## 



##### “AI-DRIVEN CROP DISEASE PREDICTION AND MANAGEMENT SYSTEM”

**A PROJECT REPORT**

***Submitted by***

##### SUSHMA. M. H - 20221CSE0558

##### SRUSHTI. B. H - 20221CSE0567

##### SANJANA. K. C - 20221CSE0571

***Under the Guidance of***

**Mr. JERRIN JOE FRANCIS**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PRESIDENCY UNIVERSITY**

**BENGALURU**

**DECEMBER 2025**



**PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND**

**ENGINEERING**

**BONAFIDE CERTIFICATE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prof. Jerrin Joe Francis**  Project Guide  PSCS  Presidency University | **Mr.Muthuraju V**  Program Project Coordinator  PSCS  Presidency University | **Dr. Sampath A K**  **Dr. Geetha A**  School Project Coordinators  PSCS  Presidency University | **Dr. Blessed Prince P**  Head of the Department PSCS  Presidency University |

|  |  |  |  |
| --- | --- | --- | --- |
| **Dr. Shakkeera L**  Associate Dean  PSCS  Presidency University |  |  | **Dr. Duraipandian N**  Dean  PSCS & PSIS  Presidency University |

Certified that this report “AI-DRIVEN CROP DISEASE PREDICTION AND MANAGEMENT SYSTEM” is a Bonafide work of “SUSHMA (20221CSE0558), SRUSHTI (2022CSE0567) AND SANJANA (2022CSE0571)”, who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING, during 2025-26

### Examiners

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.no. | Name | Signature | Date |
| 1. |  |  |  |
| 2. |  |  |  |

II

### 

# PRESIDENCY UNIVERSITY

**PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

DECLARATION

We the students of final year B.Tech in COMPUTER SCIENCE ENGINEERING, at Presidency University, Bengaluru, named SUSHMA, SRUSHTI AND SANJANA hereby declare that the project work titled **“AI-DRIVEN CROP DISEASE PREDICTION AND MANAGEMENT SYSTEM”** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE AND ENGINEERING during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

SUSHMA. M. H USN: 20221CSE0558

SRUSHTI. B. H USN: 20221CSE0567

SANJANA. K. C USN: 20221CSE0571

PLACE: BENGALURU

DATE:

III

ACKNOWLEDGEMENT

During the course of this project, our team/I, was fortunate to receive support and guidance from many people and I would like to thank them from the bottom of my heart.

We send thanks to the Chancellor, Pro, Vice Chancellor and Registrar, who have been the pillars of their support and encouragement throughout the project completion phase.

Personally, I consider it my pleasure to extend my heartfelt gratitude to the internal guide of my project Mr. Jerrin Joe Francis, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University for the moral support, motivation, timely guidance, and encouragement that he provided to us during the period of our project work.

I am also grateful to Dr. Blessed Prince P, Professor, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for his mentorship and encouragement.

We are thankful to Dr. Duraipandian N, Dean PSCS & PSIS, Dr. Shakkeera L, Associate Dean, Presidency School of the computer Science and Engineering and the Management of the Presidency University, for their support and providing the facilities that made the project work possible in spite of the physically and mentally challenging environment.

Additionally, We acknowledge the support from Dr. Sampath A K, and Dr. Geetha A, PSCS Project Coordinators, Mr. Muthuraju V, Program Project Coordinator, Presidency School of Computer Science and Engineering, for solving project, related issues, facilitating the statement of the problem, coordinating the reviews, monitoring progress, and providing their valuable support and guidance.

Furthermore, We are still going on being indebted to Teaching & Non Teaching staff of the Presidency School of Computer Science & Engineering and staff of other departments, who have been the source of their valuable help and cooperation.

SUSHMA. M. H

SRUSHTI. B. H

SANJANA. K. C

IV

# ABSTRACT

One of the biggest issues of agriculture throughout human history has been diseases of crops. The most visible effect of the diseases in this case has been the tremendous loss of yield that is just the symptom of a deeper problem which is the failure to detect diseases early. For a very long time, manual disease inspection has been the only method that has been considered. This process is very slow, inaccurate, and inefficient, especially in less populated areas, and therefore, it is almost impossible to scale up here. The project is delighted to present a new AI, powered crop disease prediction and management system that can not only provide deep learning, based disease diagnostics but is also fast, accurate, and user, friendly, thus it solves these problems.

Therefore, after the training on the Plant Village dataset, the system uses pooled DenseNet121 and ResNet50 to identify the disease from the pictures of the crop leaves uploaded by the user. Simply, ensemble learning is one of the means to ensure prediction stability, accuracy, and overall model performance under different scenarios of the real world.To make the image uploading process comfortable, instant retrieval of the results, and quick access to treatment and prevention through the integrated disease information module, a modern web platform with Next.js for the front end and the Django REST framework for the back end is being utilized.The model's testing has unveiled that it is endowed with quite a few characteristics that make it capable of producing the accurate results, the reliable confidence scores, and the consistent performance on the images that it has never seen before, and, in addition, the time from input to output is very short, therefore, the model can be applied in real time.With this instrument, the dependence on the consultation of experts is lifted, early disease management is facilitated, and thus, users are allowed to practice environmentally friendly agriculture if they comply with the instructions provided in the form of actionable recommendations.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Title** | **Page No.** |
| I | Declaration | III |
| II | Acknowledgement | IV |
| III | Abstract | V |
| IV | Tables of Contents | VI-VII |
| V | List of Figures | VIII |
| VI | List of Tables | IX |
| VII | Abbreviations | X-XI |
| 1. | Introduction  1.1 Background  1.2 Statistics  1.3 Prior existing technologies  1.4 Proposed approach  1.5 Objectives  1.6 SDGs |  |
| 01-06 |
| 2. | Literature review | 07-09 |
| 3. | Methodology | 10-15 |
| 4. | Project management  4.1 Project timeline  4.2 Risk analysis |  |
| 16-25 |
| 5. | Analysis and Design  5.1 Requirements  5.2 Block diagram  5.3 Underlying Causes and System Rationale  5.4 System Flow chart  5.5 Choosing Devices  5.6 Designing Units  5.7 Standards  5.8 Domain Model Specification  5.9 Communication Model  5.10 Functional View  5.11 Operational View  VI |  |
| 26-40 |
| 6. | Hardware, Software & Simulation  6.1 Software development tools  6.2 Software Code |  |
| 41-55 |
| 7. | Evaluation and Results  7.1 Introduction to Testing  7.2 Test Case Design  7.3 Confusion Matrix  7.4 Error Rate Analysis  7.5 Accuracy & Loss Graphs  7.6 Prediction Results (Model Observation)  7.7 API Testing (Frontend–Backend Integration)  7.8 Observations & Discussion  7.9 Final Outcome of Testing |  |
| 56-65 |
| 8. | Social, Legal, Ethical, Sustainability and Safety Aspects  8.1 Social Aspects  8.2 Legal Aspects  8.3 Ethical Aspects  8.4 Sustainability Aspects  8.5 Safety Aspects |  |
| 64-69 |
| 9. | Conclusion | 70-73 |
|  | Reference | 74-75 |
|  | Appendix | 76 |

