TEST PLAN FOR

AI Assisted Plant Disease Prediction,

Crop And Fertilizer Recommendation

ChangeLog

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made
001	30.10.2023	Harsh Srivastava	Initial Draft

1 Introduction

Our project focuses on the integration of ML into agriculture, specifically for plant disease prediction and crop/fertilizer recommendations. Leveraging the power of ML, we aim to enhance crop management by accurately identifying plant diseases and providing personalized advice on crop selection and fertilizer usage. This technology has the potential to revolutionize farming practices, offering efficiency and sustainability in an ever-evolving agricultural landscape.

1.1 Scope

1.1.1 In Scope

Scope defines the features, functional or non-functional requirements of the software that **will be** tested.Features of the Project:

1. **Image Upload and Recognition: Functional Requirements**
    - Evaluate the functionality of image uploading.
    - Implement image format verification and validation.
    - Ensure accurate recognition and identification of crop diseases and conditions.
2. **Detailed Descriptions: Non-Functional Requirements**
    - Verify the accurate provision of detailed descriptions for recognized crop diseases and recommended actions.
3. **Deep Learning Recognition: Functional Requirements**
    - Test the deep learning recognition algorithms for accuracy and reliability in disease identification.
4. **Machine Learning Model: Functional Requirements**
    - Validate the accuracy and performance of the machine learning model (e.g., ResNet-50) for crop disease prediction.
5. **User-Friendly Interface: Non-Functional Requirements**
    - Conduct usability testing to assess the user interface for its intuitiveness and ease of interaction for farmers and users.
6. **Cross-Platform Accessibility: Functional Requirements**
    - Test the platform's compatibility across various web browsers and devices to ensure accessibility.
7. **Reliable Cross-Browser Compatibility: Functional Requirements**
    - Ensure the web-based interface works seamlessly with popular web browsers to facilitate widespread accessibility.
8. **Early Stopping Mechanism: Functional Requirements**
    - Assess the effectiveness of the early stopping mechanism in the machine learning model to prevent overfitting and enhance prediction reliability.

1.1.2 Out of Scope

Out Of Scope defines the features, functional or non-functional requirements of the software that **will NOT be**
tested :

1. **Cloud Computation Offloading:**
    - Excluding the implementation of computational offloading to the cloud for performance enhancement on client devices. The project focuses on on-device processing.
2. **Scalability and Cloud Hosting:**
    - Omitting load testing to ensure the platform's efficiency in handling increased user loads when hosted on the cloud. The current scope does not involve cloud hosting.
3. **Custom Knowledge Database:**
    - Excluding the creation and validation of a custom knowledge database that provides detailed descriptions and context to users. The project relies on existing sources and algorithms for recommendations.

1.2 Quality Objective

Here make a mention of the overall objective that you plan to achieve without your

testingSome objectives of your testing project could be

- Ensure the Application Under Test conforms to functional and non-functional requirements
- Ensure the AUT meets the quality specifications defined by the client
- Bugs/issues are identified and fixed before go live

1.3 Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

- QA Analyst : Harsh Srivastava
- Test Manager : Prof. Shreela Pareek
- Configuration Manager: Prof. Neha Shukla
- Developers : Harsh Srivastava, Yash Srivastava , Sejal Gupta
- Installation Team : Prof. Shreela Pareek , Prof. Neha Shukla, Harsh Srivastava, Yash Srivastava , Sejal Gupta

2 Test Methodology

2.1 Overview

We are using an iterative testing approach to make sure our project works well. This means we test it in small steps, starting with checking if each part works on its own. Then, we see how different parts work together.

We keep testing as we make changes and add new things. This way, we make sure our project is always working well, even after modification.

2.2 Test Levels

Test Levels define the Types of Testing to be executed on the Application Under Test (AUT).

We aim to test our project at the following levels:

1) Unit Testing: This is the lowest level of testing and focuses on individual components or functions within the software. Developers often perform unit tests to verify that specific parts ofthe code work correctly.

2) Integration Testing: This level of testing checks how different components or modules of the software work together. It ensures that integrated parts of the software function as intended.

3) System Testing: At this level, the entire system is tested as a whole. It verifies that the softwaremeets its specified requirements and functions properly in its intended environment.

2.3 Test Completeness

Here you define the criterias that will deem your testing complete.
For instance, a few criteria to check Test Completeness would be

