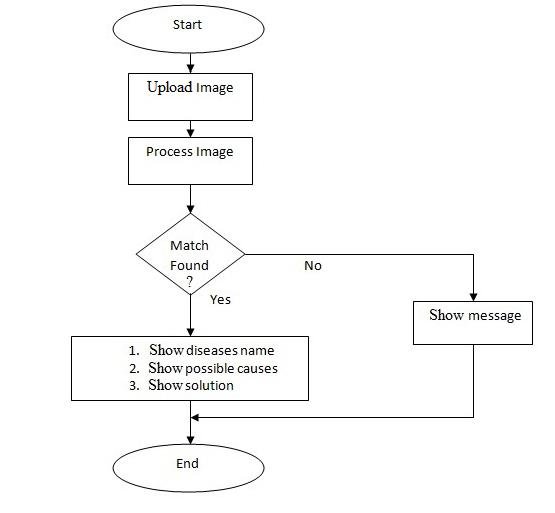
This project aims to develop a mobile app that can detect three common diseases of cotton crops, Bacterial Blight, Curl Virus, and Fusarium Wilt, using computer vision techniques. The app will allow users to take a picture of a cotton plant leaf or stem and analyze it to determine if it is infected with any of these diseases.

The app will have a user-friendly interface that guides the user through the process of taking a photo of the cotton plant and submitting it for analysis. The app will be connected to a server though API’s where AI trained model will be stored. After taking the image from user, it will send it back to server where AI trained model will make prediction and that prediction will be displayed to user.

The app will also have a database that stores information about the diseases, including symptoms. The app will provide users with detailed information about the diseases.

The app will be developed for the Android operating system, using the Java programming language and the Android SDK. The app will also make use of third-party computer vision libraries and APIs to perform image processing and analysis.

The development person will follow software design principles and best practices to ensure the app is reliable, secure, and scalable. The person will also implement testing and quality assurance procedures to ensure the app meets the requirements and functions as intended.

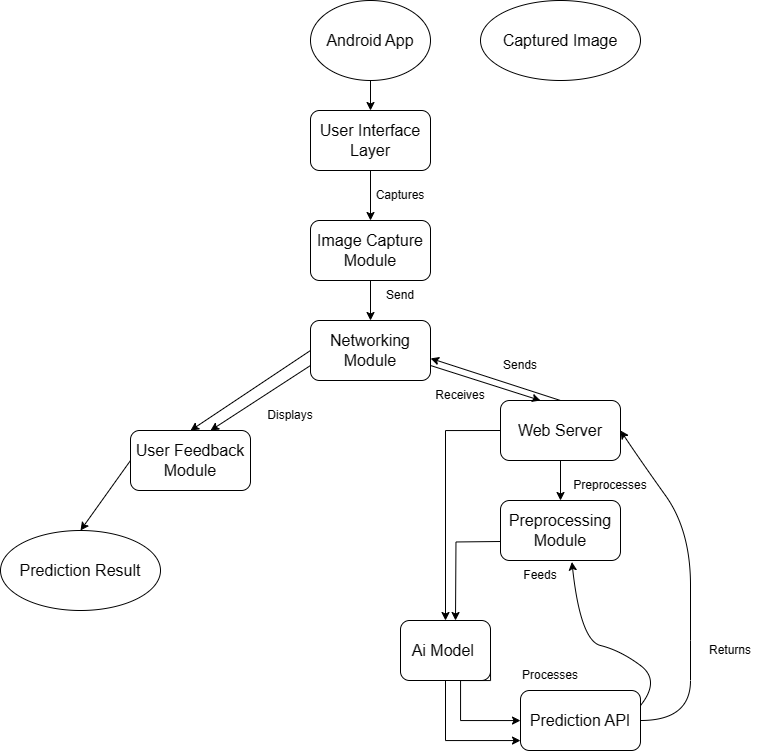


**Fig 2.1 System Overview Diagram**

# 3. SYSTEM ARCHITECTURE

## 3.1 Architectural Design

1. Client-Side (Android App) Architecture:
   * User Interface (UI) Layer: This layer includes the screens, views, and components that interact with the user. It captures the image using the device's camera and displays the prediction results.
   * Image Capture Module: This module handles the functionality related to capturing images using the device's camera and preparing them for transmission to the server.
   * Networking Module: This module manages the communication with the server, including sending the captured image and receiving the prediction results.
   * User Feedback Module: This module provides feedback to the user, displaying progress indicators and notifications during image capture and prediction processing.
2. Server-Side Architecture:
   * Web Server: The server hosts the AI trained model and exposes it through a RESTful API.
   * AI Model: The trained model is responsible for processing the received images and predicting the presence of cotton crop diseases.
   * Prediction API: This API receives the captured images from the Android app, passes them to the AI model for prediction, and returns the prediction results back to the app.
   * Preprocessing Module: This module preprocesses the images received from the Android app before feeding them to the AI model. It may include image resizing, normalization, and other necessary transformations.
3. Communication Flow:
   * The Android app captures an image of the cotton crop using the device's camera.
   * The captured image is sent from the app to the server's Prediction API through a network request.
   * The server preprocesses the received image and passes it to the AI model for prediction.
   * The AI model processes the image and generates a prediction result indicating the presence or absence of cotton crop diseases.
   * The server sends the prediction result back to the Android app.
   * The app receives the prediction result and displays it to the user.



