Smart fields: Enhancing Agricultural Sustainability with Technology

A Mini Project Report Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND

ENGINEERING

Team members:

Mr. Mohith (21071A05E0)

Mr. Mahimood Pasha (21071A05J1)

Ms. Varshitha (21071A05J5)

Mr. Sandeep (22075A0518)

Under the guidance of:

Dr. P. Siva Kumar

 (Associate Professor, Department of CSE, VNR VJIET)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with A++ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, AIML, IT B. Tech Courses Approved by AICTE, New

Delhi, Affiliated to JNTUH Recognized as a College with Potential for Excellence a by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad 500090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with A++ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, AIML, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as â College with Potential for Excellence by UGC ISO 9001:2015 Certified, QSI GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet, Hyderabad â 500 090, TS, India

 DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CERTIFICATE**

This is to certify that the project report entitled “SMART FIELDS: ENHANCING AGRICULTURAL SUSTAINABILITY WITH TECHNOLOGY" is a bonafide work done under our supervision and is being submitted by **Mr. Chekuri Mohit(21071A05E0), Mr. Mahimood Pasha (21071A05J1), Ms. Thurpu Varshitha (21071A05J5), Mr. Jangamgari Sandeep Kumar(22075A0518)** in partial fulfilment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, of the VNRVJIET, Hyderabad during the academic year 2022-2023. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

|  |  |
| --- | --- |
| Dr. P. Sivakumar  Associate Professor  CSE Department,  VNR VJIET | Dr. V Baby  HOD CSE Department  CSE Department,  VNR VJIET |

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with A++ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, AIML, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as College with Potential for Excellence by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet, Hyderabad â 500090, TS, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



**DECLARATION**

We declare that the major project work entitled “**Smart fields: Enhancing Agricultural Sustainability with Technology**” submitted in the department of Computer Science and Engineering, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide record of our own work carried out under the supervision of Mrs. P. Prasanna, Assistant Professor, Department of CSE, VNRVJIET. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously

Place: Hyderabad

|  |  |  |  |
| --- | --- | --- | --- |
| **Mr. Ch Mohit**  **(21071A05E0)** | **Mr.Mahimood Pasha**  **(21071A05J1)** | **Ms. T Varshitha**  **(21071A05J5)** | **Mr. J Sandeep Kumar**  **(22075A0518)** |

ACKNOWLEDGEMENTS

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, Dr. Challa Dhanunjaya Naidu, and our Head of Department, Dr. V. Baby, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, Dr. P. Sivakumar **Mrs. N. Sarika** and **Dr. P. Tejaswi** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Computer Science Engineering department and to our fellow classmates who directly or indirectly helped us.

**Mr.Chekuri Mohit (21071A05E0)**

**Mr. Mahmood Pasha (21071A05J1)**

**Mr. Thurpu Varshitha (21071A05J5)**

**Mr. Jangamgari Sandeep (22075A0518)**

ABSTRACT

The Smart Field Project is an innovative agricultural initiative designed to revolutionize crop management through disease prediction, crop recommendations, weather forecasting, and market pricing analysis. Tailoring disease prediction to specific crops, the system utilizes machine learning algorithms for real-time monitoring, analyzing indicators such as leaf patterns and growth anomalies. This precision enables early detection and targeted interventions, reducing crop losses and promoting sustainable farming practices. The incorporation of a sophisticated weather forecasting feature adds an additional layer of precision to the crop recommendation system, ensuring that farmers receive personalized guidance. In essence, the Smart Field Project offers a comprehensive and accessible tool for farmers to navigate the dynamic agricultural landscape. By integrating advanced technologies into disease management, crop selection, and market strategies, it represents a pivotal step towards informed decision-making and sustainable practices in modern agriculture.

**INDEX**

1. [Introduction](#_TOC_250059) 1
2. [Literature Survey/ Existing System](#_TOC_250058) 2
   1. [Literature Review](#_TOC_250052) 2
   2. [Existing System](#_TOC_250051) 2
   3. [Drawbacks Of the Existing System](#_TOC_250050) 3
3. [Software Requirement Analysis](#_TOC_250049) 4
   1. [Introduction](#_TOC_250048) 4
      1. [Document Purpose](#_TOC_250047) 4
      2. [Definitions](#_TOC_250046) 4
   2. [Feasibility Study](#_TOC_250057) 4
      1. [Operational Feasibility](#_TOC_250056) 5
      2. [Economic Feasibility](#_TOC_250055) 5
      3. [Technical Feasibility](#_TOC_250054) 5
   3. [System Architecture](#_TOC_250045) 5
   4. [Functional Requirements](#_TOC_250044) 6
   5. [System Analysis](#_TOC_250043) 7
   6. [Non-Functional Requirements](#_TOC_250042) 8
   7. [Software Requirement Specification](#_TOC_250041) 8
   8. [Software Requirements](#_TOC_250040) 8
   9. Hardware Requirements 8
4. [Software Design](#_TOC_250038)  9
   1. [UML Diagrams](#_TOC_250037)  9
      1. [Use Case Diagram](#_TOC_250036) 12
      2. [Sequence Diagram](#_TOC_250035) 13
      3. [Activity Diagram](#_TOC_250034) 18
      4. [Class Diagram](#_TOC_250033) 20
5. [Proposed System](#_TOC_250032) 21
   1. [Methodology](#_TOC_250031) 21
      1. [Overall Framework](#_TOC_250030) 21
      2. Modules 22
      3. Functionalities 22
   2. [Advantages Of Proposed System](#_TOC_250027) 23
6. Coding/Implementation 24
7. [Testing](#_TOC_250023) 30
   1. Types of Testing 30
      1. Manual Testing 30
      2. Automated Testing 30
      3. Functional Testing 31
         1. Integration Testing 31
         2. Unit Testing 31
8. [Results](#_TOC_250001)  34
   1. User Interface – Contents 34
   2. User Interface – Leaf Disease Predictor 34
   3. Prediction Result 35
   4. Recommendation 35
   5. Weather Forecast 36
   6. Features and About 36
9. [Conclusion And Further Work](#_TOC_250000) 37
10. References 38

**List of Figures**

**Figure Page No**

Fig 3.3.1: System Architecture 6

Fig 3.5: System Analysis 7

Fig 4.1.1.1 Use Case Representation 10

Fig. 4.1.1.2 Use Case Diagram for application 12

Fig. 4.1.2.1 Sequence Diagram for plant disease prediction 16

Fig. 4.1.2.1 Sequence Diagram for crop recommendation 16

Fig. 4.1.3.1 Activity Diagram for application 18

Fig. 4.1.4.1 Class Diagram for application 20

**List of Tables**

**Table Page No**

Table. 4.1.2.1 Sequence Diagram Symbols 15

# **1.INTRODUCTION**

The Smart Field Project stands as a groundbreaking initiative in the quest to revolutionize global agriculture, addressing the critical challenges of crop disease identification, soil suitability, and resource optimization. In many regions of the world, the timely and accurate identification of crop diseases remains an elusive goal, primarily due to inadequate infrastructure and limited access to modern diagnostic tools. This issue poses a significant threat to food security, as delayed or incorrect diagnosis can lead to widespread crop failures and substantial economic losses for farmers.

Moreover, the ability to determine which crops are best suited to specific soil types is essential for maximizing agricultural productivity and ensuring sustainable farming practices. With climate change, soil degradation, and other environmental factors constantly altering the landscape of agriculture, traditional methods of crop selection are increasingly insufficient. Farmers need precise, data-driven insights to make informed decisions about what to plant, when to plant, and how to manage their fields effectively.

The Smart Field Project addresses these challenges by harnessing the power of advanced machine learning algorithms and data analytics. Through real-time disease prediction, the project enables farmers to detect potential threats to their crops at an early stage, allowing for timely intervention and minimizing losses. The system’s personalized crop recommendations are tailored to the specific characteristics of the soil, climate, and other environmental factors, ensuring that farmers can choose the most suitable crops for their land.

In addition to disease prediction and crop recommendations, the Smart Field Project provides accurate weather forecasting and market pricing analysis, further empowering farmers to make strategic decisions. By anticipating weather patterns and market trends, farmers can optimize their planting schedules, manage resources more efficiently, and maximize their profits.

Overall, the Smart Field Project represents a transformative tool in modern agriculture, offering a holistic approach to crop management that integrates technology, data, and sustainability. By providing farmers with actionable insights and empowering them to navigate the complexities of today’s agricultural landscape, the project paves the way for a more prosperous, resilient, and food-secure future.

**2. LITERATURE SURVEY/ EXISTING SYSTEM**

**2.1 LITERATURE REVIEW**

In recent years, significant advancements have been made in leveraging machine learning and deep learning techniques for agricultural applications, particularly in crop recommendation and plant disease prediction.

For crop recommendation, studies have focused on developing robust frameworks that utilize sensor data and environmental features to recommend suitable crops for specific regions. These frameworks often employ machine learning algorithms such as decision trees, support vector machines, and recurrent neural networks. Pros include improved crop selection and increased yield and profitability. However, challenges such as scalability, data privacy, and implementation complexity remain areas of concern.

Similarly, in plant disease prediction, researchers have explored various methodologies, including traditional machine learning algorithms and deep learning techniques like convolutional neural networks (CNNs). These studies aim to detect and diagnose plant diseases accurately and efficiently using techniques such as spectral imaging, Internet of Things (IoT)-based systems, and spectral analysis. Pros include accurate and timely disease detection, supporting crop protection and yield optimization. However, challenges such as limited labelled data, model interpretability, and deployment in real-world agricultural settings persist.

Overall, these combined efforts highlight the growing importance of data-driven approaches in agriculture and the potential of machine learning and deep learning techniques to address critical challenges in crop management and plant health. While these approaches offer promising solutions, further research is needed to overcome existing limitations and ensure the practicality and effectiveness of these systems in supporting sustainable agricultural practices and food security.

## **2.2 EXISTING SYSTEM**

* Limited by the farmer's experience and observation skills, relies on cultural knowledge and experience, which may not always be optimal or scientifically validated.
* Highly subjective and dependent on the farmer's ability to recognize symptoms accurately. Some diseases may not exhibit visible symptoms until they have already caused significant damage.
* The effectiveness of traditional remedies may vary and may not provide consistent control of crop diseases. Lack of scientific validation and potential environmental or health risks associated with some traditional practices.
* Subject to biases and individual interpretations. Farmers may miss out on opportunities or make suboptimal decisions based on incomplete or outdated market knowledge.
* Limited access to market information, especially for farmers in remote or rural areas. Prices at local markets may not reflect broader market trends.

**2.3 DRAWBACKS OF THE EXISTING SYSTEM**

* Relies heavily on the availability and accessibility of agricultural experts. In areas with limited access to experts, farmers may not receive timely or accurate advice.
* Limited Dataset: The dataset being used is very less for a proper training of the classification models, the current dataset causes the models to overfit and the results get exaggerated.

# **SOFTWARE REQUIREMENT ANALYSIS**

# **3.1 INTRODUCTION**

# Requirement analysis comes after elicitation and is a crucial step. We review, enhance, and carefully scrutinise the gathered requirements in order to produce standard and unambiguous requirements. This exercise goes over every criterion and might display a graphic of the entire system.

### **3.1.1 DOCUMENT PURPOSE**

* Describe the different crop recommendation techniques employed in the model.
* Discuss anomaly detection, pattern recognition, machine learning, or other relevant techniques.
* Explain how these techniques are applied to identify fraudulent transactions

**3.1.2 DEFINITIONS**

**Machine Learning**

Machine learning is AI's subfield where computers learn from data, recognize patterns, and make predictions without explicit programming. It involves data collection, pre-processing, and model selection.

**Google Colab**

## Google Colab, short for Google Collaboratory, is an online cloud-based development environment provided by Google. It is designed to facilitate collaborative coding and machine learning experimentation. Colab allows users to write and execute Python code in a browser-based notebook interface, eliminating the need for local installations of development tools and libraries.

**Kaggle**

Kaggle offers a wide range of datasets, challenges, and resources for individuals and teams to explore, analyze, and build predictive models on real-world data. These competitions often involve solving complex problems and applying machine learning techniques to achieve the best possible predictive accuracy.

## **3.2 FEASIBILITY STUDY**

The feasibility study is a preliminary evaluation of a proposed project's viability and likelihood of success. It is carried out to ascertain whether a project is operationally, financially, and technically feasible. A feasibility study's primary goals are to assess a proposed project's advantages and disadvantages, pinpoint potential risks, and decide whether the project is worthwhile. The research takes into account elements like project objectives, resources on hand, technical specifications, market demand, legal restrictions, and financial ramifications.

Our Feasibility Study report includes:

* Operational Feasibility Study
* Economic Feasibility Study
* Technical Feasibility Study

### **3.2.1 OPERATIONAL FEASIBILITY**

Operational feasibility is an important consideration in any project and involves evaluating whether the project can be implemented and operated effectively within the existing organizational structure and processes. By creating a more accurate model, it reduces the negative aspects of the existing system.

Compared to alternative systems now on the market, the proposed solution benefits both the business and the customers more effectively. Reliability, maintainability, usefulness, accessibility, sustainability, and cost are design-dependent factors of the application.

### **3.2.2** **ECONOMIC FEASIBILITY**

Economic feasibility is an important consideration in any project and involves evaluating whether the project can be completed within the available budget and whether the project is financially viable.