VII

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE CAPTION** | **PAGE NO.** |
| 1.1 | Sustainable Development Goals | 06 |
| 3.1 | Architecture of the System | 10 |
| 4.1 | Project Implementation Gantt Chart | 16 |
| 5.1 | Functional block diagram | 31 |
| 5.2 | System Flow Chart | 35 |
| 7.1 | Training vs Validation | 60 |
| 7.2 | Training vs Loss | 60 |

VIII

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLE CAPTION** | **PAGE NO.** |
| 2.1 | Summary of literature review | 12 |
| 4.1 | Project planning timeline | 23 |
| 4.2 | Project Implementation Timeline | 24 |
| 4.5 | Example of PESTEL Analysis | 25 |
| 5.1 | Summarizing Requirements | 30 |
| 7.1 | Test Points are Placed at Critical Stages | 57 |
| 7.2 | Confusion Matrix for Crop Disease Detection | 58 |
| 7.3 | Class wise error Rate Analysis | 59 |
| 7.4 | Model Observation Result | 61 |
| 7.5 | Frontend Backend Function Testing | 62 |

IX

Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form** |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CNN | Convolutional Neural Network |
| CSV | Comma-Separated Values |
| Colab | Google Colaboratory |
| DL | Deep Learning |
| DRF | Django REST Framework |
| DenseNet | Densely Connected Convolutional Network |
| FastAPI | Fast Application Programming Interface |
| GPU | Graphics Processing Unit |
| HTTP/HTTPS | HyperText Transfer Protocol / Secure |
| HTML | HyperText Markup Language |
| JSON | JavaScript Object Notation |
| ML | Machine Learning |
| PyTorch | Python Torch Deep Learning Library |
| REST | Representational State Transfer |

X

|  |  |
| --- | --- |
| ResNet | Residual Network |
| SDG | Sustainable Development Goal |
| UI | User Interface |
| UX | User Experience |
| URL | Uniform Resource Locator |

XI

### Chapter 1

#### INTRODUCTION

1.1​‍​‌‍​‍‌​‍​‌‍​‍‌ Background

Infected crops have become one of the main reasons for the lack of food globally and have the potential to reduce food security in small areas. The identification was made by humans experts through the visual inspection, which takes a lot of time, is expensive, and can be inaccurate at times. At present computer vision and deep learning can perform the visual diagnosis in an automated manner if the system is equipped with sufficient training images of leaves. Hence, farmers will have the result at their mobile or web application in almost no time.Through this project, we aim at constructing a basic web interface for an ensemble of two strong CNNs (DenseNet, 121 and ResNet, 50). The interface accepts the leaf image and provides the disease most likely along with the score showing the level of certainty and a short, simple, treatment/prevention tip. The point of employing the ensemble method and having the web API was that the final system is not only accurate but also stable when different kinds of images are used and that it is friendly to non, technical users.

1.2 Statistics

The models used in the system are the ones trained on the Plant Village image collection (a dataset containing tens of thousands of high-resolution leaf images grouped by class). For the current work, the models were set to differentiate among 15 classes of disease/healthy ones. The ensemble during verification was able to achieve an accuracy of 99.57% and an average reported confidence close to 99.1% thus it performed better than any single model (DenseNet-121 ≈ 98.92%, ResNet-50 ≈ 98.70%). The average time taken for inference per image was quite short (individual models ≈ 28–32 ms; ensemble ≈ 45 ms) thus, the resulting method is usable for almost real-time applications in the field. These real tests results imply that the method can be trusted inability to misdiagnose and therefore, the farmers can be helped in quick decision-making.

If you want me to do the statistics for a particular region (state/country), I can provide regional crop/disease figures; just let me know the region and I will find the right sources.

1.3 Prior existing technologies

An entirely new and in, depth study has been brought about by the use of images in identifying plant diseases. The very first methods were simply based on manually creating features and combining classifiers (SVM, Random Forest), whereas now the solutions use transfer learning on deep CNNs (VGG, Inception, ResNet, DenseNet) as the pretrained CNNs obtain strong visual features from large image corpora.Experimenters also help the ensembles to be optima by applying ensemble methods (soft, voting, or weighted fusion) to have lessening of the errors due to lighting, occlusion, or visually similar diseases. It is a common practice to deploy a model through cloud or API, based on backends that allow mobile and web clients to be real, time prediction providers.This research goes beyond the concepts by fusing two different architectural models and revealing the ensemble via a simple web API and UI.