**Fig 3.1 Architecture Design Diagram**

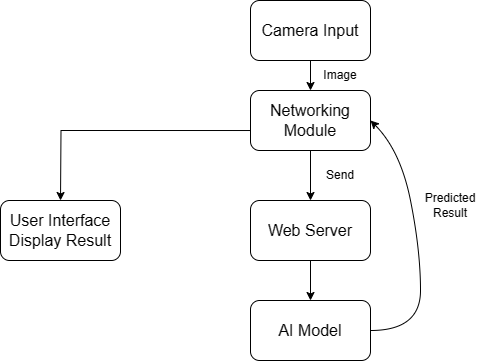
## 3.2 Decomposition Description

Decomposition is the process of breaking down a system into smaller, more manageable parts. In the case of this project, the system has been decomposed into three modules as described in section 3.1:

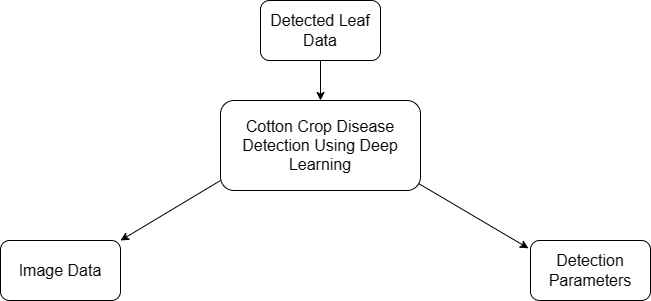
1. Client-Side (Android App) Decomposition Description:
   * User Interface (UI) Layer: This layer consists of the screens, views, and components responsible for capturing user input, displaying images, and showing prediction results.
   * Image Capture Module: This module handles the functionality related to accessing the device's camera, capturing images, and preparing them for transmission to the server.
   * Networking Module: This module manages the communication between the Android app and the server. It handles the transmission of captured images and the reception of prediction results.
   * User Feedback Module: This module provides visual feedback to the user, such as progress indicators and notifications, during image capture and prediction processing.
2. Server-Side Decomposition Description:
   * Web Server: This component hosts the AI trained model and exposes a RESTful API endpoint for receiving image data from the Android app.
   * AI Model: The AI model component processes the received images and makes predictions on the presence of cotton crop diseases based on the trained model's algorithms.
   * Prediction API: This API component receives the captured images from the Android app, performs any necessary preprocessing, passes them to the AI model for prediction, and returns the prediction results back to the app.
   * Preprocessing Module: This module preprocesses the received images, including resizing, normalization, and other transformations, to ensure compatibility with the AI model's requirements.
3. Communication Flow Decomposition Description:
   * Image Capture: The Android app captures images using the device's camera and prepares them for transmission.
   * Image Transmission: The captured images are sent from the Android app to the server's Prediction API through a network request using appropriate protocols and data formats.
   * Preprocessing: The server's Preprocessing Module performs any necessary image preprocessing to ensure optimal input for the AI model.
   * Prediction: The AI model processes the preprocessed images and generates prediction results based on the presence or absence of cotton crop diseases.
   * Result Transmission: The prediction results are sent back from the server to the Android app through a network response.
   * Result Presentation: The Android app receives the prediction results and displays them to the user in an appropriate format.

## 3.2.1 Top Level DFD:

A top-level DFD provides a high-level view of data flow within the system. The below diagram shows the top level DFD for the Detection of Cotton Crop Diseases system.



**Fig 3.2.1.1 Top-Level DFD**



**Fig 3.2.1.2 Level 0 DFD**

## 3.2.2 Object Oriented Description:

An object-oriented system can be used to decompose the subsystems. This includes Class diagrams, Activity diagrams and Sequence diagrams. The Cotton Crop Disease detection system consists of following classes:  
**1. Class: AndroidApp**

* Description: Represents the Android application for cotton crop disease detection.
* Properties:
  + uiLayer: Instance of the UserInterfaceLayer class.
  + imageCaptureModule: Instance of the ImageCaptureModule class.
  + networkingModule: Instance of the NetworkingModule class.
  + userFeedbackModule: Instance of the UserFeedbackModule class.
* Methods:
  + startApp(): Initializes the Android app.
  + captureImage(): Invokes the image capture functionality.
  + processPredictionResult(result: PredictionResult): Displays the prediction result to the user.

**2. Class: UserInterfaceLayer**

* Description: Manages the user interface components and screens.
* Properties:
  + screens: Array of Screen objects.
* Methods:
  + displayPredictionResult(result: PredictionResult): Updates the UI to show the prediction result.

**3. Class: Screen**

* Description: Represents a screen in the Android app.
* Properties:
  + components: Array of UIComponent objects.
* Methods:
  + render(): Renders the screen with its UI components.

**4. Class: UIComponent**

* Description: Represents a UI component on a screen.
* Properties:
  + type: Type of the UI component (e.g., button, image view).
  + properties: Properties specific to the UI component (e.g., text for a button).
* Methods:
  + interact(): Handles user interactions with the UI component.

**5. Class: ImageCaptureModule**

* Description: Handles image capturing functionality.
* Methods:
  + captureImage(): Captures an image using the device's camera.
  + prepareImageForTransmission(image: Image): Prepares the captured image for transmission to the server.

**6. Class: NetworkingModule**

* Description: Manages communication with the server.
* Methods:
  + sendImage(image: Image): Sends the captured image to the server's Prediction API.
  + receivePredictionResult(result: PredictionResult): Receives the prediction result from the server.

**7. Class: UserFeedbackModule**

* Description: Provides visual feedback to the user during image capture and prediction processing.
* Methods:
  + displayProgressIndicator(): Shows a progress indicator during image capture and transmission.
  + displayNotification(message: string): Displays a notification to the user.

**8. Class: Server**

* Description: Represents the server hosting the AI trained model and Prediction API.
* Properties:
  + webServer: Instance of the WebServer class.
  + aiModel: Instance of the AIModel class.
  + predictionAPI: Instance of the PredictionAPI class.
  + preprocessingModule: Instance of the PreprocessingModule class.