This project was built using minimum resources like 16 GB Ram and Intel i5 Processor. Colab provides free access to enabling GPU to speed up the training process. Users can use our application without any investment, it's free of cost.

### **3.2.3 TECHNICAL FEASIBILITY**

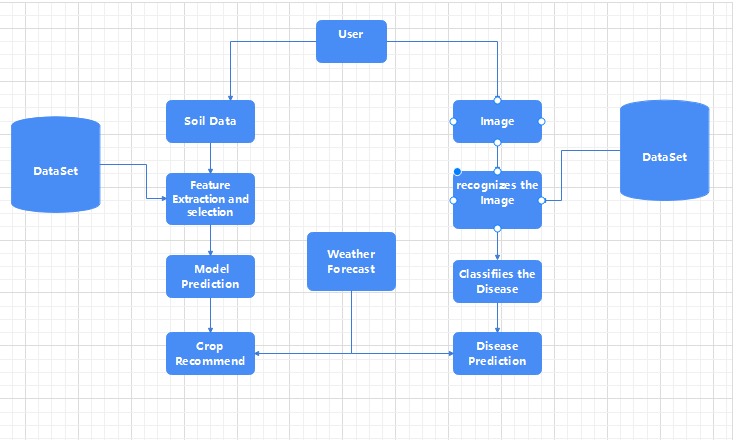
Evaluation of the proposed system's hardware, software, and other technical requirements is referred to as technical feasibility.

A website called Kaggle provided the dataset from which we constructed the model. The largest data science community in the world, Kaggle offers you strong tools and resources to assist you in achieving your data science objectives. The dataset pre-processing is another aspect of this research.

The Python code can be run in a Kaggle kernel or a Google Collab (I Python notebook). The posted data is extracted to create the best models for predicting and describing the data on the Kaggle platform for predictive modelling and analytics competitions.

**3.3 SYSTEM ARCHITECTURE**

It is a conceptual model that explains the organisation, perspectives, and actions of a system. An architecture description is a formal description and representation of a system that is set up so that it is easy to reason about the structures and behaviours of the system. A system architecture is made up of built sub-systems and system components that work together to implement the whole system.



*Fig 3.3.1: System Architecture*

The system integrates soil data and image recognition to assist farmers in selecting suitable crops and diagnosing plant diseases. It processes soil characteristics through feature extraction and model prediction, offering crop recommendations based on current soil conditions and weather forecasts. Simultaneously, it analyzes plant images using machine learning for disease classification, providing insights into plant health issues. Ultimately, this system aims to enhance agricultural decision-making by combining soil analysis, image diagnostics, and weather factors.

**3.4 FUNCTIONAL REQUIREMENTS**

* The system would need to be able to process incoming data in real-time
* The system would need to include machine learning models like classification algorithms, supervised algorithms, ensemble methods, random forest, xgboost etc.
* The system would need to provide users with clear and informative visualizations for better understating of underlined pattern in dataset.

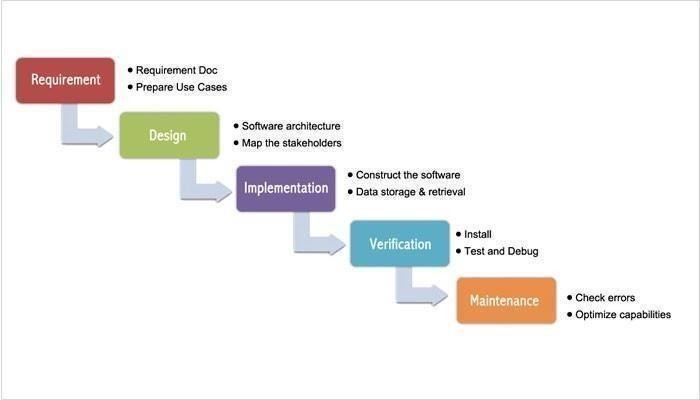
**3.5 SYSTEM ANALYSIS**

Fig 3.5.1: System Analysis

It was the initial Process Model that was presented. It is also known as a life cycle model that is linearly sequential. It's quite easy to use and comprehend. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The first SDLC approach used during software development was the waterfall model.

The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap.

The steps in the waterfall model are explained below.

**Requirements:** The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented.

**Design:** This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

**Implementation:** The design was converted into a machine-readable format in order for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.

**Verification:** It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply to the previously specified requirements.

**Maintenance:** Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.

**Useful factors:** The Waterfall model has its advantages like it is simple to use. Additionally, while using the model all the system requirements can be defined as a whole, explicitly and at the start the product can run without many issues.

It is economic to make changes in the early stages of the project when there are problems with system requirements then when the problems which arise in later stages.

## **3.6 NON-FUNCTIONAL REQUIREMENTS**

**Ease of Use**

The system is simple, user friendly.

**Extensibility**

The system can be easily extended to incorporate additional functionality.

**Maintainability**

The system will be as self-contained as possible to allow for ongoing maintenance.

**Reliability**

The system is reliable because of the advanced technology that is used to develop the system, the system can achieve accuracy up to 97 % to detect plant disease and crop recommendation.

**3.7 SOFTWARE REQUIREMENT SPECIFICATION**

**GOOGLE COLAB**

Google Colab is a cloud-based platform provided by Google that allows users to write, execute, and collaborate on Python code using Jupyter Notebook environment. Colab provides access to GPUs and TPUs, which can significantly accelerate machine learning and deep learning tasks. It supports popular libraries like TensorFlow and PyTorch, allowing users to build and train models efficiently.

**3.8 SOFTWARE REQUIREMENTS**

* + Windows 7/8/10/11
  + Google Colab
  + Python v3.10

## **3.9 HARDWARE REQUIREMENTS**

* + - SSD 256GB (minimum)
    - RAM 8GB (minimum)
    - GPU (Up to 12GB, supported by Google Colab)
    - Processor Core i5/i7

# **4. SOFTWARE DESIGN**

## **4.1 UML DIAGRAMS**

The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. The system is presented from a different point of view in a UML context.

These components are similar to modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

**User Model View**

The client’s point of view is referred to as the perspective. The exam depiction depicts a situation of utilization. The systems operation is defined in light of the user and what they want from it, with the user view providing a window into the system from the perspective of the user.

**Structural model view**

The details of the device are represented in this layout. Rephrase the static structures are mapped out in this software design. Sequence diagrams and state machine diagrams are included in this view.

**Behavioral Model View**

The social dynamics are depicted in the client model and basic model view. The system dynamics and how they interact are illustrated in the Behavioral Diagrams. The behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagram.

**Implementation Model View**

The implementation view is also referred to. Rephrase the diagram describes the system components. The package diagram is a diagram used to show the development view.

**Environmental Model View**

System and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

* The client model and auxiliary model perspectives were the focus of the demonstration.
* The presentation focuses on demos, usage, and natural model perspectives.

### **USE CASE DIAGRAM**

The dynamic nature of a system is depicted in a use case diagram. There is a purpose that distinguishes it from the other diagrams.

Use case diagrams are used to collect the needs of the system. Design specifications are the majority of them. When examining a system to gather its functions, use cases are constructed and actors identified.

Once the primary task is completed, use case diagrams are used to represent the outside view. Use case diagrams are useful. It is:

* Used to collect requirements.
* Used to get a birds-eye view of a system.
* The system is being influenced by various factors.
* The requirements should be displayed as actors.

The use case diagrams are used to analyze the systems requirements. When the requirements are examined, the system's function is recorded in use cases. There are use cases that are written in a logical order. The actors are important in use cases. The system has any entities that interact with it referred to as actors.

Internal applications, human users and external applications can all be actors. The following factors should be kept in mind when constructing a use case diagram.

* + - * As a use case, functionalities will be represented.
      * Actors.
      * Relationships between use cases and actors.

**Use Cases**

A use case is a depiction of how visitors will do things. The way a system responds to a request is defined from the standpoint of a user. Each use case has a sequence of basic actions that start with the user’s goal and end when they achieve it.

**Graphical Representation**

The below given shape is used to represent use case.

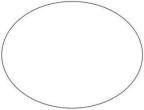


Figure 4.1. SOFTWARE DESIGN‑1.1.1Use Case Representation

The following is a more precise analysis of a use case:

* The system displays a pattern of behavior.
* An actor performs a series of transactions with the system.
* Something useful can be delivered to the actor.

To document system requirements, connect with top users and test the system, you can use use cases. The best way to uncover use cases is to look at the actors and see what they can do with the system.

**Flow of events**

A sequence of times can be thought of as a collection of interactions (or opportunities) carried out by the system. They provide daily point-by-point details, published in terms of what the framework can do rather than whether the framework performs the task.

* + - * When and how the employment case begins and ends.
      * Interactions between the use case and the actor.
      * Information required by the employment case.
      * The employment case's normal sequence of events.

**Construction of Use case**

The behavior of the framework is graphically illustrated in use-case outlines. These graphs show how the framework is utilized at a high level, when seen through the perspective of an untouchable (actor). A utilization case graph can depict all or some of a framework's work instances.

A use-case diagram may include the following elements:

* + - * Actors.
      * Use cases.

**Relationships in use cases**

In use case diagrams, active relationships, also known as behavioral relationships, are shown. There are four main types of behavioral relationships.

1. **Communicates**

An actor is connected to a use case through a behavioral relationship. The purpose of the use case is to help the systems actor. It's important to document interactions between actors and use cases. An actor is connected to a use case with a line with no arrowheads.

1. **Includes**

The uses relationship is a situation in which multiple use cases share the same behavior. The other use cases include the common use case. A dotted arrow points to the common use case.

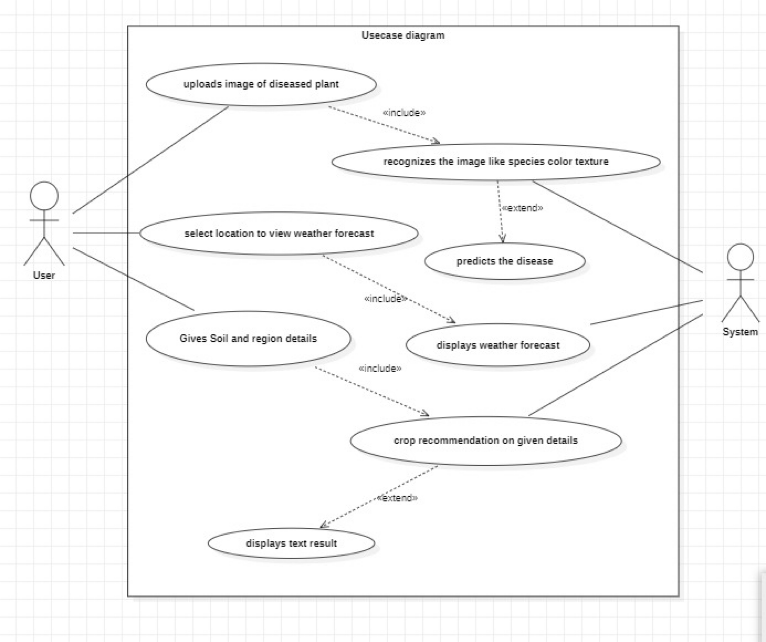
1. **Extends**

When one use case contains behavior that allows a new use case to handle a variant or exception, the extended connection describes it. The basic use case has exceptions. The basic and extended use case are connected by the arrow.

1. **Generalizes**

One thing is more prevalent than another according to the generalized relationship. There is a link between two use cases and two actors. The arrow points to something that is more general.

In this system Use Case diagram:



*Fig 4.1.1.2: UML use case diagram for the application*

**Actors:**

**User:** The person who interacts with the system, possibly a farmer or agricultural expert.

System: The automated system that processes the user's input and provides recommendations.

**Use Cases:**

**Uploads Image of Diseased Plant:**

* The user uploads an image of a plant that appears to be diseased.
* This use case is linked to the system’s ability to recognize the image.
* Recognizes the Image:
* The system processes the uploaded image and recognizes various aspects, such as species, color, and texture of the plant.

**Predicts the Disease:**

* Based on the recognition, the system predicts the possible disease affecting the plant.
* Select Location to View Weather Forecast:
* The user selects a location to view weather-related data, which may impact both disease progression and crop recommendation.

**Gives Soil and Region Details:**

The user provides soil and regional details. These are crucial inputs for the system to generate accurate crop recommendations.

**Displays Weather Forecast:**

The system displays the weather forecast for the selected location.

**Crop Recommendation on Given Details:**

Based on the soil, regional details, and weather forecast, the system provides crop recommendations.

**Displays Text Result:**

The system presents the result in text form to the user, showing disease predictions and crop recommendations.

### **SEQUENCE DIAGRAM**

A sequence diagram shows how a group of items interact. Business people and software engineers use these diagrams to understand requirements for a new system. The process is currently going on. Sequence diagrams are used for event scenarios. Businesses can use sequence diagrams as a reference. To show the diagram, make it.

* The specifics of a use case should be described.
* A model of the logic of a procedure is needed.
* In order to complete a process, objects and components need to interact with one another.
* Understand the specific functions of a current scenario. The following scenarios can be used to create a sequence diagram.

A usage scenario shows how technology might be used in the future. You have to think through every possible system usage situation.

**Method logic:**

A sequence diagram can be used to look at the logic of any function, method, or complex process just as it can be used to look at the rationale of a use case. A sequence diagram is a great way to map out service logic if you view a service to be high level.