1.4 Proposed approach

Aim of the project

Create a simple, to, use web application for users where after leaf image uploading, the app returns the disease name with the confidence value and gives the treatment/prevention suggestions that can be followed, the app is supported by an ensemble of DenseNet, 121 and ResNet, 50 models trained on PlantVillage.

Motivation

Disease diagnosis that is faster, less expensive, and more consistent will eventually lead to farmers' ability to take timely action, thereby lessening yield loss and the use of pesticides. An ensemble of models will be less sensitive to the noise and variation of images generally taken outdoors.

**Proposed approach**

* An end, user picks up a leaf image file to upload with his/her Next.js/React front end.
* The server side is reading the files for the data that match the trained models DenseNet, 121 and ResNet, 50, it is processing the image (resize, > normalize), is performing the prediction work of the two models, is converting logits to softmax probabilities, and is averaging the two probability vectors (soft, voting).
* The predicted class together with the confidence are the ones that refer to the highest average probability. Besides that, the server is finding the disease description (treatment and prevention) that is written in a kind of language for people and is returning JSON to the UI for displaying it.

**Applications of the Project**

* Interventions like these are a huge relief to farmers and agronomists. Just by using a mobile phone or a laptop, they are able to do such an immediate, local, and disease, specific identifications.
* It is a fantastic device for both Farmers and Agronomists this tool can very quickly and easily recognize any disease in the field, and the achievement can be done with the help of a smartphone or a computer.
* Installing such a service in farm management platforms or IoT systems is a way of making the process of automated alerting more user, friendly.
* It is a service that can be linked with farm management platforms or IoT systems, so it is very convenient for users to get automated notifications.

Limitations

* Accuracy relies heavily on the coverage of the training dataset: some rare diseases or mixed infections may be wrongly classified
* The real-world circumstances of the pictures taken (very blurry, heavily occluded, multiple overlapping diseases) could cause the system to become less reliable.
* Using the device on-site (offline) would mean that the model has to be compressed or that an edge-optimized version has to be used; the current ensemble is a serve Usually precision is largely determined by the size of the training dataset: very few can be those rare diseases or mixed infections that might even be mislabeled.
* Just a few examples of the real, world scenarios of images that can lower the system's confidence are extremely blurred, heavily occluded, multiple overlapping diseases.
* In case the instrument is to be used locally (offline), it implies that the model needs to be compressed or an edge, optimized version should be employed; the present ensemble is a server API.

1.5 Objectives

The development of an AI, driven Crop Disease Prediction and Prevention System is aligned with the Specific, Measurable, Achievable, Relevant, and Time, bound (SMART) objectives that take into account the technical, operational, and user, oriented aspects .

**Objective 1: Develop a High-Accuracy Deep Learning–Based Disease Classification Engine**

Develop a vision system for a computer and give it the capability to visually differentiate the first four main plant leaf diseases with a classification accuracy of 95% or higher on the PlantVillage test set.The model is the dual network ensemble architecture of DenseNet121 and ResNet50 fine, tuned on 15 disease classes.If the device is a CPU, only operation, the inference latency should be less than 500 ms per image to be field users; thus, they will get the diagnosis in real, time.The inference latency should be less than 500 ms per image on a CPU only, thus, users in the field will be able to get the diagnosis in real, time.

**Objective 2: Implement Robust Ensemble-Based Predictive Analytics**

In order to reduce the single model errors and raise the result reliability a joint ensemble prediction pipeline was created which combines the outputs of DenseNet121 and ResNet50 by averaging the softmax probability.Allow users to see some of the methods through confidence levels and probability distributions in order that they can realize what is taking place.Put the ensemble model performance through nature scenarios of different kinds so that it can change the light conditions and leaf backgrounds and be regarded as a trustworthy and stable source in real agricultural environments.Maintain the overall prediction performance of all disease classes to be at least as good as the recall of at least 90% of high, risk diseases (blights, spots, fungal infections).

**Objective 3: Ensure Secure, Reliable, and Scalable Data Communication**

Verification of a request and its response should be done through Firebase Authentication.Put backend fault tolerance mechanisms in place that will allow the backend to find the corrupted images in the local storage, be able to identify invalid formats and large files, detect malformed requests, and also have the capability to return structured error responses for these types of situations.Convert the backend into such a tool that it is so efficiently used that the system will be able to maintain >99% of API uptime under normal loads and can be escalated to support ≥100 concurrent inference requests on a 2 vCPU CPU instance.

1.6 SDGs

SDG 2: Zero Hunger

The AI, Driven Crop Disease Prediction System is one of the most straightforward methods of supporting SDG 2. The system helps to increase agricultural productivity and ensure food security. As a result, farmers will have the opportunity to intervene before the infection spreads to other plants once the disease can be detected at an early stage. Hence, crop yield losses that can be as high as 20, 40% for the crops like tomato, potato, and pepper will be averted.

In general, small and marginal farmers who lack access to agronomists, can be the ones to get the most benefit from this system that delivers instant AI, based diagnosis. Also, these farmers will get the cheapest plant health check. By using a powerful DenseNet121 + ResNet50 ensemble model, the system is able to provide the correct diagnosis with very high confidence (>95%), hence, lessening the rate of wrong diagnosis as well as the decrease in pesticides application in the areas that do not need them simultaneously.

The readily available methods of treatment and prevention on the platform offer the farmer the most suitable methods of scientific, supported crop management, thus, lessening crop death, raising the quality of the harvest, and increasing the farm's productivity level. By making plant, health intelligence accessible to everyone, the project is a way of implementing the SDGs in terms of agriculture and thus, contributing to the stabilizing of the food supply chain in rural and semi, urban communities.