**9. Class: WebServer**

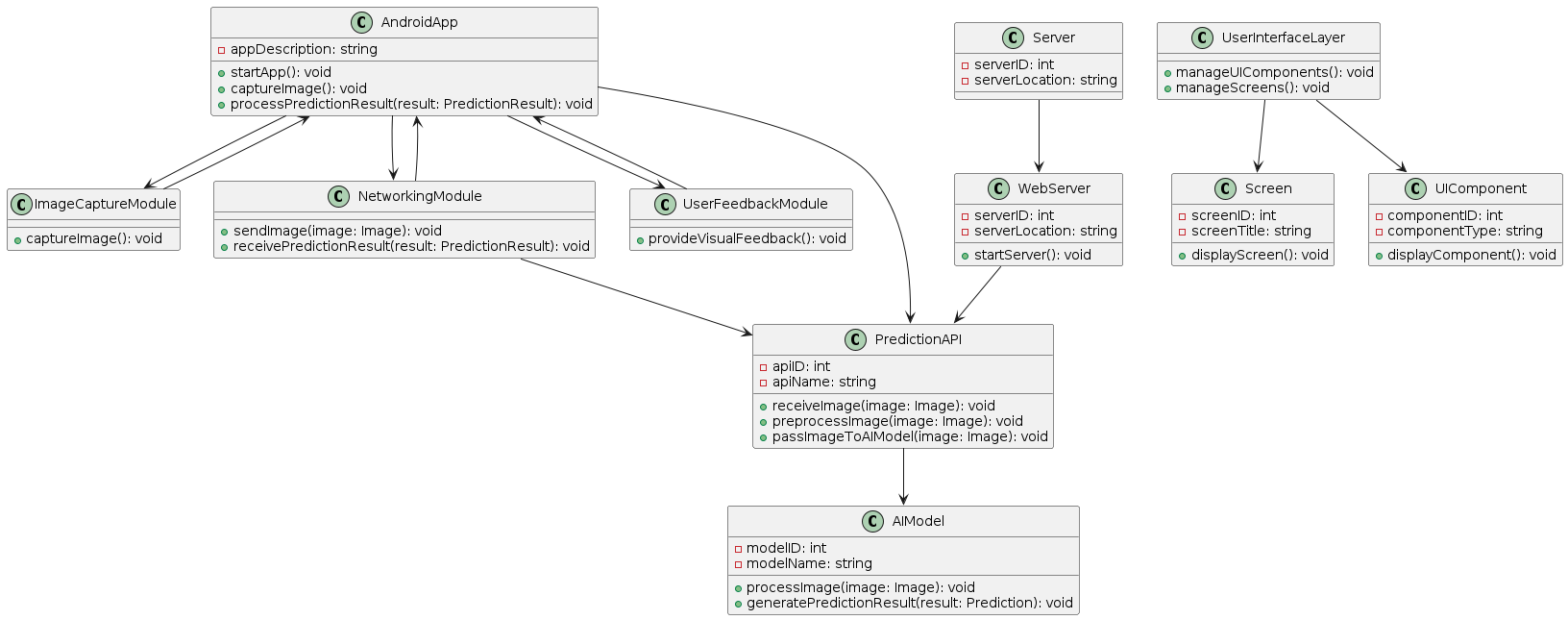
* Description: Hosts the AI trained model and exposes a RESTful API.
* Methods:
  + startServer(): Starts the web server.

**10. Class: AIModel**

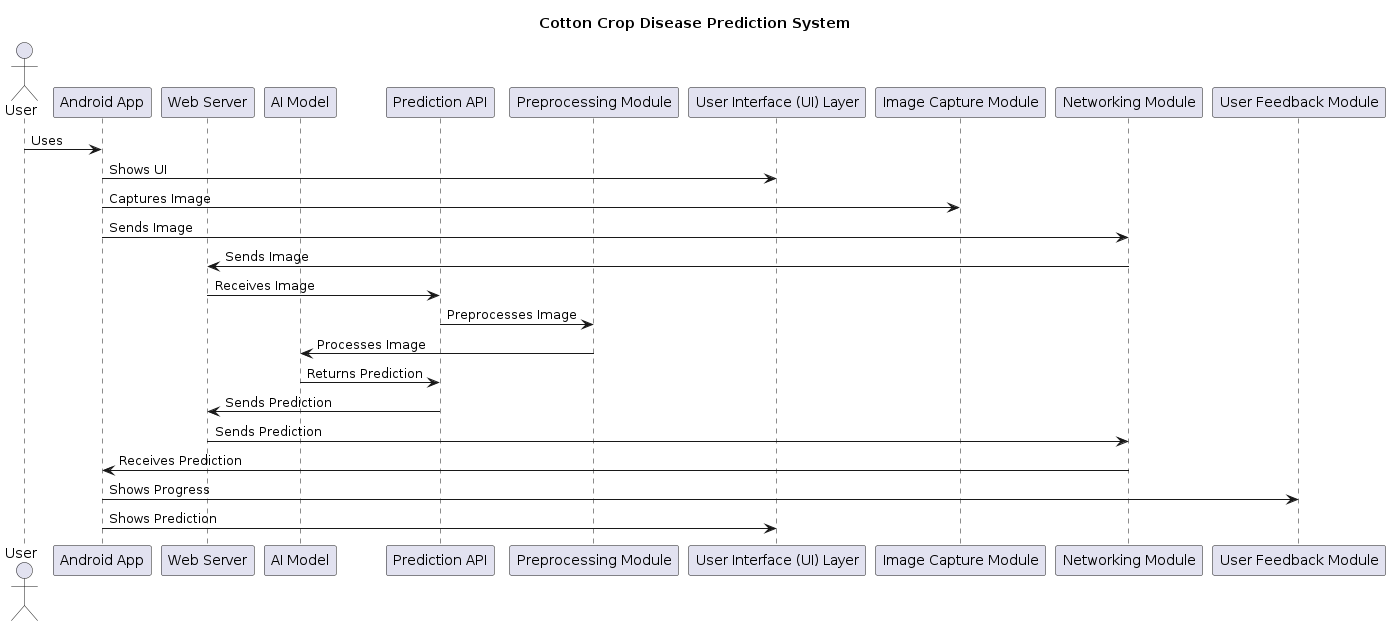
* Description: Processes the received images and makes predictions on cotton crop diseases.
* Methods:
  + processImage(image: Image): Processes the received image using the trained model.
  + generatePredictionResult(result: Prediction): Generates a prediction result based on the processed image.