**Vision sequence diagram:**

Vision sequence diagrams can be uploaded into the chart. Microsoft Vision allows you to import, but Lucid chart is an excellent alternative. The files are called vhd and vhdx.

**Object:**

There is a state, a lead and a personality in an object. For all intents and purposes, the structure and direction of objects are depicted in their fundamental class. There are objects in a diagram that represent a class. The object is an order case. That is not named.

**Message:**

An event is caused by an exchange of information between two articles. Information from the source point of control convergence to the objective point of control convergence is transmitted in a message.

**Link:**

An event is caused by an exchange of information between two articles. Information from the source point of control convergence to the objective point of control convergence is transmitted in a message.

**Lifeline:**

As time goes down, it reflects it. The events that happen to an object are depicted by a dashed vertical line. An actor symbol or a designated shape could be used as the starting point.

**Actor:**

There are entities shown that interact with the system.

**Message Symbols:**

**Synchronous Message:**

An arrowhead and solid line are used to symbolize this. When a sender has to wait for a response to a query before moving on, they use this symbol. The diagram should show both the call and the response.

**Asynchronous message:**

This is shown as a solid line with a lined arrowhead. A response is not required for asynchronous communications in order for the sender to continue. Only the call should be shown in the diagram.

**Delete message:**

A solid line with a solid arrowhead is followed by an X. This message has the result of destroying an object.

**Sequence diagram components:**

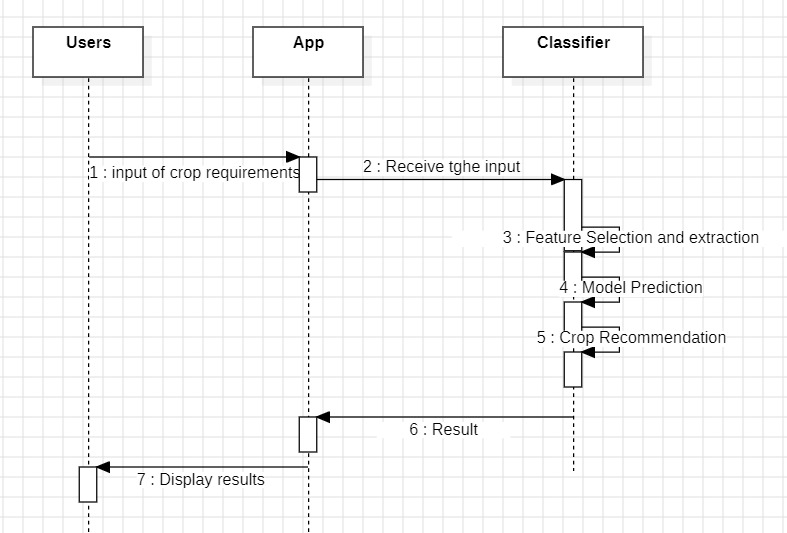
|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Symbol** |
| Object symbol | In UML, represents a class or object. The object symbol depicts an item's behavior within the framework of the system. It is inappropriate to list class attributes in this format. |  |
| The activation box | Depicts the amount of time required for an object to finish a task. The activation box gets longer the longer the task will take. |  |
| Actor symbol | Ddisplays entities that are external to the system or have interactions with it. |  |
| Lifeline symbol | Represents time passing by extending downward. The consecutive events that affect an object during the charting process are depicted by this vertical dashed line. Lifelines may start with an actor symbol or a designated rectangle form. |  |
| Alternative symbol | Symbolizes a decision between two or more message sequences (which are typically mutually exclusive). Use the marked rectangle shape with a dashed line within to represent options. |  |
| Message symbol | To send a message. |  |

*Table 4.1.2.1: Sequence Diagrams Symbols*



*Fig 4.1.2.1: Sequence Diagram for plant disease prediction*

In the plant disease prediction process, the user uploads an image of the plant through the app, which sends it to the classifier for analysis. The classifier identifies the plant, detects any disease symptoms, and classifies the specific disease. The result is then returned to the app, where it is displayed to the user as a disease diagnosis. This process efficiently links the user, app, and classifier to provide accurate disease detection.



*Fig 4.1.2.2: Sequence Diagram for crop recommendation*

In the crop recommendation process, the user inputs relevant data such as soil properties and region details into the app. The app forwards this data to the classifier, which extracts important features and uses a predictive model to determine the most suitable crops. The crop recommendations are then sent back to the app, which displays the results to the user, helping them make informed decisions.

### **ACTIVITY DIAGRAM**

A flowchart that shows the flow of information from one action to the next is called an activity diagram. The activity may be referred to as a system operation.

The control flow is led from one operation to the next. This flow in nature could be sequential, branched, or concurrent. Activity diagrams handle various types of flow control by utilising numerous sections, such as join, fork, and so forth.

The basic functions of the other four diagrams are also provided by activity diagrams. It captures the system's dynamic behaviour. Message flow from one item to the next is shown in the other four diagrams, but not in the activity diagram. An activity is a particular type of system operation. It doesn't demonstrate any communication between various activities.

**Notations**

**Initial point or start point**

Any activity diagram's beginning action state or starting point is represented by a small, filled circle and an arrow. Make sure the top left corner of the first column is where an activity diagram with swim lanes begins.

**Activity or Action state**

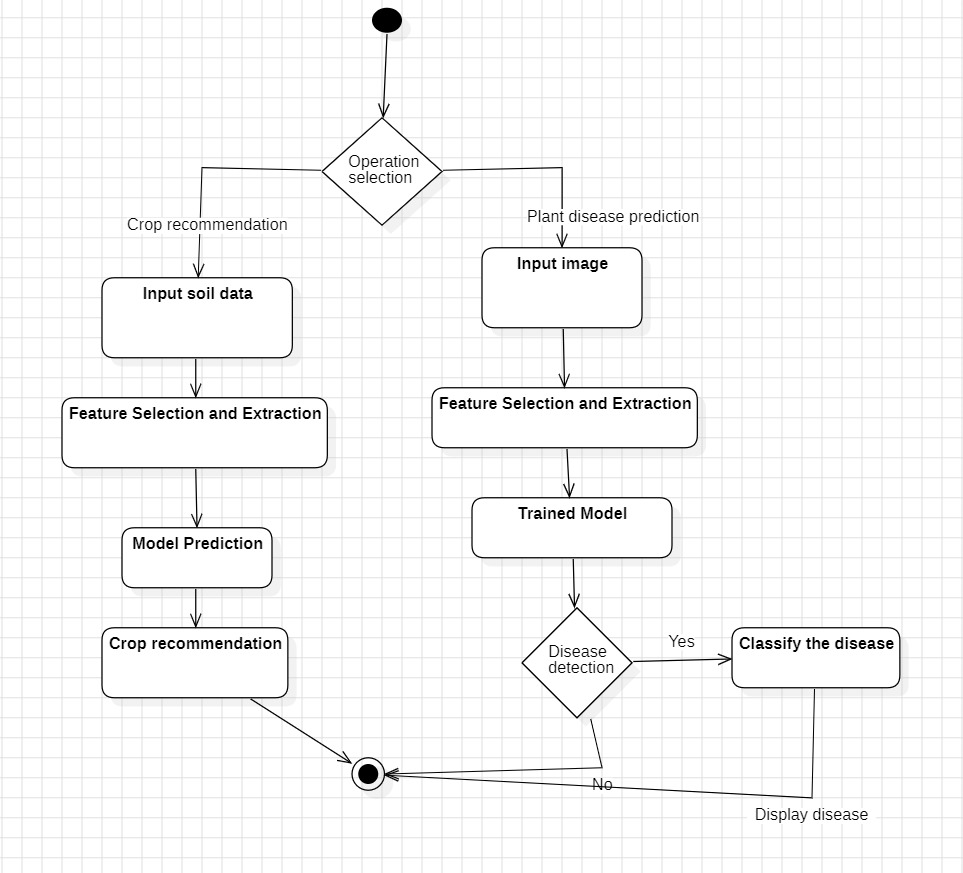
An object's non-interruptible action is represented by an action state. In Smart Draw, you may create an action state by drawing a rectangle with rounded corners.

**Action flow**

Action flows, sometimes referred to as edges and routes, show changes from one action state to another. To represent them, an arrowed line is frequently employed.

**Decisions and branching**

A diamond denotes a pick from several options. When one activity demands a choice before continuing, place a diamond between them. The outgoing alternates should be labelled with a condition or guard expression. Another way to name one of the pathways is "else."



*Fig 4.1.3.1: Activity Diagram*

In the crop recommendation process, the user inputs relevant soil data, and key features are selected and extracted. A predictive model processes these features to recommend a suitable crop, which is then displayed to the user. In the plant disease prediction process, the user provides an image of the plant, and important features are extracted for analysis. A pre-trained model analyzes the features to detect any disease. If a disease is detected, it is classified, and the result is displayed to the user; otherwise, no further action is needed.

### **CLASS DIAGRAM**

A class diagram is another name for a static diagram. It shows the application's static view. A class diagram can be used to create executable code for a software programmer as well as to visualize, describe, and document various components of a system.

A class diagram describes the characteristics, actions, and limitations of a class.

One that has been forced upon it Because they are the only UML diagrams that can be directly transferred to object-oriented languages, class diagrams are frequently employed in the modelling of object-oriented systems. An assortment of classes, interfaces, affiliations, collaborations, and constraints are shown in a class diagram. An alternative term for it is a structural diagram.

The following is a summary of the class diagram's purpose

* The static view of an application is analyzed and designed.
* Describe the responsibilities of a system.
* Item and delivery diagrams are built on this base.
* Both forward and reverse engineering are used.

When A static diagram is also known as a class diagram. It displays the static view of the application. A class diagram can be used to visualize, describe, and document different system components as well as to produce executable code for a software programmer.

A class diagram explains the traits, behaviours, and restrictions of a class.

One to which it's been compelled Class diagrams are often used in the modelling of object-oriented systems since they are the only UML diagrams that can be directly translated to object-oriented languages. A class diagram displays a variety of classes, interfaces, affiliations, collaborations, and limitations. It can also be referred to as a structural diagram.

**Aggregation or a-part-of relationship:**

It refers to situations in which a classification is made up of various class segments. A classification made up of different classes does not function as a whole. It's extremely difficult to keep going. This relationship's fundamental characteristics are transitivity and hostility to evenness.

The answers to the following questions indicate the distinction between a part and a whole relationship:

* Does the part class have a place in the problem area?
* Is the part class subject to the framework's responsibilities?
* Is the part class capable of catching a serious single worth? (If not, simply incorporate it as a class trait.)
* Does this provide valuable deliberation for dealing with the issue?

**Assembly:**

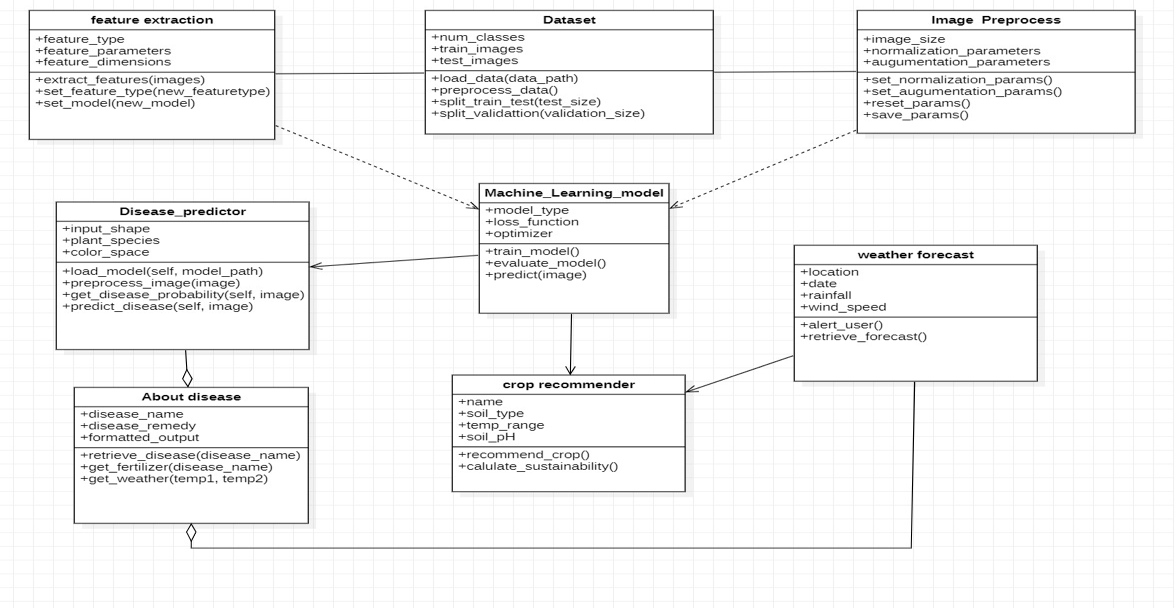
It is assembled from its parts, and there is a physical assembly-part circumstance.

**Container:**

The physical entire envelope, on the other hand, is not made up of physical parts.

**Member of the collection:**

The theoretical entire is made up of physical or applied parts. Empty precious stones speak to the container and collection, but strong jewels speak to the creation.



*Fig 4.1.4.1: Class Diagram*

This class diagram represents a system for crop recommendation and plant disease prediction using machine learning. It consists of classes like Feature Extraction, which handles feature selection from data, and Dataset, which manages data loading and splitting for training, testing, and validation. The Image Preprocess class deals with image normalization and augmentation for better model accuracy. The core Machine Learning Model class trains, evaluates, and predicts outcomes, feeding into specialized components like the Disease Predictor for detecting plant diseases and the Crop Recommender, which suggests crops based on input parameters. Additionally, the About Disease class provides detailed information about detected diseases, linking predictions to practical advice such as fertilizer recommendations.