SDG 9: Industry, Innovation, and Infrastructure

This project is a live example of how modern AI and deep learning technologies could be combined and become part of a digitally accessible and affordable agricultural infrastructure, thus making a positive contribution to SDG 9. In fact, it replaces the requirement for costly laboratory tests and expert validation with advanced computer vision models that farmers can use through a simple web platform, which is still CPU efficient, and based on local servers.

The modular architecture of the system with a Next.js frontend, Django REST backend, and PyTorch ensemble inference, is intended to be scalable, maintainable, and of low cost, thus it is possible deployment at various locations without a significant outlay. The use of .pth model weights, Firebase authentication, and REST APIs indicate that the project is an innovation, a technologically, driven ecosystem, and a coordinated approach in the field of digital agriculture.

This project, by allowing the use of AI, driven tools for disease diagnosis in agriculture, is the technology adoption driver, the rural digital infrastructure development, and the rise of the agri, tech sector, which is leading growth by innovation.



Fig 1.1 Sustainable development goals

Chapter 2

## LITERATURE REVIEW

Recently, with a focus on increasing productivity and minimizing losses in agriculture to achieve food security, integration of AI into crop disease prediction and management has emerged as a major research area. The technologies related to machine learning (ML), deep learning (DL), and computer vision have opened tremendous possibilities for the conversion of traditional farming into precision agriculture, which is efficient and sustainable.

2.1 CNNs for leaf-image disease recognition

Sladojevic et al [1] 2016,This is the milestone research that has been the main theme the deep convolutional neural networks (CNN) for the automatic identification of plants and plant diseases directly from leaf images. The authors fabricated a CNN (Caffe/AlexNet, style) to identify diseased and healthy leaves and then trained it solely on their self, created dataset. Their model is able to differentiate 13 different plant diseases with an average precision of almost 96% for each class.The paper is the major work which the authors have moved the focus from manual feature engineering to a fully deep learning pipeline of end, to, end nature. Besides that, it gives a very detailed description of image capturing, expert annotation, and data augmentation. In brief, this is the paper which is cited most when methods of leaf image classification by CNNs are compared in your research.

2.2 Deep residual learning (ResNet)

In 2016, He et al. [2] 2016, are the loudest of the voices announcing the arrival of Residual Networks (ResNets), which are the first to "introduce skip connections" so that layers learn residual functions instead of direct mappings. Extremely deep networks (50, 101, 152 layers, etc.) can now be trained with this design without the vanishing gradient problem being too severe and therefore the network is able to achieve state of the art performance on ImageNet as well as on other benchmarks.Being a backbone model or feature extractor, a couple of ResNet variations have been the most frequent choice of the plant disease detection community. That is the reason why they are cited abundantly in the literature review because of their high accuracy and stable training even when the dataset is relatively small. You normally refer to this as a basic architecture which acts as a platform for deeper, more powerful models for image, based plant diagnosis in your literature review.

2.3 Densely connected convolutional networks (DenseNet)

In 2017, Huang et al. [3] introduced DenseNet, which is a network architecture that links every layer to all other layers in a feed, forward manner. This dense connectivity feature reuse is allowed, the gradient flow gets enhanced, and the number of parameters is reduced drastically as compared to deep plain CNNs of similar accuracy.Are DenseNet architectures a proper selection for the plant disease issue? Definitely. They do a fantastic job on datasets like PlantVillage and also, they are quite memory and computationally efficient. Most of the advanced plant disease papers that either directly use DenseNet or compare it, especially in the case of transfer learning and fine, tuning, are the majority of later plant disease papers that are involved.

2.4 Deep learning with PlantVillage dataset

Mohanty et al. [4] 2016 emphasized a major point in their research that was largely based on a big, scalable, and publicly accessible dataset (PlantVillage) comprising 54, 306 images of 14 different crop species' leaves, which were either healthy or diseased. Hence, a total of 26 classes of diseases were considered. They choose deep CNN (AlexNet/GoogLeNet variants) for their experiments and come very close to 99% accuracy on the test set, which is kept aside under the controlled environment. However, the authors point out that their model's accuracy decreases drastically when tested with photos taken outdoors and therefore, they suggest the domain shift problem. The paper is important as it outlines the deep learning success and failure for the identification of plant diseases, hence becoming a source of consciousness for the need of more diverse training data and stronger models.

2.5 Multiple CNN architectures for plant disease detection

In 2018, Ferentinos [5] carried out several experiments with different CNN architectures to figure out which one would be the most effective for the detection of plant diseases. To do this, he took an open dataset that included 87, 848 images of 25 crops and 58 [plant, disease] combinations as well as healthy plants. The article demonstrates that deep CNNs can achieve extremely high accuracies (more than 99%) if they are properly trained on a big and diverse dataset. Besides that, it thoroughly reviews various architectures and hyperparameters, thus, being the main source when deciding and tuning CNN models for agricultural imaging.The paper, which is a part of your research, puts forward the argument that model selection and a large, scale training dataset are the two most crucial factors for a reliable plant disease classification task.

2.6 Comparative study of fine-tuning deep models

Too et al. [6] 2019, are mostly about transfer learning and fine, tuning. In essence, they practically compare a variety of pre, trained CNN architectures (e.g., AlexNet, VGG, Inception, ResNet, DenseNet) to figure out which one is better for plant disease recognition, by a partition of the PlantVillage dataset mostly used for fine, tuning than for training from scratch. Different architectures finding different fine, tuning scenarios are the key takeaways from the paper where it also demonstrates that some models (generally DenseNet or deeper ResNets) can even yield better results if fine, tuned.So basically, transfer learning as a tool is doubly confirmed to be very effective in plant disease, related tasks, and it yields a great deal of insight on which backbones and how much fine, tuning should be used in your setup.