**11. Class: PredictionAPI**

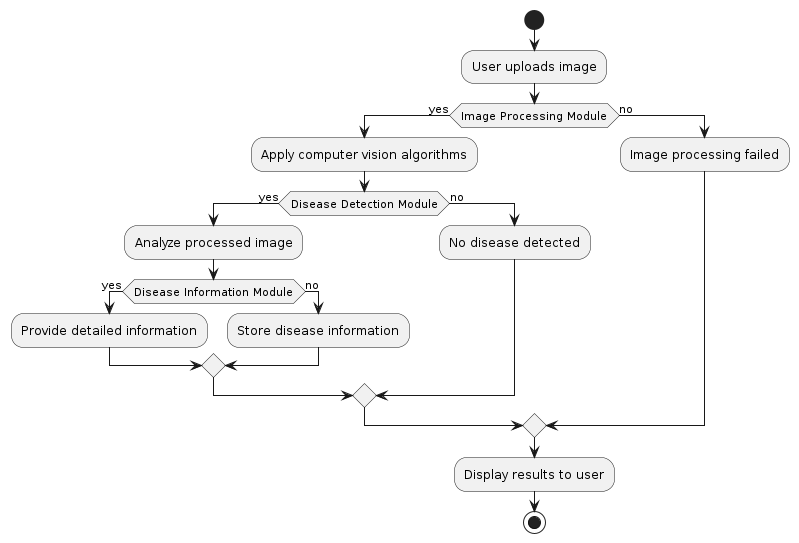
* Description: Receives captured images, performs preprocessing, and communicates with the AI model for prediction.
* Methods:
  + receiveImage(image: Image): Receives the captured image from the Android app.
  + preprocessImage(image: Image): Performs necessary preprocessing on the received image.
  + passImageToAIModel(image: Image): Passes the preprocessed image to the AI model



**Fig 3.2.2.1 Class Diagram**

****

**Fig 3.2.2.2 Sequence Diagram**

****

**Fig 3.2.2.3 Activity Diagram**

## 3.3 Design Rationale

1. Client-Side (Android App) Design Rationale:

* User Interface (UI) Layer: The UI layer is responsible for capturing user input, displaying images, and showing prediction results. It provides a user-friendly interface to interact with the app and view the disease detection outcomes.
* Image Capture Module: The image capture module enables the app to access the device's camera and capture images. It ensures that the images are properly prepared for transmission to the server for further processing.
* Networking Module: The networking module facilitates communication between the Android app and the server. It handles the transmission of captured images and the reception of prediction results, ensuring secure and efficient data transfer.
* User Feedback Module: The user feedback module enhances the user experience by providing visual feedback, such as progress indicators and notifications, during image capture and prediction processing. It keeps the user informed about the app's current state and any ongoing processes.

1. Server-Side Design Rationale:

* Web Server: Hosting the AI trained model on a web server allows for centralized access and management of the model. It provides scalability and accessibility to multiple clients simultaneously.
* AI Model: The AI model is responsible for processing the received images and making predictions on the presence of cotton crop diseases. It leverages computer vision algorithms and machine learning techniques to analyze the images and provide accurate results.
* Prediction API: The prediction API receives the captured images from the Android app and coordinates the interaction between the app and the AI model. It performs any necessary preprocessing on the images before passing them to the AI model for prediction.
* Preprocessing Module: The preprocessing module prepares the received images for the AI model by performing tasks such as resizing, normalization, and other transformations. It ensures that the images are in the appropriate format and quality for accurate disease detection.