# **PROPOSED SYSTEM**

* 1. **METHODOLOGY**
* Implement a data-driven crop recommendation system that utilizes advanced analytics and machine learning algorithms to analyze soil data, climate data, historical performance, and crop rotation principles.
* Develop an image processing system that can automatically detect the disease of the crop (if any) and also recommend the fertilizers necessary.
* Utilize the previous market price data to analyze historical market data, and provide the trends in the data through graphs. So that they can analyze and make decisions from it.
* Incorporate advanced meteorological models and satellite data to provide accurate short-term and long-term weather forecasts tailored to specific geographic regions.
* Offer training programs and educational resources to help farmers understand how to effectively use the proposed system and interpret the results.
  + 1. **OVERALL FRAMEWORK**

This framework combines front-end development, API integration, and machine learning. The front end collects input data, the API fetches the data, and the machine learning models process the data to make predictions. This framework is designed to provide users with a user-friendly interface for inputting information and receiving accurate disease predictions based on machine learning algorithms.

**Steps Included:**

1. Training the datasets.

2. Fetching the input data from the users.

3. Predicting the diagnosis.

**Train the Dataset:**

- Select an appropriate and diverse dataset for each disease.

- Extract the required features from the dataset for diagnosis purposes.

- Train each dataset using several machine learning models.

- Identify the model giving the best accuracy and use it for disease prediction.

The proposed framework begins with data preprocessing, which includes handling null values, removing duplicates, and applying label encoding. Different classifiers are applied to the processed data, resulting in a prediction system.

**Concepts Used:**

**TensorFlow:** An open-source machine learning library known for its versatility and scalability, ideal for developing and deploying machine learning models.

**Pandas:** A powerful Python library for data manipulation and analysis, providing high-performance data structures and tools.

**NumPy:** A Python library used for numerical computing, supporting large, multi-dimensional arrays and matrices.

**Keras:** A deep learning library that provides a high-level interface for building and training neural networks.

**Long Short-Term Memory (LSTM):** A recurrent neural network architecture designed to process sequential data and capture long-term dependencies.

**Data Preprocessing:**

Data preprocessing is a crucial step that involves cleaning, transforming, and integrating data to prepare it for analysis.

**Fetching the Input Data and Processing:**

* Load the dataset.
* Encode the target variable.
* Remove unnecessary columns.
* Split the dataset into training and validation sets.

**5.1.2 MODULES**

**Data Loading and Cleaning Module:**

* Loads the dataset, checks for missing values, and cleans the data.
* Outputs a clean dataset for analysis.

**Exploratory Data Analysis (EDA) Module:**

* Performs EDA to understand relationships between variables and the target variable.
* Generates visualizations to highlight these relationships and insights.

**Feature Selection and Engineering Module:**

* Selects the most relevant features for the prediction model.
* Performs feature engineering if necessary.

**Model Training and Evaluation Module:**

* Trains different machine learning models on the dataset.
* Evaluates their performance and selects the best model.

**Model Deployment Module:**

* Deploys the selected machine learning model to a production environment for making predictions on new data.

**5.1.3 FUNCTIONALITIES**

**Functionality of Data Collection and Cleaning:**

* Loads the dataset.
* Encodes the target variable.
* Removes unnecessary columns.
* Splits the dataset into training and validation sets.

**Functionality of Exploratory Data Analysis (EDA):**

* Performs EDA to identify patterns and correlations.
* Generates visualizations to help understand the relationships between different variables.

**Functionality of Model Architecture:**

* Constructs a sequential model using a deep learning library.
* Adds various layers including Conv1D, MaxPooling1D, and LSTM to capture temporal dependencies.
* Adds Dense layers with different units and an output layer based on the number of classes.

**Functionality of Model Training:**

* Compiles the model using an optimizer and loss function.
* Trains the model, monitoring progress and evaluating performance.

**Functionality of Model Evaluation and Prediction:**

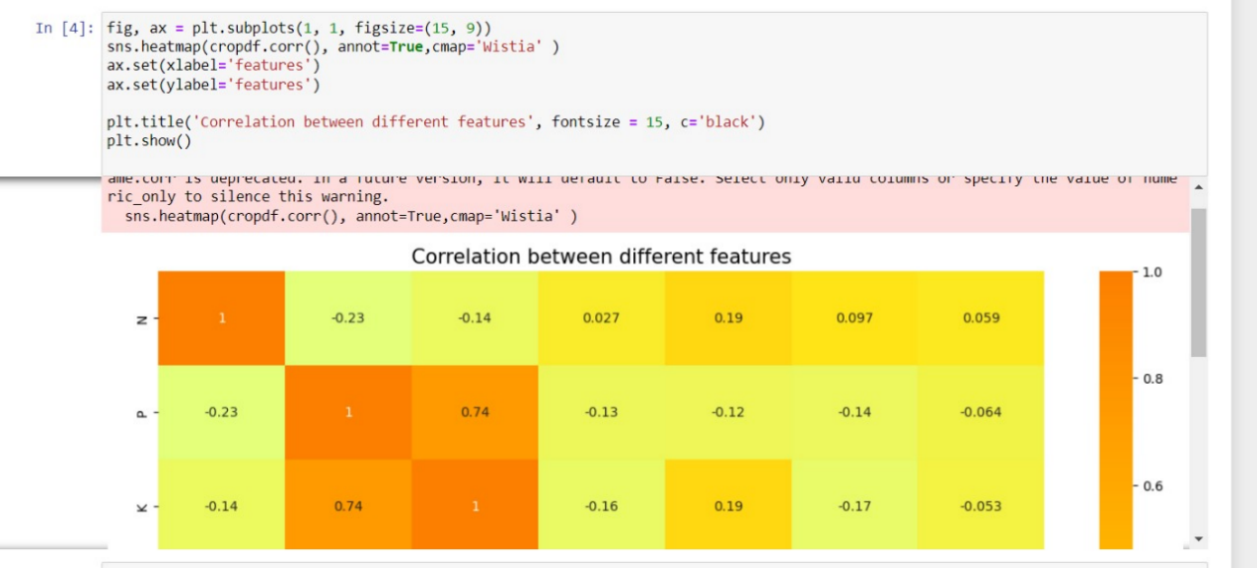
* Generates predictions on the validation set.
* Calculates misclassifications and evaluates the model's performance.

**5.2 ADVANTAGES OF PROPOSED SYSTEM**

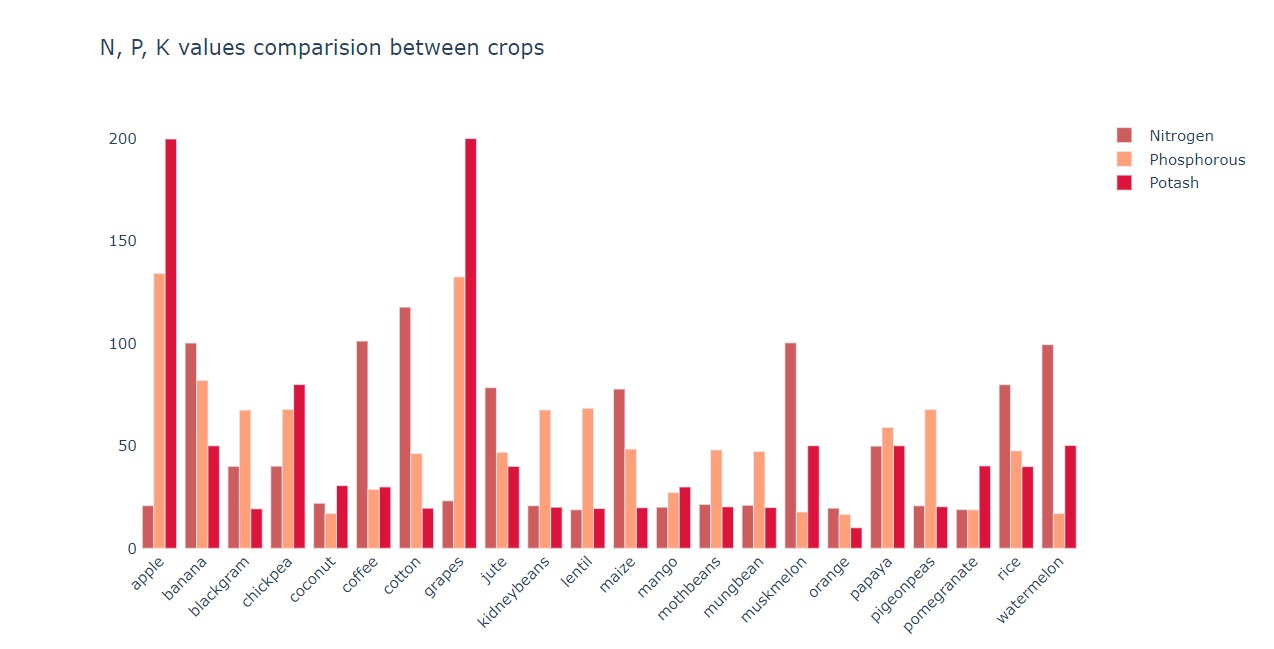
* Data-Driven Insights: The app provides farmers with real-time data and actionable insights, enabling informed decision-making, which is crucial in adapting to changing agricultural conditions.
* Adaptability to Climate Change: With its weather forecasting capabilities, the app can help farmers adapt to unpredictable weather patterns caused by climate change, ensuring better preparedness**.**
* User-Friendly Interface: Designed with ease of use in mind, the app can be accessible to farmers with varying levels of technical expertise, ensuring widespread adoption.
* Scalability: The app can be scaled to different types of crops and farming systems, making it a versatile tool for diverse agricultural environments.

# **6. CODING AND IMPLEMENTATION**

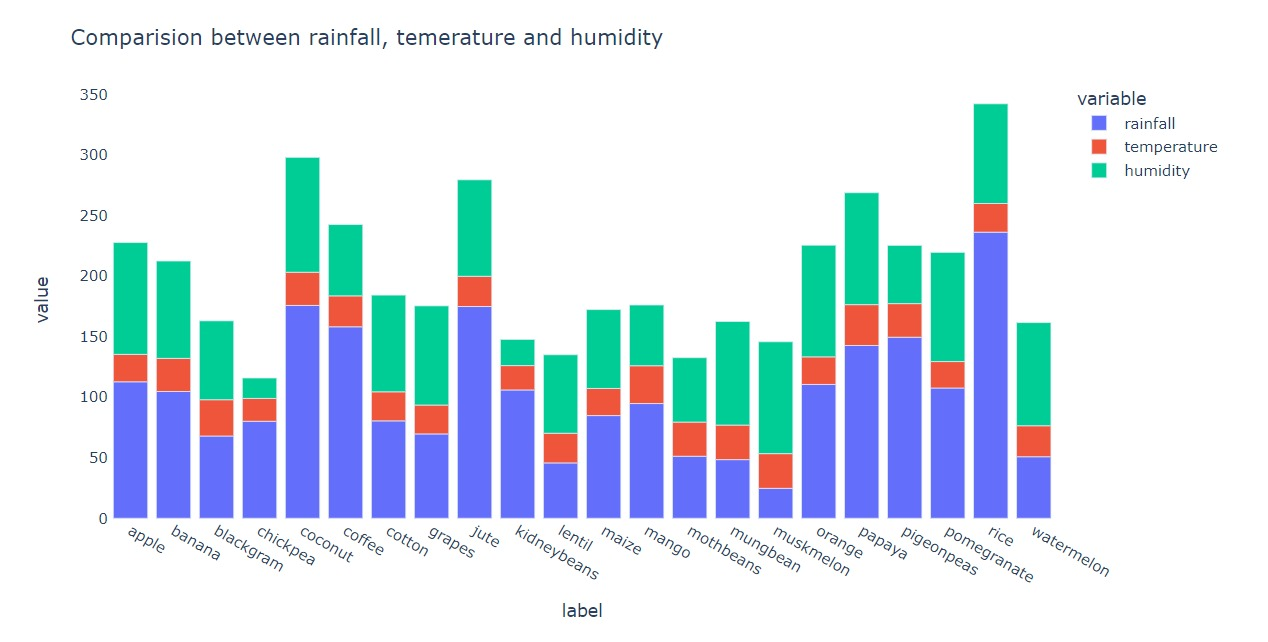
**Correlation between different features in crop dataset**

****

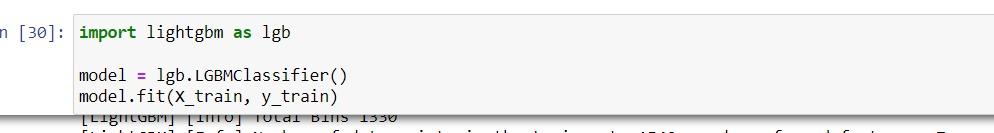
**N, P, K values comparisons between crops**