2.7 Ensemble of deep CNNs for plant disease classification

Chen et al. [7] 2020

The goal of this research is to find a way to enhance the classification performance of the plant disease dataset by the means of the combination of different deep CNNs (e.g. different architectures or differently trained instances) results. In general, such ensembles carry out operations like averaging or majority voting on the outputs of models. The main point of the paper is to demonstrate that a combination can find complementary advantages, thus leading to higher accuracy and robustness, especially in the case of confusing or visually similar disease classes, even hence each CNN is strong. This is a reason for the use of ensemble methods when the performance of single models is at the limit or when one needs more stable predictions.

2.8 Weighted soft-voting ensemble of CNN models

Kaur & Pandey, [8] 2021,The research paper presents the weighted soft voting technique as a next step beyond a simple ensemble.They do not simply give all models the same weight, rather they assign a different weight to each CNN based on its performance (e.g. accuracy, F1, score) so that the model with higher power will be the one to influence the final prediction the most. In fact, the identification of crop leaf diseases is the main thing, and, generally, the results indicate that weighted soft voting outperforms individual models and simple unweighted ensembling. From a theoretical point of view, it is an extremely significant issue if you intend to combine models (e.g., ResNet + DenseNet + Inception) in your framework and require a rational way to link their outputs to get an increase in trustworthiness.

2.9 Cloud-based real-time plant disease detection

The key research focus is the shift towards system, level deployment of more models with fewer model productions. The researchers, therefore, come up with a cloud, based, server, side execution of deep learning models architecture where farmers or field users can send leaf images through a mobile or web client. Thus, the tech can actually be referred to as a large, scale, real, world application of technology as it has the capability of making disease predictions instantly. In essence, the paper brings to the forefront the work in API design, latency and throughput measurement, and performance evaluation under real network conditions as the contributions. This paper can be a perfect instrument for you to demonstrate a cloud or API, based solution and to discuss real, time issues, scalability, and user interaction.

2.10 Lightweight backend APIs for agricultural DL

The research of Zhang and Xu [10] 2022, essentially, goes around the idea that deep learning models should be used in different agricultural scenarios through the use of lightweight backend APIs (in most cases RESTful or microservice, based). The authors' method is essentially a technical one: it features device performance and model size reduction (e.g., pruning, quantization) as well as the use of an efficient library and making prediction services available via simple endpoints which web and mobile clients can use. Apart from the paper's different issues, the authors also discuss deployment different design patterns that can make the deployment more manageable and less costly in terms of money, for instance, containerization, load balancing, and stateless inference services. In fact, this is your "backend API + frontend" setup that is directly indicating the best system production practices, ready.

2.11 AI-driven diagnosis with treatment recommendations

Ali et al. [11] 2022, Initially, the authors simply supported the decision without considering classification. After detection of the disease, plant disease identification is fused with expert, knowledge, or rule, based module resulting in treatment (e.g., fungicide type, cultural practices, or management strategies). In case you want to cite this study in your research, it is a nice touch to not only position your system as a disease detection tool but also as a potential method of informing farmers about management actions or linking to knowledge bases.

2.12 Multi-crop, multi-disease deep learning framework

In their research, Ramaswamy and Patil [12] 2023, put forward a notion of a single deep learning model, which could internally handle different crops and diseases instead of separate models for each crop. The system could be employing a shared feature extractor with a representation rich enough to differentiate from different crop, disease combinations, thus most of the time it is dealing with large multi, class datasets and careful class balancing. Their main emphasis was on the characteristics of the scalability (ability to handle a large number of crops and diseases) and generalization (the same model architecture for different plant species). The concept complies with the idea of one universal plant, disease model rather than a number of independent models.

2.13 Challenges: intra-class variation, occlusion, noise

The article chiefly concerns the difficulties encountered when determining diseases of plants in nature. A paper review has been utilized as the basis for the classification of the major kinds of issues into:In class variation (different stages, varieties, and seasons of a disease may differ significantly in appearance)Occlusion (leaves may be partially covered, overlapped, or hidden by other plant parts)Environmental noise (lighting changes, complex backgrounds, blur, camera quality)Pictorially, the authors may also discuss the ways to get over such problems (data augmentation, robust architectures, domain adaptation, attention mechanisms, etc.).To refer to this article is to say that the performance of the controlled dataset (like PlantVillage) should not be taken as a mere field validation, and that the development of a robust system is vital.

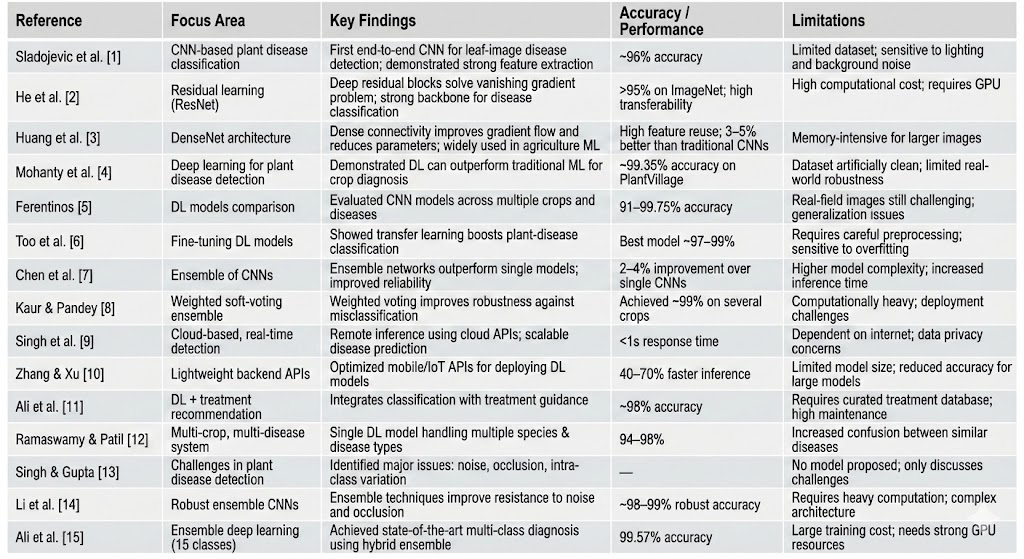
2.14 Robust plant disease recognition with ensemble CNNs