1. Communication Flow Design Rationale:

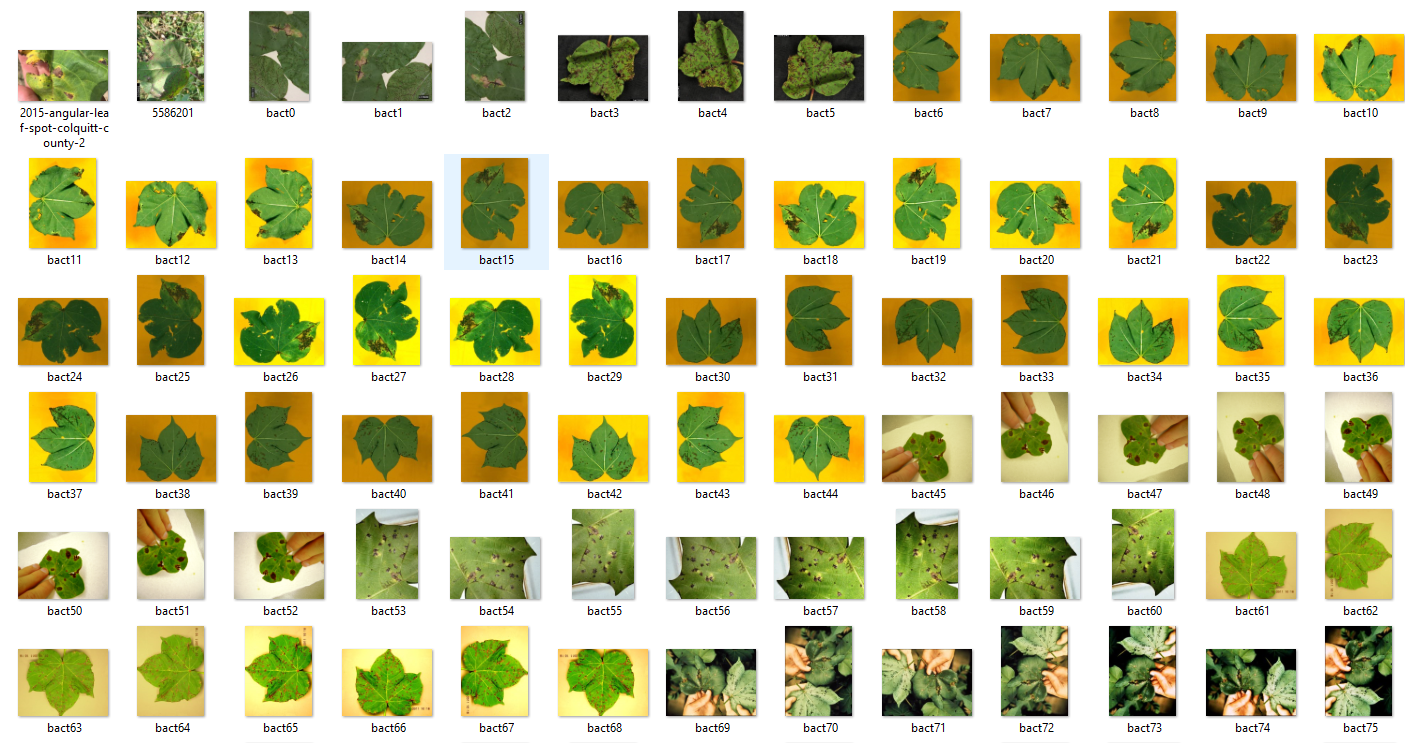
* Image Capture: Capturing images using the device's camera allows for real-time data acquisition and analysis. It provides a convenient way for users to capture images of cotton crops and receive instant disease detection results.
* Image Transmission: Transmitting the captured images from the Android app to the server's prediction API ensures that the images are securely and efficiently transferred for further processing. It enables remote processing of the images on a more powerful server infrastructure.
* Preprocessing: Preprocessing the received images on the server ensures that they meet the requirements of the AI model. It prepares the images for optimal analysis and increases the accuracy of disease detection.
* Prediction: Processing the preprocessed images using the AI model enables accurate prediction of the presence or absence of cotton crop diseases. The AI model's algorithms and training allow for effective analysis and classification of the images.
* Result Transmission: Sending the prediction results back from the server to the Android app ensures that the user receives the disease detection outcomes in a timely manner. It allows for immediate feedback and decision-making based on the results.
* Result Presentation: Presenting the prediction results to the user in an appropriate format enhances the user experience and facilitates understanding of the disease detection outcomes. It enables the user to take necessary actions based on the displayed results.