****

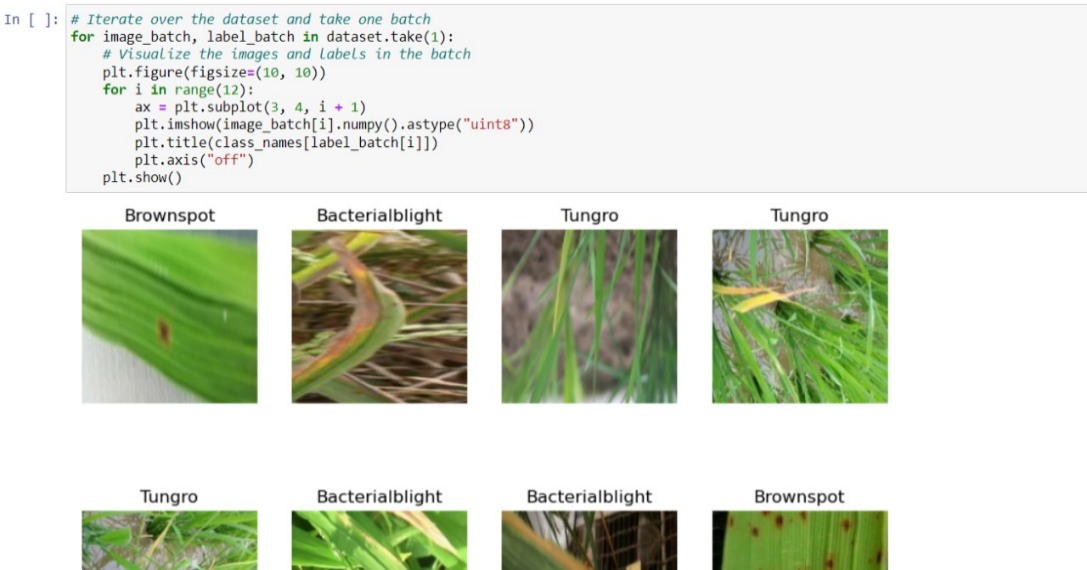
**Comparisons between rainfall, temperature and humidity**