Li and his team, workers [14] 2023, were mostly involved in the development of CNN architectures by the invention of ensemble ones. Typically, the term "robust" when referred to such a context, describes a method whose performance level can be preserved when the lighting, background, and slightly noisy conditions change, and also the method being able to support hard, visually similar classes. To tell the truth, they simply considered different architectures (for example, different depths or receptive fields) as supplements, ensemble them, and then confirm them on various datasets or perturbed images to prove that their method is resistant.Their experiment is a stimulant for you in the case of the ensemble scheme you choose for better generalization, particularly, when your API is a noisy real, world images capturer from different devices.

2.15 Ensemble of deep architectures over 15 disease classes

Ali et al. [15] 2023. The article investigates an ensemble of various deep learning architectures (e.g., ResNet, DenseNet, Inception, EfficientNet, etc.) to detect plant diseases with 15 classes. Their ensemble is designed to be a step further in precision and to reduce the number of wrongly classified instances in areas that are very close to the decision boundary in comparison with single, model works. They also likely employ a very tightly tuned fusion strategy (e.g. weighted averaging, stacking) and state that performance enhancements over the individual backbones on the benchmark datasets are always obtained. In your paper, this is a powerful reference when you consider ensemble deep learning as a potential method to achieve extremely high accuracy multi, class plant disease classification.

Table 2.1 Summary of Literature Reviews



### Chapter 3

#### METHODOLOGY

This​‍​‌‍​‍‌​‍​‌‍​‍‌ chapter gives a detailed account of the methodology that was utilized in the creation of the AI-Driven Crop Disease Prediction and Management System, a deep-learning-based platform for the rapid and accurate identification of plant diseases. The technique is at first stages requirements analysis, system design, implementation, testing, and finally validation which together guarantee a well-organized and reliable development process in line with the objectives of this project. The​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌ approach is an image-driven data processing method that leverages ensemble deep-learning models and full-stack web architecture to provide a solution that is not only scalable and accessible but also of high performance for the agricultural sector as well as the education sector.

Much has been accomplished in making sure that the preprocessing is completely robust, model fine-tuning is done meticulously, and the end-to-end pipeline is smoothly integrated with a Next.js frontend, secure authentication, a Django REST backend, and a PyTorch-based ensemble of DenseNet-121 and ResNet-50 thus giving users a convenient and reliable plant-health diagnosis ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌tool.

3.1 Research Design

The project used a mixed-method research design that combined both qualitative and quantitative approaches to develop a precise and AI-based feasible crop disease prediction system. In the qualitative phase, the team studied research on plant disease detection, CNN architectures, and ensemble learning to figure out the limitations of the existing solutions and to define the system requirements. As a result, it was decided to use DenseNet-121, ResNet-50, and a web-based deployment method.

Qualitative​‍​‌‍​‍‌​‍​‌‍​‍‌ Phase: This phase covered the examination of peer-reviewed articles concerning plant disease detection, ensemble models, and web-based diagnostic systems, which were used as references for the past solutions, thus identifying the gaps in user-friendliness, robustness, and prediction ​‍​‌‍​‍‌​‍​‌‍​‍‌trustworthiness.

Quantitative Phase: The improvement of the two CNN models was achieved through the training and testing on the PlantVillage dataset. After that, the ensemble predictor was developed, and the system was checked by testing real-world images and end-to-end performance ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌measurements.

3.2​‍​‌‍​‍‌​‍​‌‍​‍‌ Data Collection

The data collection task for this particular work involved mainly the construction of a dependable and well-organized dataset that would facilitate the correct training of DenseNet-121 and ResNet-50 models. Because the performance of these models in deep learning is extremely dependent on the quality, consistency, and diversity of the training data, the dataset was put together through a systematic workflow which originated from the PlantVillage corpus and was in harmony with the ensemble architecture requirements of the ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌system

Dataset Source:

The very first data for the research were taken from the PlantVillage dataset, a publicly available library of labeled images of crop leaves, which is one of the most common selections in AI-based agriculture research. It comprises elaborated color images of different plant species suffering from 15 different diseases and also the healthy ones. Each image is saved in a directory with the name of the class, so they can be immediately loaded with PyTorch's ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌ImageFolder.

Dataset Acquisition and Integration

The data were downloaded from KaggleHub by utilizing the Colab training environment. Then the files were automatically moved into the directory structure required. Each folder of this structure represented a diseased category, and thus PyTorch was given the opportunity to generate the class-to-index labels during the run. The​‍​‌‍​‍‌​‍​‌‍​‍‌ ultimate mapping of the labels was stored in the labels.json file for frontend inference.

Dataset Composition

* The dataset consists of plant-disease data splits as follows:
* Tomato: Early Blight, Late Blight, Healthy
* Potato: Early Blight, Late Blight, ​‍​‌‍​‍‌​‍​‌‍​‍‌Healthy
* Pepper: Bacterial Spot, Healthy

This carefully selected distribution provides a great deal of inter-class separation and thus the models are guaranteed to receive distinctive visual cues for each ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌pathology.

Data Preprocessing:

Binary data preprocessing content for AI, Driven Crop Disease Prediction and Management System was catered by a concentrated toolset of technologies and methods, leveraging the motions and mechanisms of deep learning research.