# 4. DATA DESIGN

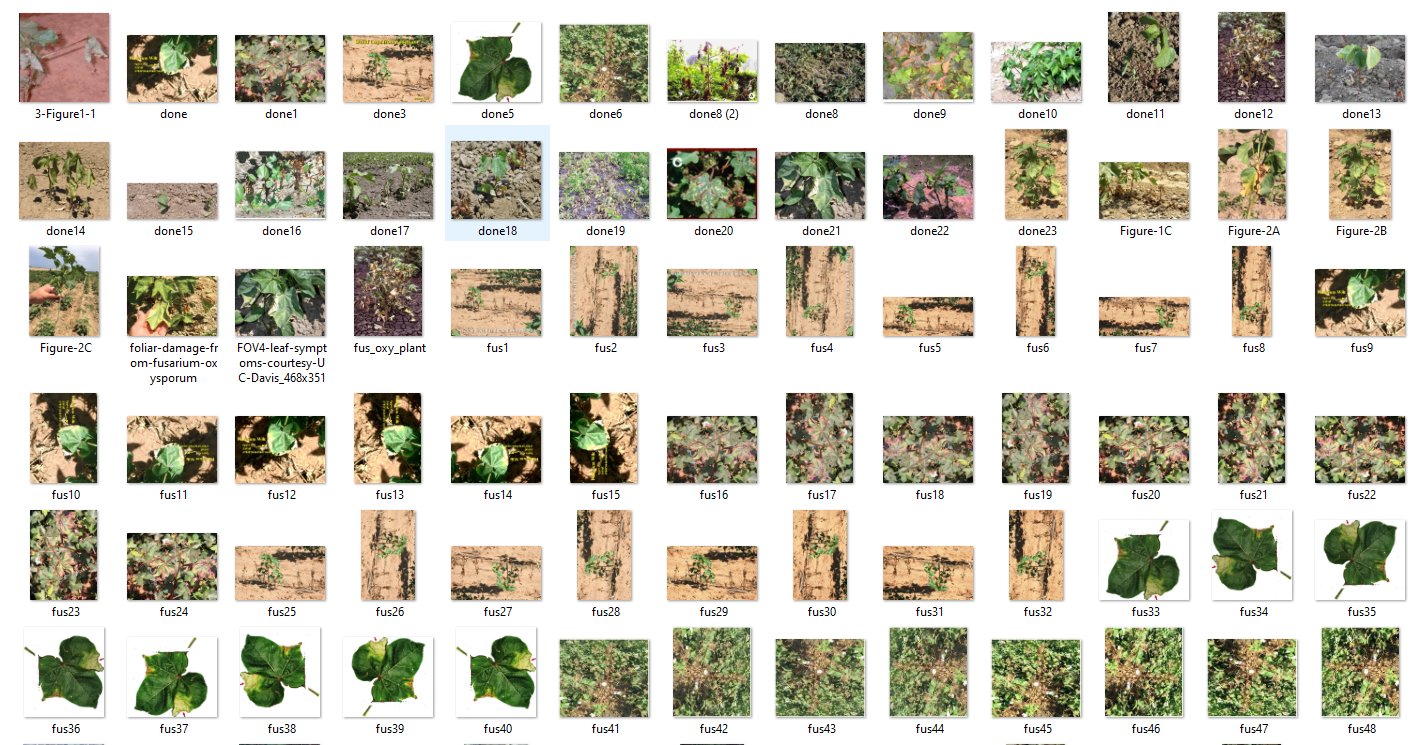
## 4.1 Data Description

The data description section of the Software Design Document (SDD) provides an overview of the data used in the project. In the case of Android app for cotton crop disease detection, the data description includes the following components:

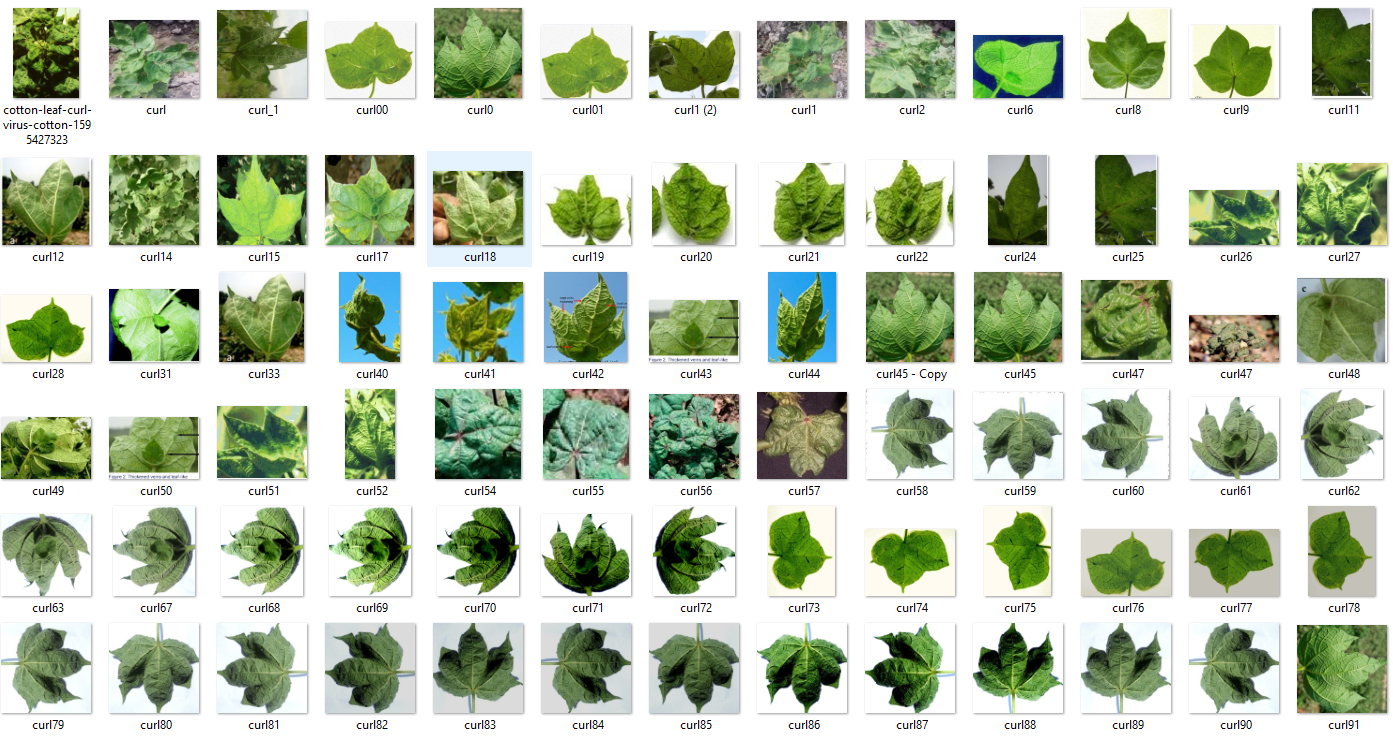
1. Image Data:
   * Input Images: The Android app captures images of cotton crops using the device's camera. These images serve as input for the disease detection process. They are in the form of digital image files, typically in formats such as JPEG or PNG.
   * Preprocessed Images: The captured images are transmitted to the server and undergo preprocessing before being passed to the AI model. The preprocessing module performs tasks such as image resizing, normalization, and other necessary transformations to ensure compatibility with the AI model's requirements.
2. Trained Model Data:
   * AI Trained Model: The server hosts the AI trained model, which is responsible for processing the received images and making predictions on the presence of cotton crop diseases. The trained model is developed using machine learning techniques and algorithms and is trained on a labeled dataset of images representing healthy cotton crops and those with bacterial blight, curl virus, and Fusarium wilt diseases.
3. Prediction Results:
   * Output Labels: The AI model generates prediction results in the form of labels that indicate the presence or absence of cotton crop diseases. These labels may include categories such as "Bacterial Blight," "Curl Virus," "Fusarium Wilt," or "Healthy." The labels are associated with the input images and serve as the basis for displaying the disease detection outcomes to the user.
4. Communication Data:
   * Network Requests and Responses: The Android app sends network requests to the server's Prediction API, transmitting the captured images for disease detection. The server processes the images, performs predictions using the AI model, and sends back the prediction results as network responses. These requests and responses contain relevant metadata such as headers, payloads, and response codes.