****

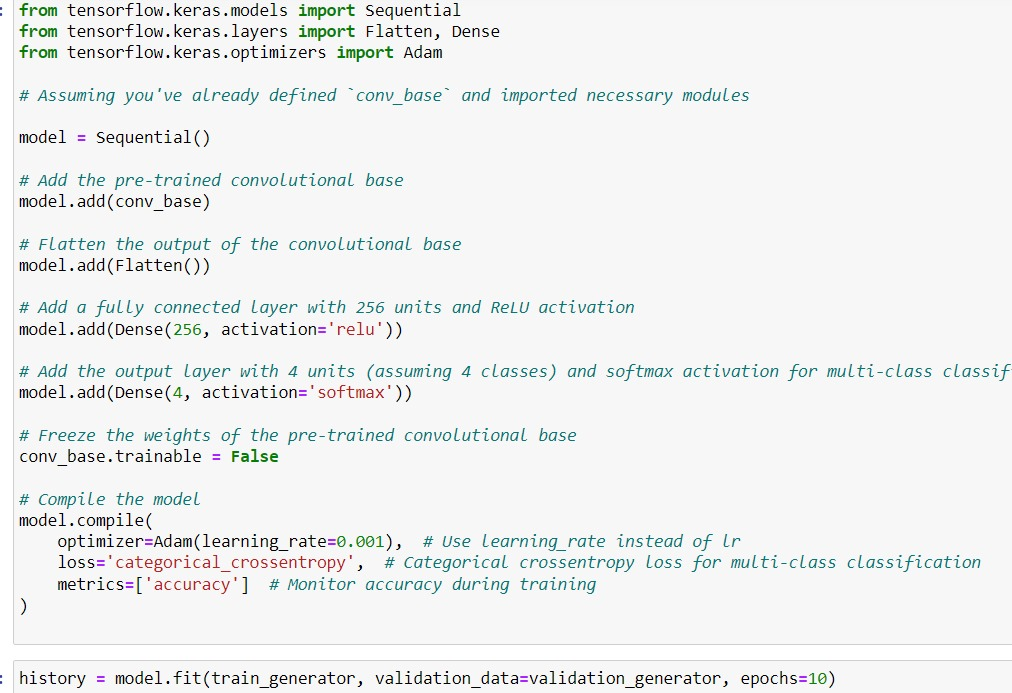
**Model fitting**

****

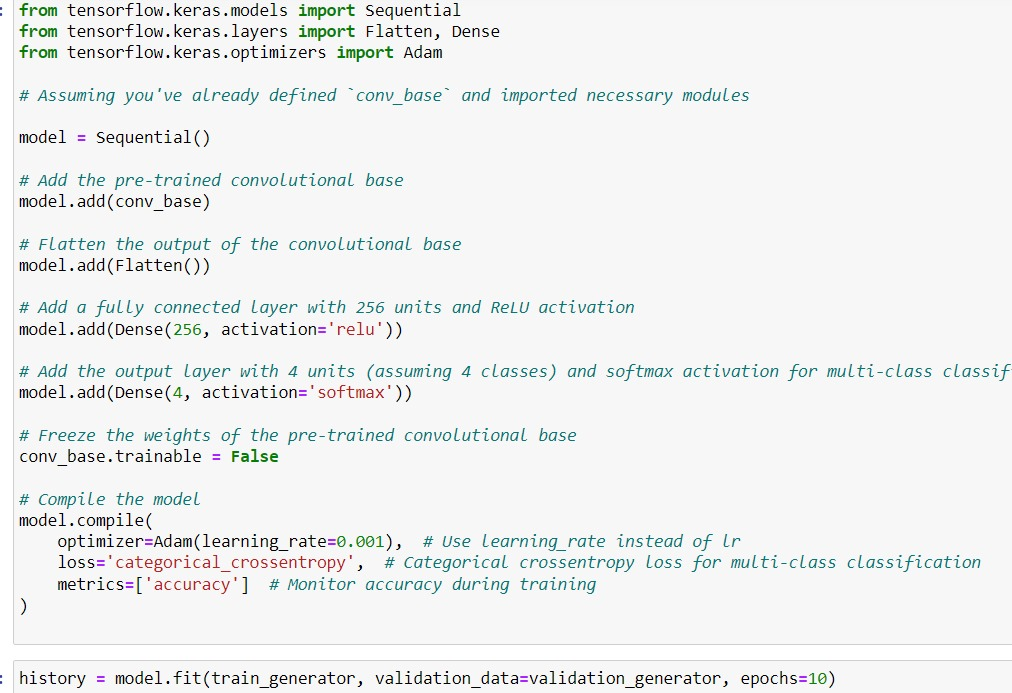
**For rice disease prediction**

****

**Model fitting**



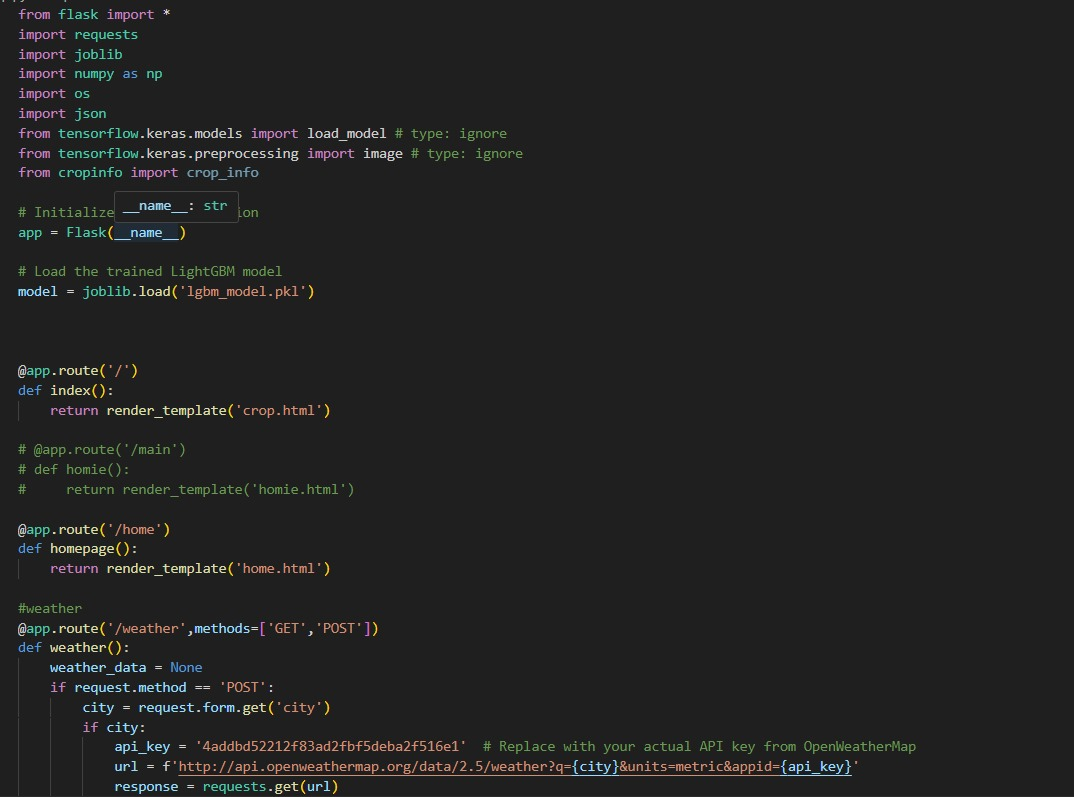
**Training model for cotton dataset**

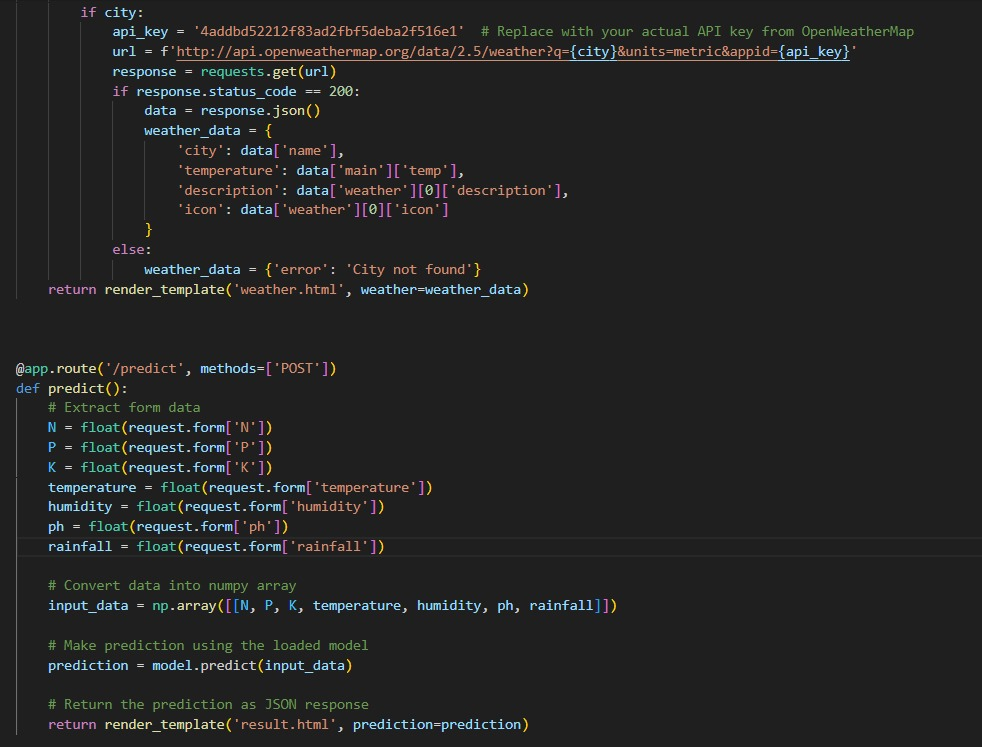


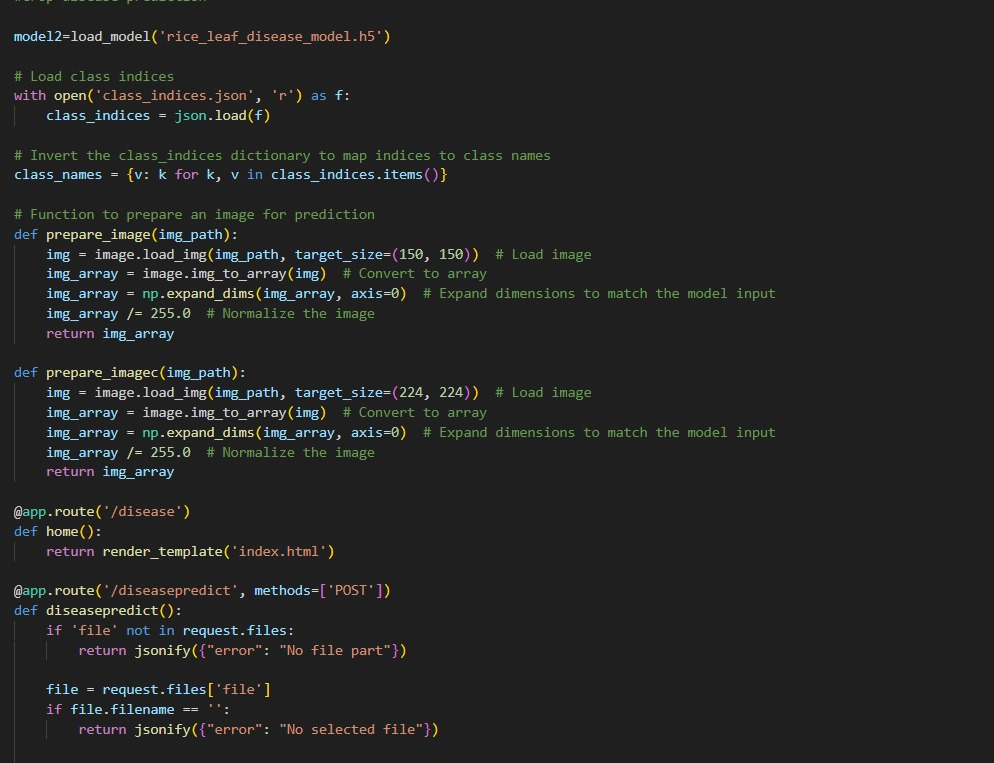
**Model fitting**

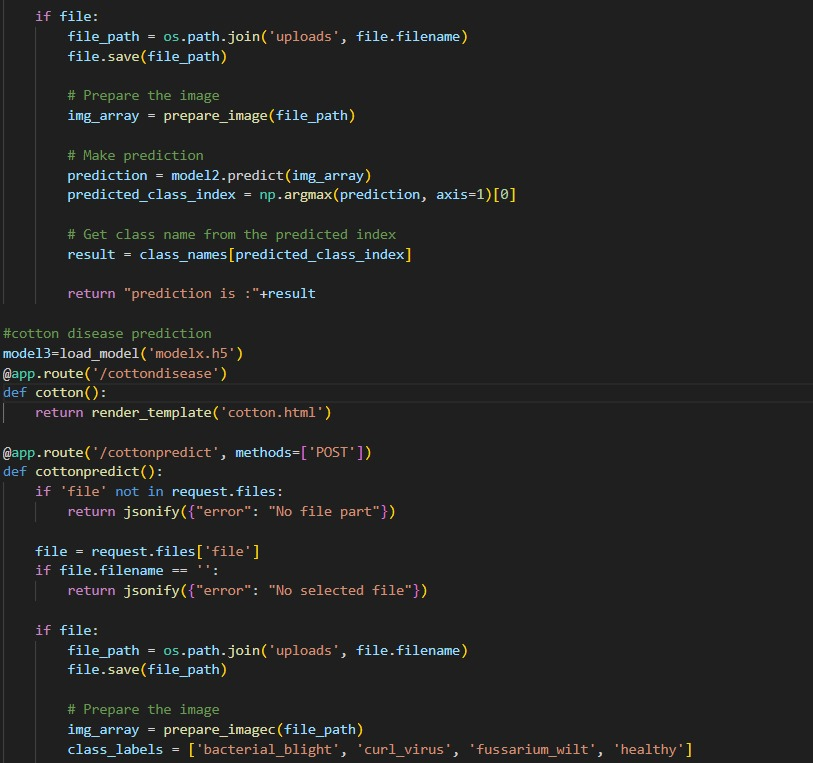


**Flask application**











**7.TESTING**

Testing in machine learning validates raw data and checks the ML model's performance. The primary goals of testing machine learning models are:

* Quality Assurance
* Detect bugs and flaws

An unseen dataset is used to test your model after the machine learning model is built. Testing data is used to assess the performance and progress of your training and to adjust it for better results.

The main criteria for testing data is that it should:

* Represent the original dataset.
* Be large enough to generate accurate results.
  1. **TYPES OF TESTING**
     1. **MANUAL TESTING**

The use of automated tools is not involved in manual testing. There are issues, bugs, and defects in the software application that can be detected with manual testing. Manual software testing provides critical help in finding bugs in the software application, and it is the most basic technique of all testing types.

It is necessary to manually test any new application. Although Manual Software Testing is a time-consuming effort, it is necessary to determine the feasibility of automation. Skills in any testing tool are not required for Manual Testing concepts. One of the fundamentals of Software Testing is that “100% Automation is not possible”. Manual testing is essential.

* + 1. **AUTOMATED TESTING**

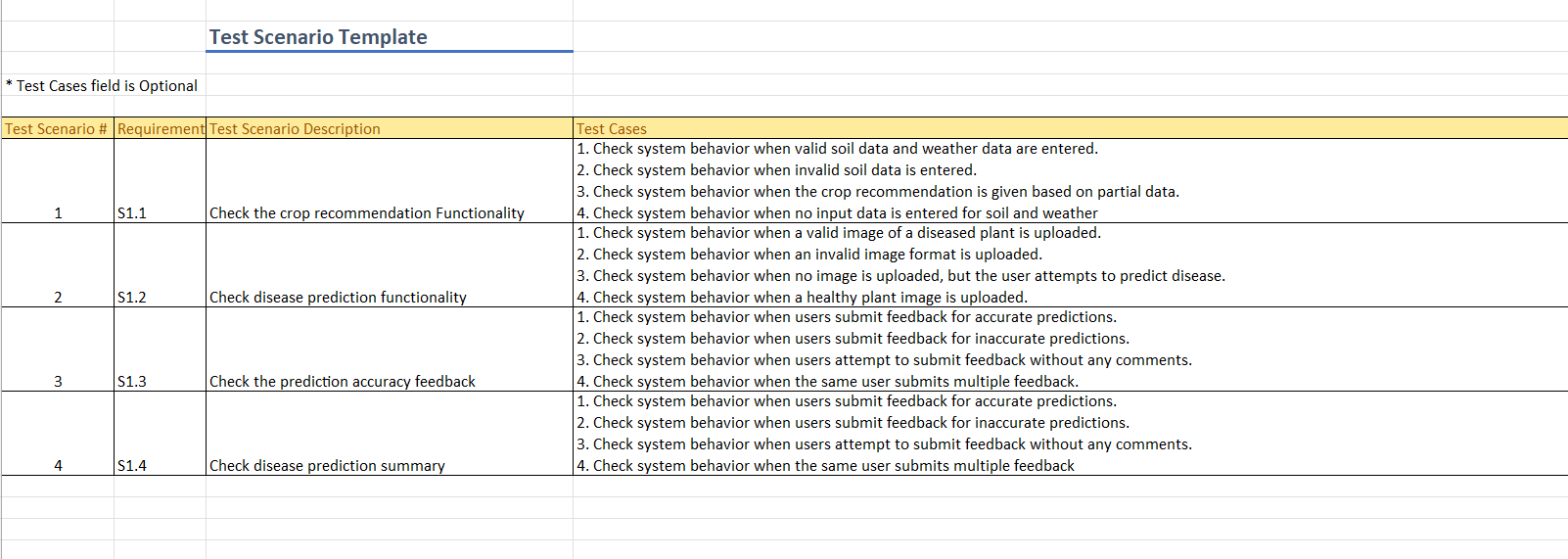
The technique of automation testing is used to execute a test case suite. Manual testing involves a person on a computer.

In addition to performing automated tests, the testing software feeds test data into the System Under Test, compares anticipated outcomes with the actual results and creates comprehensive test reports. Considerable investments of money and resources are essential for Software Test Automation.

This is a type of software test that checks whether the application functions as specified in the requirement specification. Each function shall perform functional tests by providing a value, determining output and verifying actual outputs in accordance with the expected values. In order to confirm that the functionality of an application or system behaves as we expect, functional tests performed as Blackbox tests are presented. This is to check the application's functionality. The concept of Blackbox testing was also used to describe functional tests

* + 1. **FUNCTIONAL TESTING**
       1. **INTEGRATION TESTING**

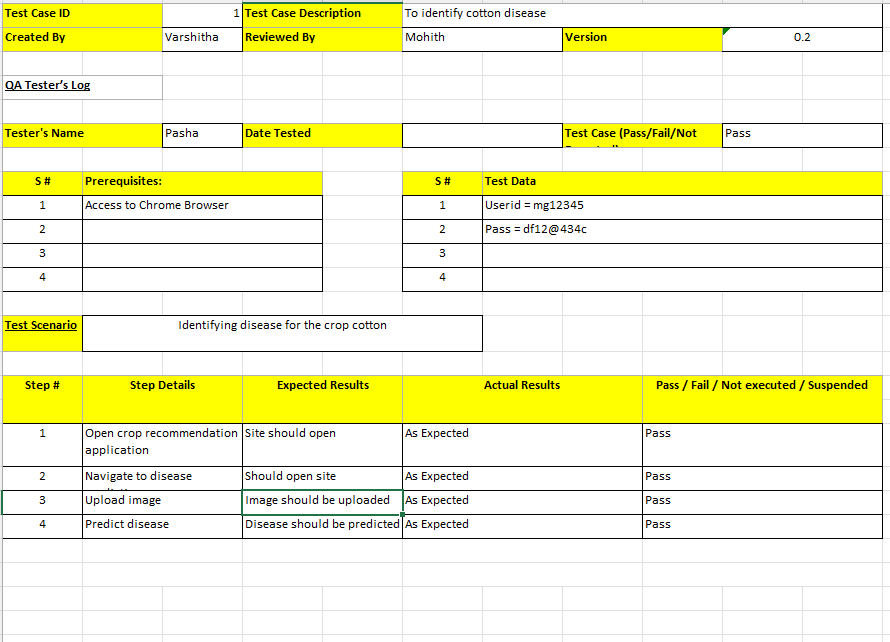
Integration tests shall be performed to test modules and components when attached, with a view to verifying that they are working as expected on an individual basis and do not have problems during the integration. Testing interfaces among units or modules is a primary function or goal of this test. Individual modules shall be subjected to isolation testing first. Once the modules have been checked and configured, they will be integrated one at a time until each module is built up to ensure combinational behavior and verify that the requirements are met correctly.



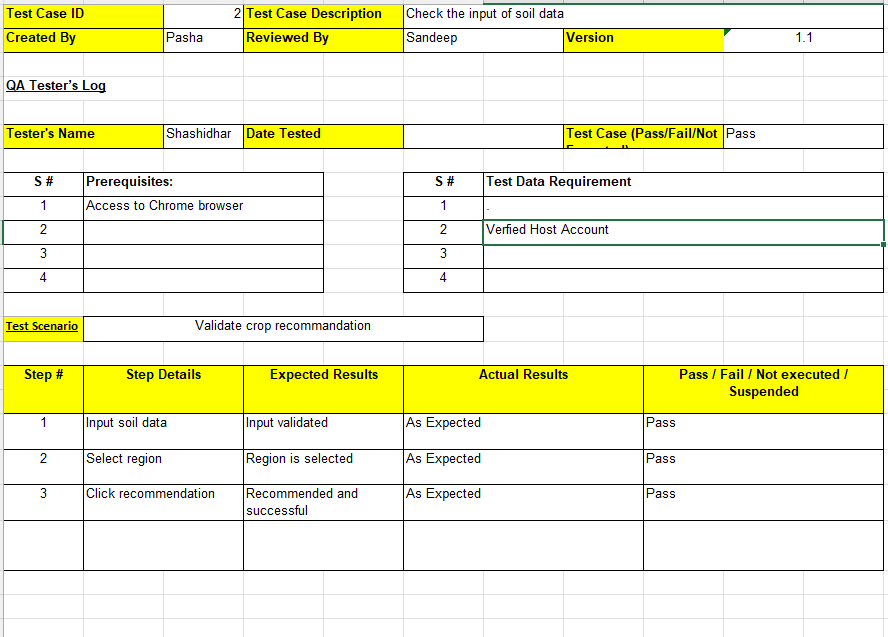
*Figure 7.1: Integration Testing*

**7.1.3.2 UNIT Testing**

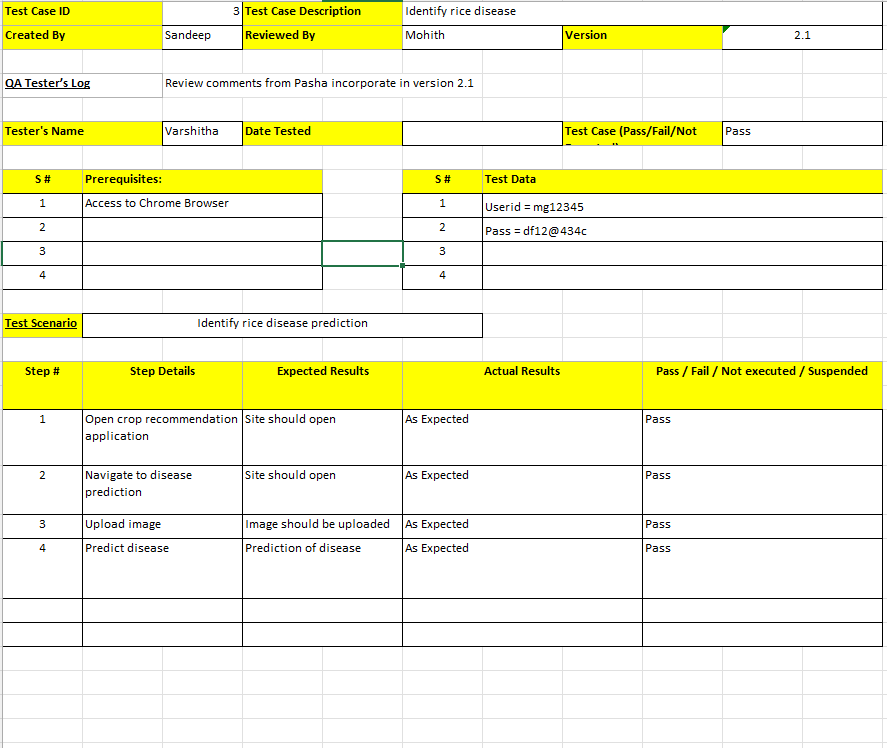
**Unit testing** is software testing technique, through which each part of the piece of software can be tested to ascertain if it's appropriate for use by means of a set of user tasks and operating procedures or not. A software testing type where each part of the software is tested shall be classified as a unit test. During the application development, a unit test is carried out on the software product.