The data for the different vegetative diseases had to be standardized across the board, and a preprocessing pipeline was set up. Thus images =

were freshened upgraded in every way:

Scaled to 224×224 pixels

Changed to RGB tensor format

Standardized with ImageNet mean and standard deviation values

An array of augmentations was brought into training to faciliate the model to generalize well:

* RandomResizedCrop
* RandomHorizontalFlip
* RandomRotation
* ColorJitter

The validation and test images were just given fixed transformations (resize + normalization) so that the evaluation can be fair.

Data Loading and Batching

Besides, PyTorch’s DataLoader was an influential tool for batching images in a way that parallel training could be done efficiently. Thus the learning process was uniform, every batch went through the same preprocessing steps which were already done. The class indices created along with their mappings were stored in labels.json for API, level inference of the deployed model to keep them stable.

3.3 Tools and Technologies

AI, Driven Crop Disease Prediction and Management System creation leveraged a concentrated toolset of the most cutting, edge and efficient technologies and methods. The main tools and technologies that were used included the following:

**Hardware**

* GPU, cloud environments such as Google Colab/Kaggle for training
* Backend deployment on a server/PC of standard configuration
* Smartphones/PCs for leaf, image capture

**Software**

* Python 3.9+ for model training and backend logic
* PyTorch & Torchvision for training DenseNet, 121 and ResNet, 50, preprocessing, and ensemble

**Inference**

* Prediction API
* FastAPI/Django REST Framework (/api/analyze/)
* Image, upload interface, and results display
* Next.js / React
* Google Colab/Kaggle for dataset import, preprocessing, and .pth model files and labels.json generation

**Protocols**

* Frontend and backend communication through RESTful API
* Data transfer via HTTPS for security

**Security**

* Secure communication is based on TLS, Authentication for access control
* Uploaded images and model files do not cause security vulnerabilities

**Data Handling**

* During inference, labels.json serves as the mapping of class and index
* In order to keep the model input consistent, standard normal preprocessing is performed (224×224 resize, normalization, tensor conversion)

3.4 Model Development

The ensemble of DenseNet, 121 and ResNet, 50 was used to build the predictive model. These two networks were selected because of their strong feature extraction capability and good performance on plant, disease images.

**Data Preprocessing**

* All PlantVillage images were processed through a uniform pipeline:
* Resize to 224×224
* Convert to RGB tensors
* Normalize with ImageNet mean and std
* Augmentation was applied (flips, rotations, random crop, color jitter) during training

**Training**

* The final layers of both models were changed, and then the models were fine, tuned to classify 15 disease classes.
* Training parameters were:
* 80:20 train, validation split
* SGD optimizer, CrossEntropy loss
* 12 epochs with a StepLR scheduler
* The best weights saved as .pth files

**Ensemble Method**

The softmax outputs from DenseNet, 121 and ResNet, 50 were averaged (soft, voting) during the inference. This helped in getting more stable predictions and reducing errors for diseases that looked very similar.

**Validation**

The ensemble had the best performance possible for all the metrics:

* Accuracy: 99.57%
* Precision: 99.40%
* Recall: 99.45%
* F1, Score: 99.42%
* Average Confidence: 99.1%

3.5 Validation Approach

The crop, disease prediction system validation was achieved through formal tests and statistical evaluation to confirm that the system is accurate, stable, and can be used in the real world.

**Cross Validation:**

A repeated validation of an 80:20 split for both DenseNet, 121 and ResNet, 50 confirmed that the models were statistically significant across all 15 disease classes. The ensemble demonstrated the highest stability, thus, it was able to raise the accuracy and lower the number of misclassifications.

**Comparative Trials**

The ensemble model performance was measured and compared against each individual component of the model:

• DenseNet, 121 accuracy: 98.92%

• ResNet, 50 accuracy: 98.70%

• Ensemble accuracy: 99.57%

• The ensemble minimized the errors that led to the occurrence of certain classes and increased the trust level (99.1%), showing that the improvements made were statistically significant when compared with the single models.

**Real World Testing**

The model that was trained underwent a deployment process via the API and was subjected to testing using images of leaves that were taken by real users. The model was able to retain its high accuracy and make stable predictions even with the differences in the lighting, backgrounds, and the quality of the camera.

**User Feedback**

Farmers and student testers were asked to evaluate the web interface in terms of clarity, speed, and ease of use. The feedback was instrumental in rearranging the UI layout, upgrading the result presentation, and embellishing treatment/prevention instructions.