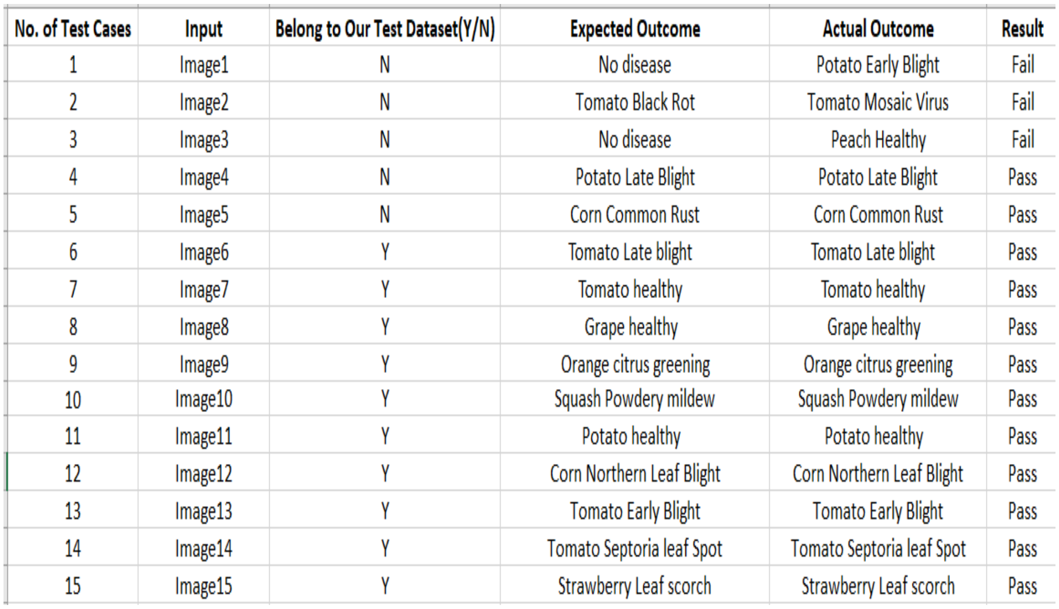
- 80% test coverage
- All Manual Test cases executed
- All open bugs are fixed or will be fixed in next release

3 Test Deliverables

Here are the deliverables

- Test Plan
- Test Cases
- Bug Reports
- Test Strategy

4 Test Cases



No. of Test Cases	Input	Belong to Our Test Dataset(Y/N)	Expected Outcome	Actual Outcome	Result
1	Image1	N	No disease	Potato Early Blight	Fail
2	Image2	N	Tomato Black Rot	Tomato Mosaic Virus	Fail
3	Image3	N	No disease	Peach Healthy	Fail
4	Image4	N	Potato Late Blight	Potato Late Blight	Pass
5	Image5	N	Corn Common Rust	Corn Common Rust	Pass
6	Image6	Y	Tomato Late blight	Tomato Late blight	Pass
7	Image7	Y	Tomato healthy	Tomato healthy	Pass
8	Image8	Y	Grape healthy	Grape healthy	Pass
9	Image9	Y	Orange citrus greening	Orange citrus greening	Pass
10	Image10	Y	Squash Powdery mildew	Squash Powdery mildew	Pass
11	Image11	Y	Potato healthy	Potato healthy	Pass
12	Image12	Y	Corn Northern Leaf Blight	Corn Northern Leaf Blight	Pass
13	Image13	Y	Tomato Early Blight	Tomato Early Blight	Pass
14	Image14	Y	Tomato Septoria leaf Spot	Tomato Septoria leaf Spot	Pass
15	Image15	Y	Strawberry Leaf scorch	Strawberry Leaf scorch	Pass

Drive Link -
https://docs.google.com/spreadsheets/d/1ib222sDEbL6BsOts9asL_j4mrallvIlj/edit?usp=sharing&ouid=106948280387447911264&rtpof=true&sd=true

Testing Conditions –

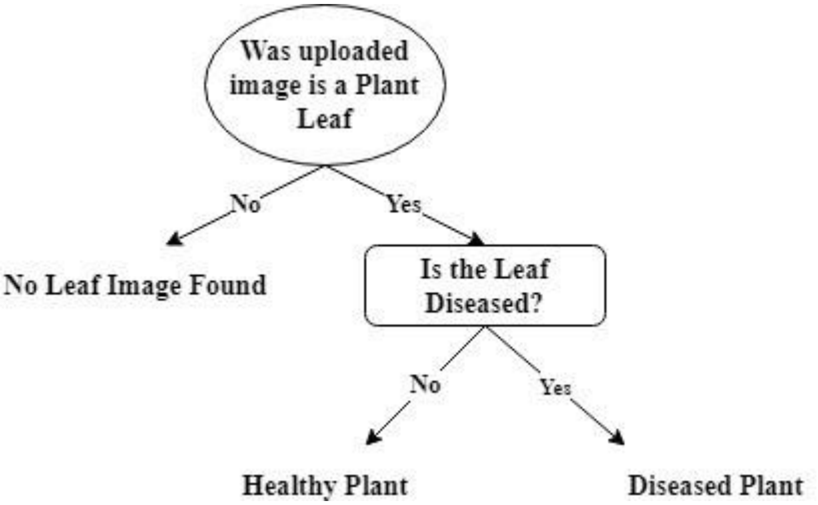
Equivalence Class Partitioning based on Number of Input Images:

Equivalence Class	Partition 0	Partition 1	Partition 2
No. of Input Image	0	1	>1
Result	Invalid	Valid	Invalid

Equivalence Class Partitioning based on Input File Type:

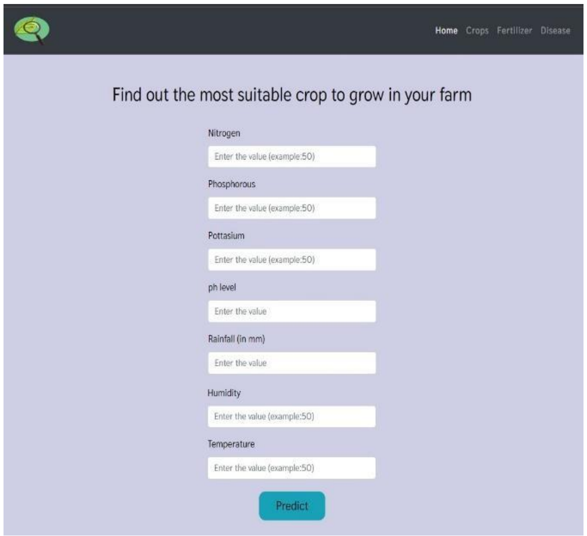
Equivalence Class	Partition 0	Partition 1
Input File Type	.jpeg/.jpg	.gif / .csv / .pdf
result	Valid	Invalid

Decision Tree



Regression Testing

Crop Recommendation System: In this we had to manually provide the temperature and humidity of a place in order to get crop recommendation, but now using open weather api we have to select the state and city to fetch the details of temp. and humidity of that place.



Find out the most suitable crop to grow in your farm

Nitrogen

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

Enter the value (example:50)

ph level

Enter the value

Rainfall (in mm)

Enter the value

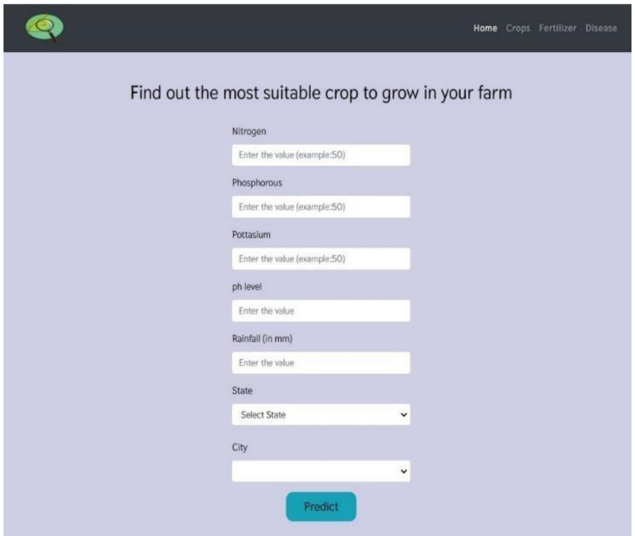
Humidity

Enter the value (example:50)

Temperature

Enter the value (example:50)

Before



Find out the most suitable crop to grow in your farm

Nitrogen

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

Enter the value (example:50)

ph level

Enter the value

Rainfall (in mm)

Enter the value

State

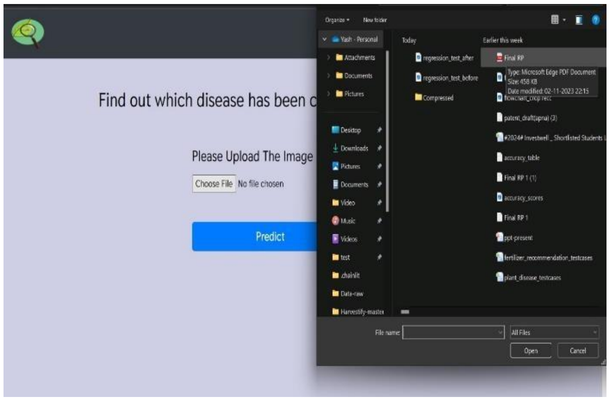
Select State

City

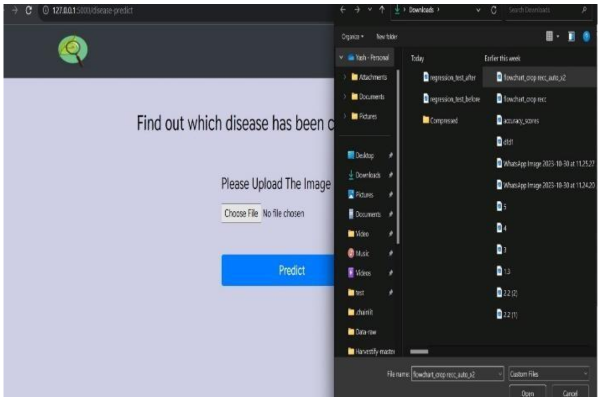
After

Disease Prediction Model:

Earlier our model was taking all kinds of files like .jpg,.png,.pdf,.docx but we only wanted images , now it will take only images as input and not any files other than images.



Before



After

5.Resource & Environment Needs

4.1 Testing Tools

We have performed manual testing:

- **Black Box Testing** – Boundary Value Analysis, Equivalance Class Partitioning, Decision Tree and Regression Testing.
- **White Box Testing** – Unit and Integration Testing.

4.2 Test Environment

It mentions the minimum **hardware** requirements that will be used to test the

Application.Following **software's** are required in addition to client-specific

software.

- Windows 10 and above preferred
- VSCode 2022 or above preferred
- Chrome, Mozilla or Edge Preferred over non-chromium based browsers

5 Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test