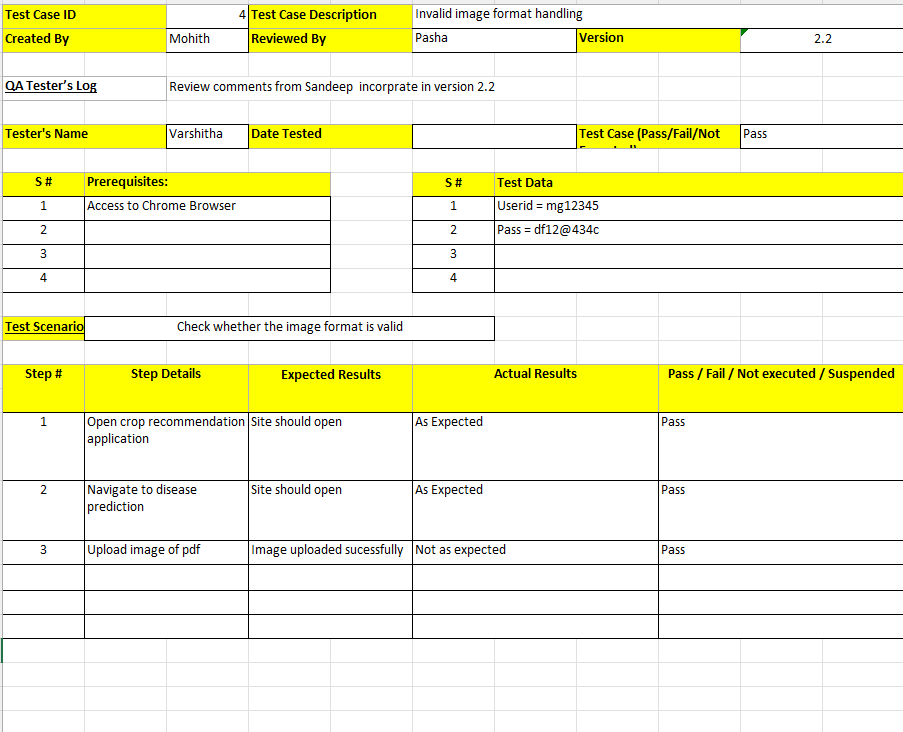
*Figure 7.2: Cotton Crop Disease Identification Testing*



*Figure 7.3: Crop Recommendation Validation Testing*



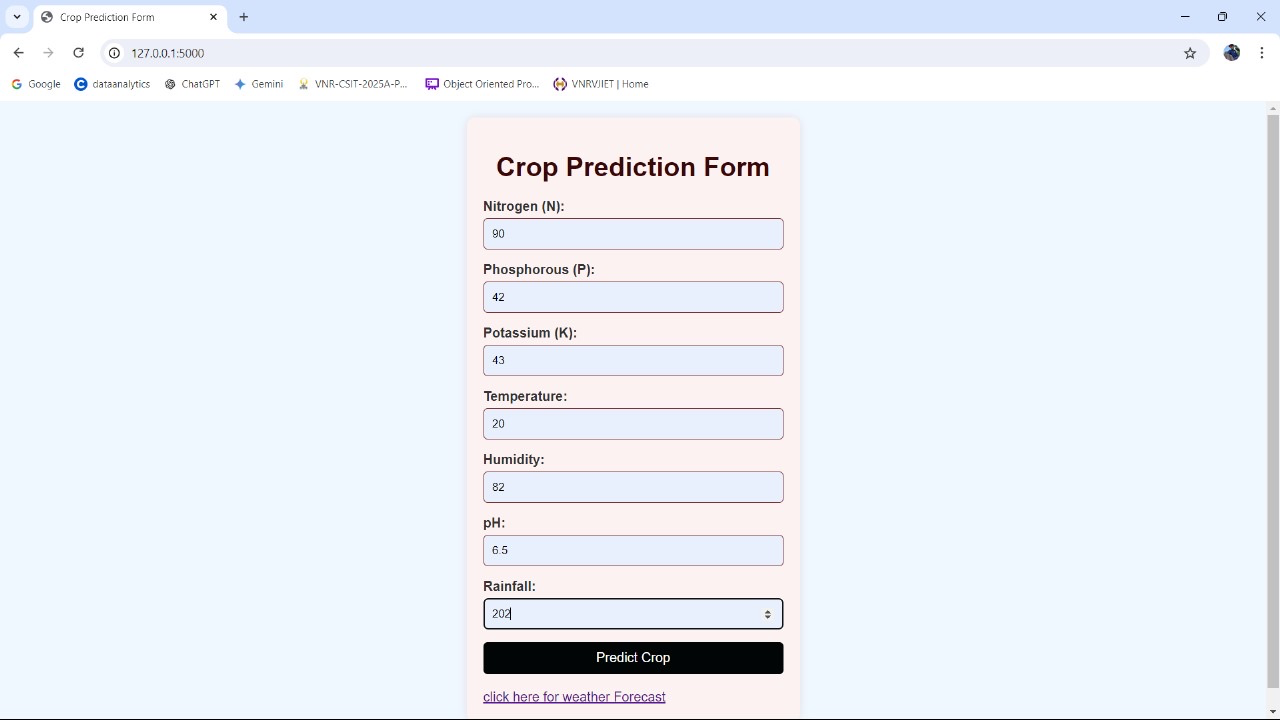
*Figure 7.4: Rice Disease Prediction Testing*



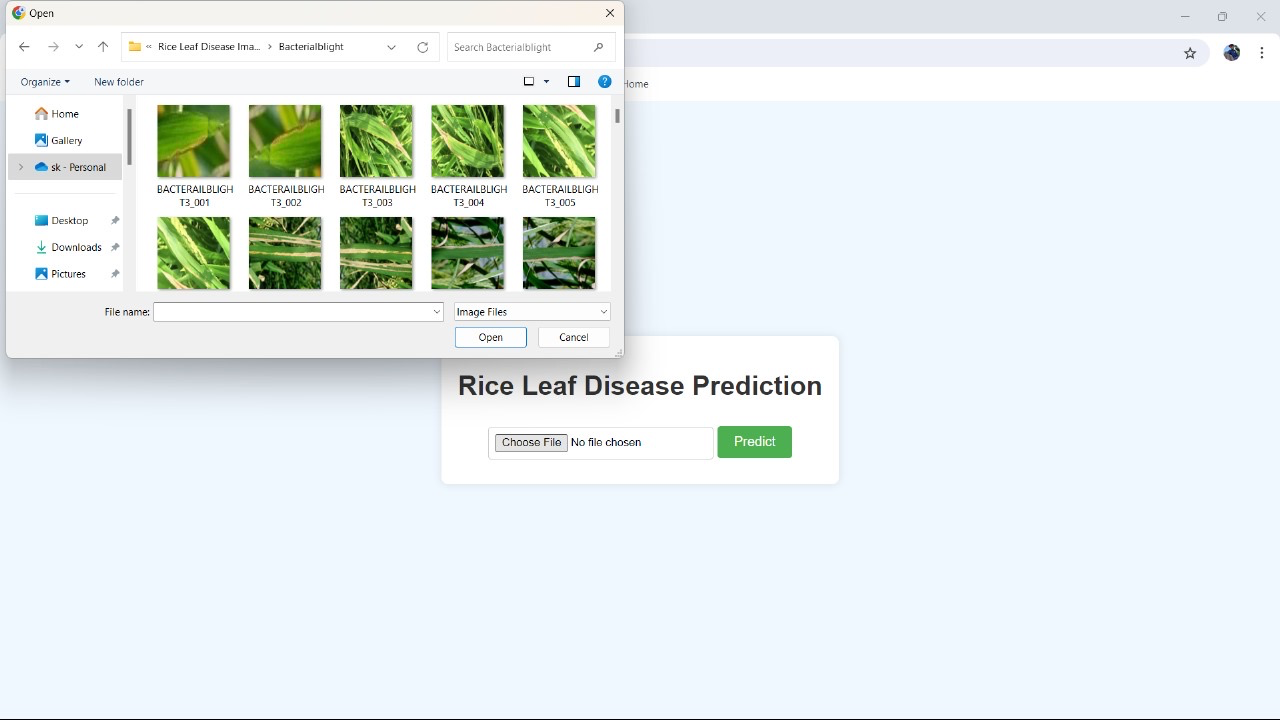
*Figure 7.1: Image Format Validation Testing*

# **8. RESULTS**

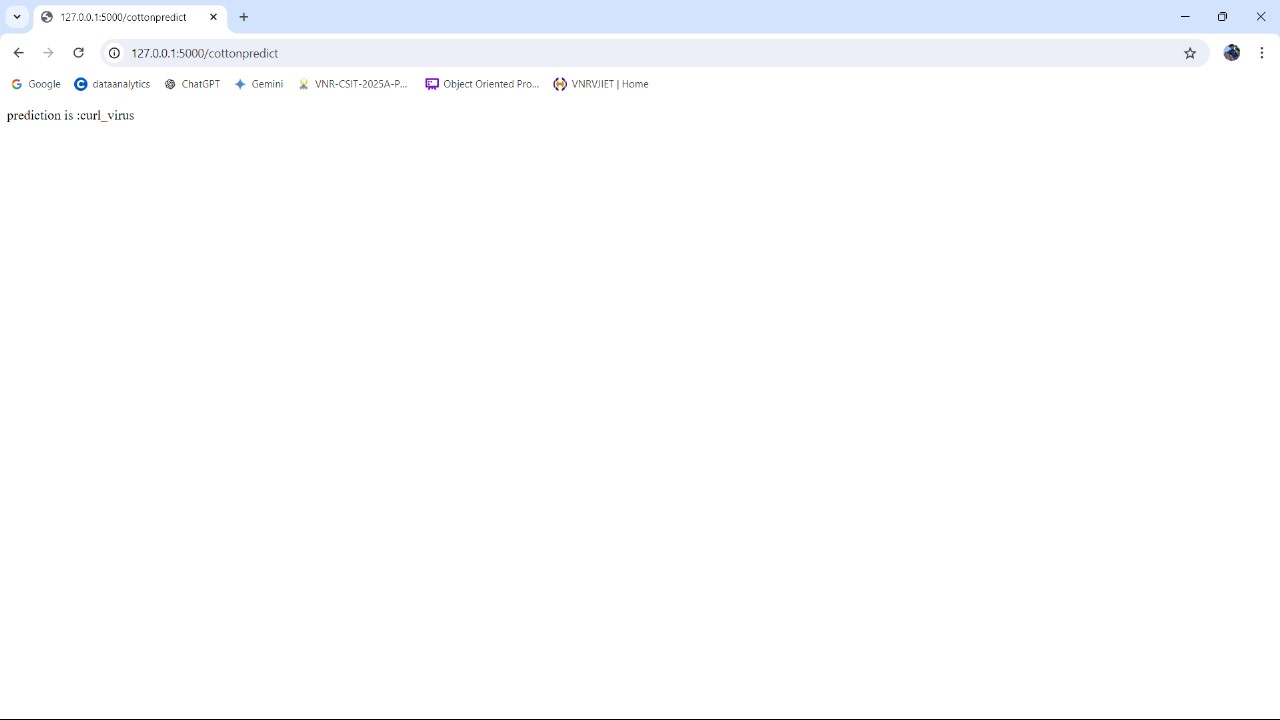
**8.1 Interface of crop prediction form**