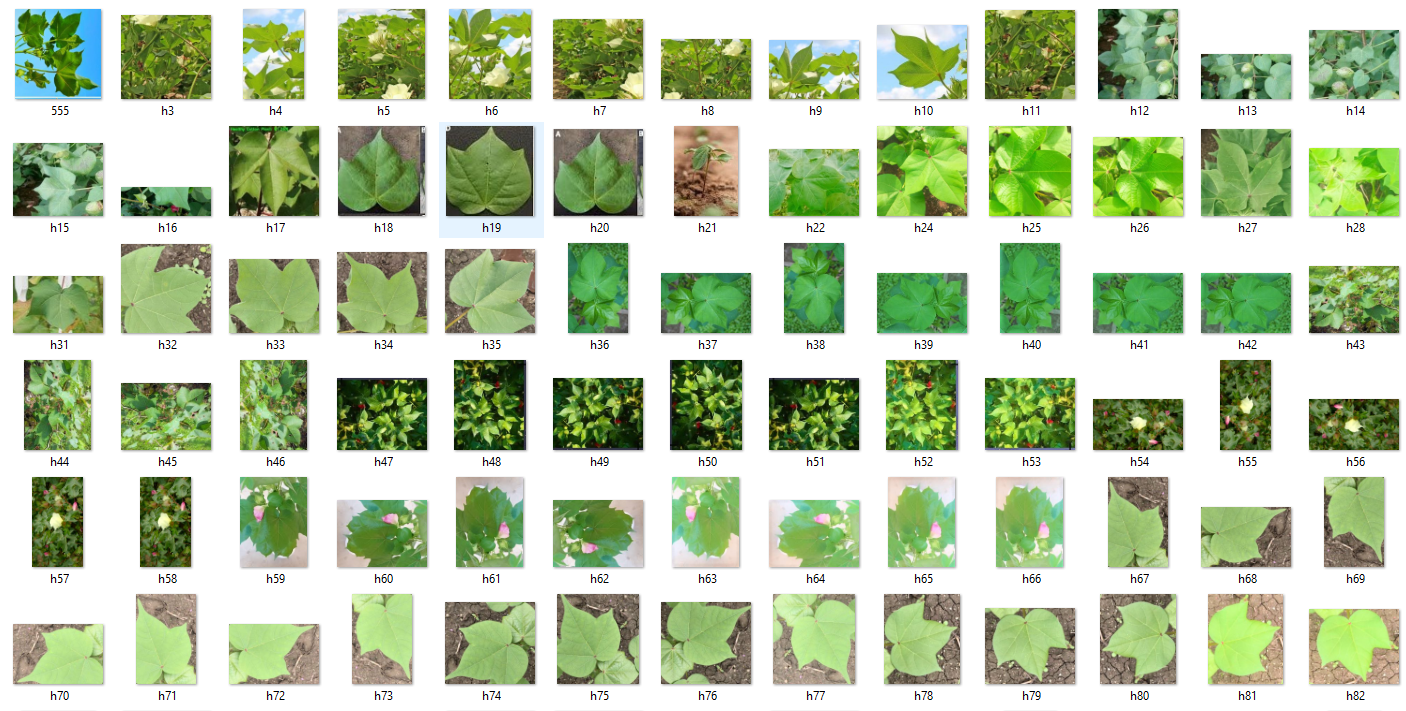
**Fig 4.1.1 Crops with Bacterial Blight**



**Fig 4.1.2 Crops with Fusarium Wilt**

****

**Fig 4.1.3 Crops with Curl Virus**

****

**Fig 4.1.4 Healthy Crops**

## 4.2 Data Dictionary

A data dictionary provides detailed information about the data elements used in a project. Based on the information provided in the system architecture and data description sections, here is a data dictionary for Android app's Software Design Document (SDD):

Data Dictionary:

1. Image Data:
   * Input Images:
     + File Name: [string] The name of the captured image file.
     + Format: [string] The format of the captured image file (e.g., JPEG, PNG).
   * Preprocessed Images:
     + File Name: [string] The name of the preprocessed image file.
     + Format: [string] The format of the preprocessed image file (e.g., JPEG, PNG).
2. Trained Model Data:
   * AI Trained Model:
     + Model File: [string] The file path or identifier of the AI trained model.
     + Model Version: [string] The version number or identifier of the trained model.
3. Prediction Results:
   * Output Labels:
     + Label: [string] The predicted label indicating the presence or absence of a cotton crop disease (e.g., "Bacterial Blight," "Curl Virus," "Fusarium Wilt," "Healthy").
4. Communication Data:
   * Network Requests:
     + Request Method: [string] The HTTP method used for the network request (e.g., POST).
     + Request URL: [string] The URL of the server's Prediction API endpoint.
     + Request Headers: [key-value pairs] Headers included in the network request.
     + Request Payload: [data] The captured image data sent as the payload in the network request.
   * Network Responses:
     + Response Code: [integer] The HTTP status code of the network response.
     + Response Headers: [key-value pairs] Headers included in the network response.
     + Response Payload: [data] The prediction results sent as the payload in the network response.