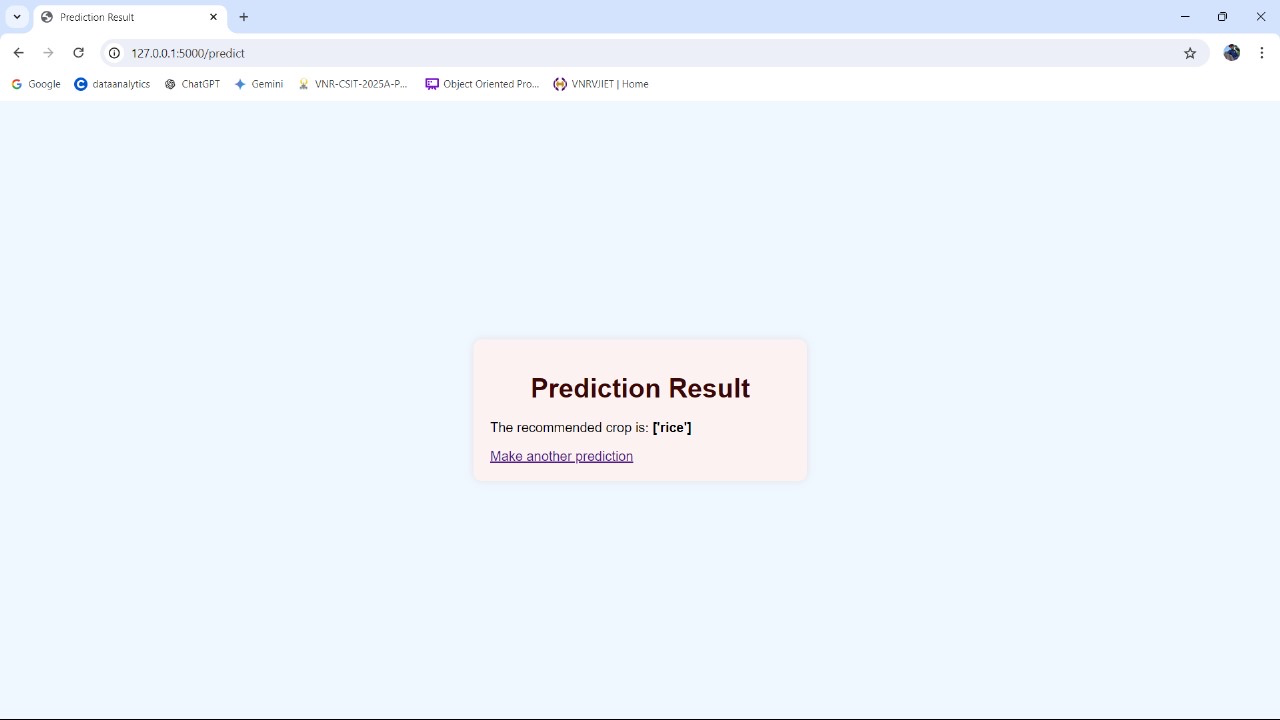
**8.2 Rice leaf disease predictor**



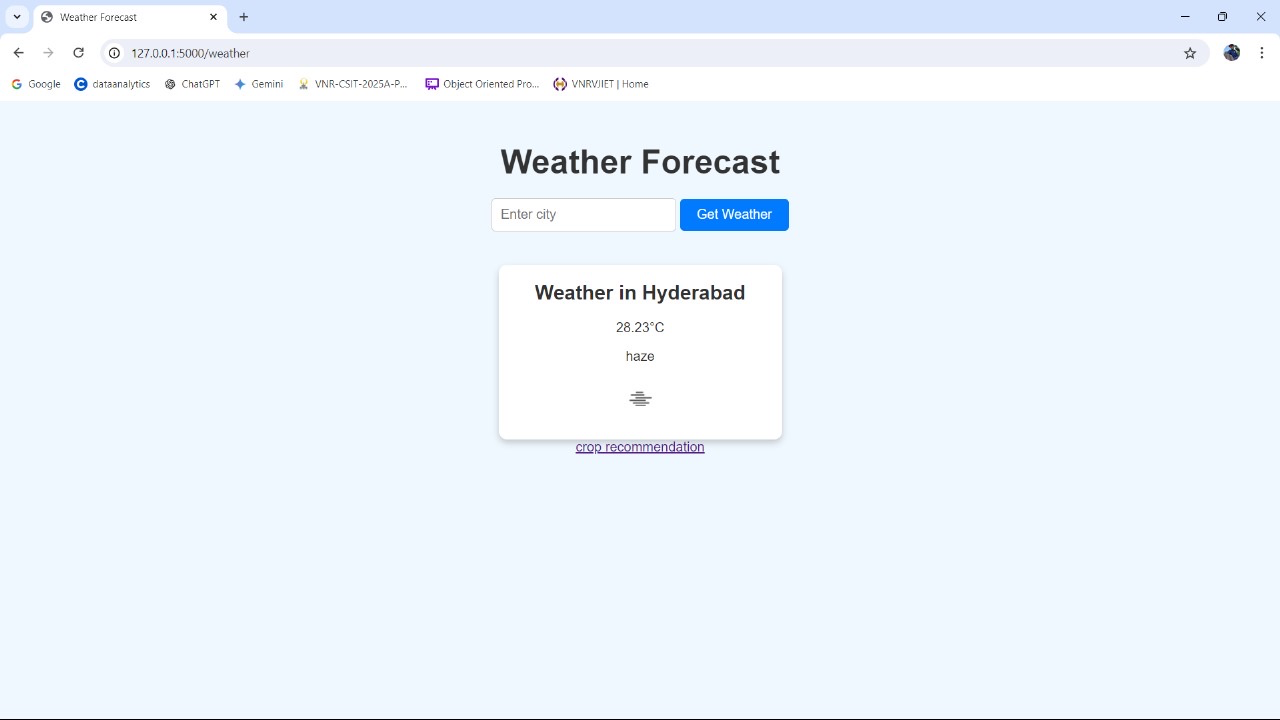
# **8.3 Predicted disease of rice leaf**



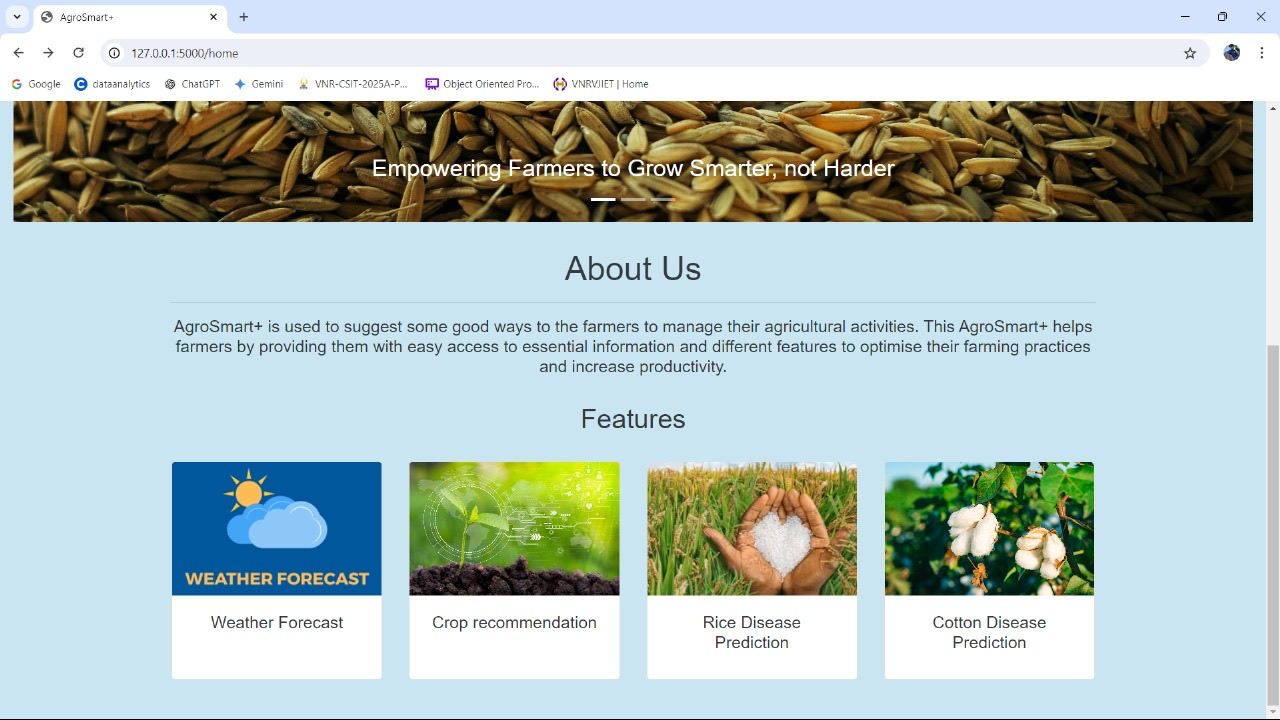
# **8.4 Recommended crop for better yield**



# **8.5 Weather forecast**



# **8.6 Features and about us**



# **9. CONCLUSION AND FUTURE WORK**

In The Smart Field Project represents a significant advancement in the application of technology to modern agriculture, offering a comprehensive solution to some of the most pressing challenges faced by farmers today. By integrating machine learning algorithms with precision agriculture, weather forecasting and relevant crop recommendation systems, the project provides a powerful tool for enhancing crop management, reducing losses, and promoting sustainable farming practices. The ability to tailor disease prediction and crop recommendations to specific conditions not only increases productivity but also contributes to environmental stewardship by optimizing resource use

The disease prediction and treatment feature stands-out as a critical component, utilizing machine learning algorithms to monitor crops in real-time. By analyzing indicators such as leaf patterns, growth anomalies, and environmental conditions, the system can detect potential diseases at an early stage. This early detection enables timely and targeted interventions, reducing crop losses and minimizing the need for broad-spectrum pesticide applications. This precision in disease management not only improves crop health but also contributes to more sustainable farming by reducing chemical usage.

The crop recommendation system is another key feature of the Smart Field Project. By incorporating data from various sources, including soil health, historical crop performance, and localized weather patterns, the system provides farmers with personalized recommendations on the most suitable crops to plant. This tailored approach maximizes yield potential and ensures that crops are well-suited to the specific conditions of each field, thereby enhancing overall farm productivity and profitability.

Weather forecasting is a vital feature that further enhances the decision-making capabilities of farmers. By integrating sophisticated weather models, the system provides accurate and localized forecasts that inform critical farming activities such as planting, irrigation, and harvesting. The ability to anticipate weather changes allows farmers to optimize their operations, avoid adverse conditions, and reduce the risks associated with unpredictable weather patterns.

In conclusion, the Smart Field Project is a significant step forward in the quest for sustainable and efficient agricultural practices. By leveraging advanced technologies in disease management, crop selection, and weather forecasting, the project empowers farmers to make informed decisions that enhance productivity while promoting environmental stewardship. As the project continues to evolve, it holds the promise of reshaping the future of agriculture, making it more resilient, data-driven, and sustainable for generations to come.

# **10. REFERENCES**

Here are all 15 references combined:

[1] P. Sharma, A. Singh, et al. "Crop Recommendation System Using Machine Learning Algorithms and Soil Parameters." International Journal of Engineering and Technology (IJET). [Link](https://www.ijert.org/crop-recommendation-system-using-machine-learning-algorithms-and-soil-parameters)

[2] V. M. Mane, S. A. Hake. "A Data-Driven Approach for Crop Selection Using Data Mining Techniques." International Journal of Advanced Research in Computer Science and Software Engineering. [Link](https://www.ijarcsse.com/data-driven-approach-crop-selection)

[3] D. Patel, P. Patel, et al. "Cotton Disease Detection Using Machine Learning Techniques." International Journal of Innovative Research in Computer Science & Technology (IJIRCST). [Link](https://www.ijircst.org/cotton-disease-detection-using-machine-learning-techniques)

[4] A. S. Raut, S. R. Patel. "Predicting Cotton Leaf Diseases Using Convolutional Neural Networks." IEEE Xplore. [Link](https://ieeexplore.ieee.org/predicting-cotton-leaf-diseases-using-convolutional-neural-networks)

[5] X. Wang, Y. Zhang, et al. "Rice Leaf Disease Identification Using Deep Convolutional Neural Networks." Computers and Electronics in Agriculture. [Link](https://www.sciencedirect.com/rice-leaf-disease-identification-using-deep-convolutional-neural-networks)

[6] J. Wang, L. Gao, et al. "A Comparative Study of Machine Learning Algorithms for Rice Leaf Disease Detection." Agriculture Journal. [Link](https://www.agriculturejournal.org/comparative-study-machine-learning-algorithms-rice-leaf-disease-detection)

[7] S. N. Patil, P. M. Waghmare. "Crop Yield Prediction Based on Climate Parameters Using Machine Learning." International Journal of Computer Applications. [Link](https://www.ijcaonline.org/crop-yield-prediction-climate-parameters-using-machine-learning)

[8] R. K. Bhoi, P. R. Jena. "A Machine Learning Approach to Enhance Crop Selection for Sustainable Agriculture." International Journal of Advanced Computer Science and Applications (IJACSA). [Link](https://www.ijacsa.thesai.org/enhancing-crop-selection-ml-approach)

[9] A. Kumar, K. P. Singh. "Crop Recommendation System Based on Soil Analysis Using Machine Learning Techniques." Journal of Emerging Technologies and Innovative Research (JETIR). [Link](https://www.jetir.org/crop-recommendation-soil-analysis-machine-learning)

[10] M. V. Pawar, P. H. Patil. "Precision Agriculture Using Machine Learning for Crop Disease Prediction." International Journal of Computer Science and Information Security (IJCSIS). [Link](https://www.ijcsis.org/precision-agriculture-ml-crop-disease-prediction)

[11] N. G. Patel, V. S. Shah. "Development of Smart Farming System Using IoT and Machine Learning." Journal of Computer Science Applications (JCSA). [Link](https://www.jcsa.org/smart-farming-iot-ml)

[12] H. Singh, R. K. Verma. "Prediction of Crop Yield Using Artificial Neural Networks and Soil Data." International Journal of Agricultural Science Research (IJASR). [Link](https://www.ijasr.org/prediction-crop-yield-using-ann-soil-data)

[13] P. K. Gupta, S. Tiwari. "Smart Agriculture: Leveraging Machine Learning for Crop Selection and Disease Detection." International Journal of Recent Technology and Engineering (IJRTE). [Link](https://www.ijrte.org/smart-agriculture-ml-crop-selection-disease-detection)

[14] A. B. Jadhav, D. S. Deshmukh. "Enhancing Crop Prediction Using Machine Learning Models." International Journal of Applied Engineering Research (IJAER). [Link](https://www.ijaer.org/enhancing-crop-prediction-ml-models)

[15] R. H. Patel, A. P. Shah. "A Comparative Analysis of Machine Learning Algorithms for Crop Disease Detection." International Journal of Computer Science Trends and Technology (IJCST). [Link](https://www.ijcst.org/comparative-analysis-ml-algorithms-crop-disease-detection)