# COMPONENT DESIGN

1. User Interface (UI) Component:
   * Description: This component handles the screens, views, and components responsible for capturing user input, displaying images, and showing prediction results.
   * Responsibilities: Capturing user input, displaying images, presenting prediction results, and providing a user-friendly interface.
   * Relationships: Depends on the Image Capture, Networking, and User Feedback components.
2. Image Capture Component:
   * Description: This component handles the functionality related to accessing the device's camera, capturing images, and preparing them for transmission to the server.
   * Responsibilities: Accessing the device's camera, capturing high-quality images, and preparing captured images for transmission.
   * Relationships: Depends on the User Interface and Networking components.
3. Networking Component:
   * Description: This component manages the communication between the Android app and the server, including sending the captured image and receiving the prediction results.
   * Responsibilities: Handling network requests and responses, transmitting captured images, and receiving prediction results.
   * Relationships: Depends on the User Interface, Image Capture, and User Feedback components and interacts with the Prediction API component on the server side.
4. User Feedback Component:
   * Description: This component provides visual feedback to the user, such as progress indicators and notifications, during image capture and prediction processing.
   * Responsibilities: Displaying progress indicators, notifications, and other feedback to keep the user informed about ongoing processes.
   * Relationships: Depends on the User Interface component.
5. Prediction API Component (Server-Side):
   * Description: This component on the server side receives the captured images, performs preprocessing, passes them to the AI model for prediction, and returns the prediction results back to the app.
   * Responsibilities: Receiving captured images from the Android app, coordinating image preprocessing, forwarding images to the AI model, and returning prediction results to the app.
   * Relationships: Interacts with the Networking component on the Android app side, and interacts with the AI Model and Preprocessing Module components on the server side.

These component descriptions provide an overview of their responsibilities and relationships within the system.

# 6. HUMAN INTERFACE DESIGN

## 6.1 Overview of User Interface

From the user's perspective, the system will be a mobile app that allows them to easily detect diseases in their cotton crops using computer vision techniques. The app will have a user-friendly interface with clear and concise instructions for each step of the process.

To use the system, the user will need to register and create an account. Once registered, they can log in to the app and upload a picture of a cotton plant leaf or stem that they suspect may be infected with a disease. The app will then process the image using computer vision algorithms to identify any signs of bacterial blight, curl virus, or Fusarium wilt diseases.

The results of the image analysis will be displayed to the user, highlighting any areas of the image that show signs of disease. The app will also provide the user with detailed information about the type of disease detected.

If no signs of disease are detected in the image, the app will provide the user with feedback indicating that the crop appears healthy. The app will also allow the user to save the analyzed image and information for future reference.

Overall, the system's user interface will be intuitive and easy to use, allowing users to quickly and easily detect diseases in their cotton crops and take appropriate action to prevent further spread.

## 6.2 Screen Image



## 6.3 Screen Objects and Actions

1. Registration Screen:

* Screen Object: Registration form fields (e.g., username, email, password).
* Actions:
  + User enters registration details.
  + User submits the registration form.
  + Validation of entered data (e.g., checking for valid email format).
  + Display error messages if registration fails.

1. Login Screen:

* Screen Object: Login form fields (e.g., username/email, password).
* Actions:
  + User enters login credentials.
  + User submits the login form.
  + Validation of entered data.
  + Authentication and verification of user credentials.
  + Redirect to the Home Screen upon successful login.
  + Display error messages if login fails.

1. Account Update Screen:

* Screen Object: Account update form fields (e.g., username, email, password).
* Actions:
  + User edits account information.
  + User submits the updated account details.
  + Validation of entered data.
  + Update user account information in the backend/database.
  + Display success message upon successful account update.
  + Display error messages if the update fails.

1. Password Recovery Screen:

* Screen Object: Password recovery form fields (e.g., email).
* Actions:
  + User enters the registered email address.
  + User submits the password recovery form.
  + Validation of entered data.
  + Send a password recovery email to the registered email address.
  + Display a message informing the user to check their email for further instructions.
  + Display error messages if the recovery process fails.

1. Image Capture Screen:

* Screen Object: Camera viewfinder, capture button.
* Actions:
  + User captures an image of the cotton crop using the device's camera.
  + Display the captured image on the screen for user confirmation.

1. Prediction Result Screen:

* Screen Object: Text or visual representation of the disease prediction result.
* Actions:
  + Display the prediction result received from the server's API.
  + Provide appropriate feedback based on the predicted disease (e.g., "Bacterial Blight detected," "Healthy crop").
  + Allow the user to take further actions based on the prediction result (e.g., view more details, consult an expert).



**Fig 6.3 Screen objects and actions**

# REQUIREMENTS MATRIX

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Requirement ID | Requirement Description | UI Layer | Image Capture Module | Networking Module | User Feedback Module | Web Server | AI Model | Prediction API | Preprocessing Module |
| FR1 | Registration functionality |  |  |  |  |  |  |  |  |
| FR2 | Login functionality |  |  |  |  |  |  |  |  |
| FR3 | Account update |  |  |  |  |  |  |  |  |
| FR4 | Password Recovery |  |  |  |  |  |  |  |  |
| FR5 | Image capture using the device’s camera | X | X |  |  |  |  |  |  |
| FR6 | Communication with the server’s Prediction API |  |  | X |  | X |  | X |  |
| FR7 | User feedback during image capture and prediction processing | X |  | X | X |  |  |  |  |
| FR8 | Image preprocessing and segmentation |  |  |  |  |  | X |  | X |
| FR9 | Feature extraction from the segmented leaf image |  |  |  |  |  | X |  |  |
| FR10 | Classification of leaf image into disease categories |  |  |  |  |  | X |  |  |
| FR11 | Displaying the predicted diseases | X |  |  |  |  |  |  |  |
| FR12 | Handling Invalid Input Images | X | X |  |  |  |  |  |  |

The functional requirements from the SRS are listed in the leftmost column, and the system components are listed in the top row. The matrix indicates which components satisfy each functional requirement by placing an "X" in the corresponding cell. For example, the User Interface Module and the Database Module both satisfy FR1: Registration functionality, so there are "X" marks in the cells where those two intersect with FR1.