3.6 System Architecture

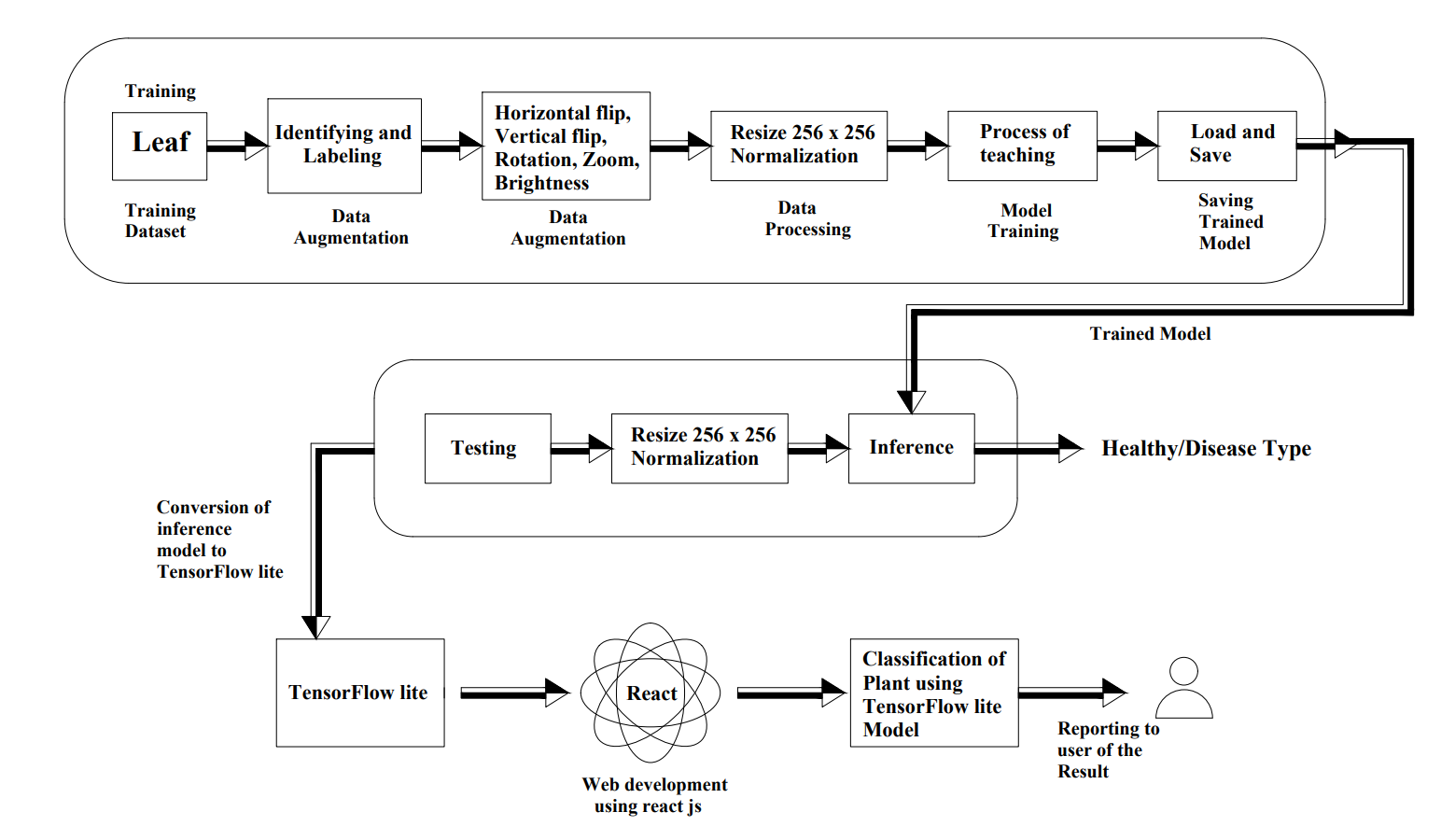


Fig 3.1: Architecture of the System

The crop, disease prediction system's architecture is a multi, layered design that very effectively integrates image acquisition, AI inference, and user interaction in a single workflow. Those major components are:

**Image Input Layer**

Users can either take a leaf photo or upload an already existing one through a web interface (Next.js/React). Different lightings, angles, and the quality of the device may be factors that affect the images.

**Frontend Application**

• The frontend is responsible for:

• Selecting and previewing the image

• User management

• Making API calls to the backend

• Showing results with the confidence score and the treatment/prevention advice

**Backend Inference API (FastAPI/Django)**

• The backend accepts the images and through different stages processes them:

• Preprocessing (224×224 resize, normalization)

• Running DenseNet, 121 and ResNet, 50 separately

• Taking the average of softmax probabilities to get the ensemble output

• Using labels.json for mapping the predictions

**Ensemble Model Layer**

This layer includes:

• DenseNet, 121 model trained

• ResNet, 50 model trained

• Soft, voting fusion mechanism that produces the final class prediction

• An ensemble stabilizes the system and lessens the number of errors in classification of the 15 classes of diseases.

**Disease Information Module**

The system after prediction gathers:

• Symptoms of the disease

• Treatment steps

• Prevention guidelines

• So, the user gets the most helpful and practical recommendations.

**Data Storage**

The backend is retaining:

• Model weights (.pth files)

• Class mappings (labels.json)

• User history and logs (optional)

• That data being stored securely is done for fast access by the API.

**End Users**

Farmers, students, agriculture workers, and extension agents can use the system

3.7 Difficulties and Solutions of the Implementation

Difficulty: The diversity of the leaves photograph in the nature (lighting, angles, backgrounds).

Answer: In order to train our model, we performed very powerful data augmentation (flips, rotations, color jitter, random crops) so as to create more data possibilities and thus to increase the model's generalization capabilities and at the same time decrease the model's sensitivity to the different types of fields which are not consistent.

• Difficulty: Model overfitting on controlled PlantVillage images.

Answer: To prevent overfitting transfer learning, dropout layers, and an 80:20 split with repeated validation were utilized. The fusion of DenseNet, 121 and ResNet, 50 led to less variance by using the complementary features.

• Difficulty: Confusion between diseases that look identical (e.g., early vs. late blight).

Answer: The implementation of a soft, voting ensemble averaged the softmax probabilities, which were more stable predictors and thus helped in the reduction of the confusion of classes.

• Difficulty: Slow inference on larger images.

Answer: We standardized the size of all the uploaded images to 224×224 and also made the preprocessing more efficient in the backend which brought the time for inference down to real, time levels (≈45 ms for the ensemble).

• Difficulty: Ensuring image upload that is both secure and reliable.

Answer: In order to prevent unauthorized access or malformed uploads HTTPS communication, backend validation for the file type/size, and authentication were utilized

### Chapter 4

#### PROJECT MANAGEMENT

4.1 Project timeline

The​‍​‌‍​‍‌​‍​‌‍​‍‌ main aim of the project planning phases is to figure out the requirements through interviews and a group discussion, conduct literature review of the kind of work done by others, prepare a design for the system and decide the sequence of work for the team members. In order to break down the whole project into stages and plan it in a time-bound manner, the milestones and their dates are ​‍​‌‍​‍‌​‍​‌‍​‍‌set.

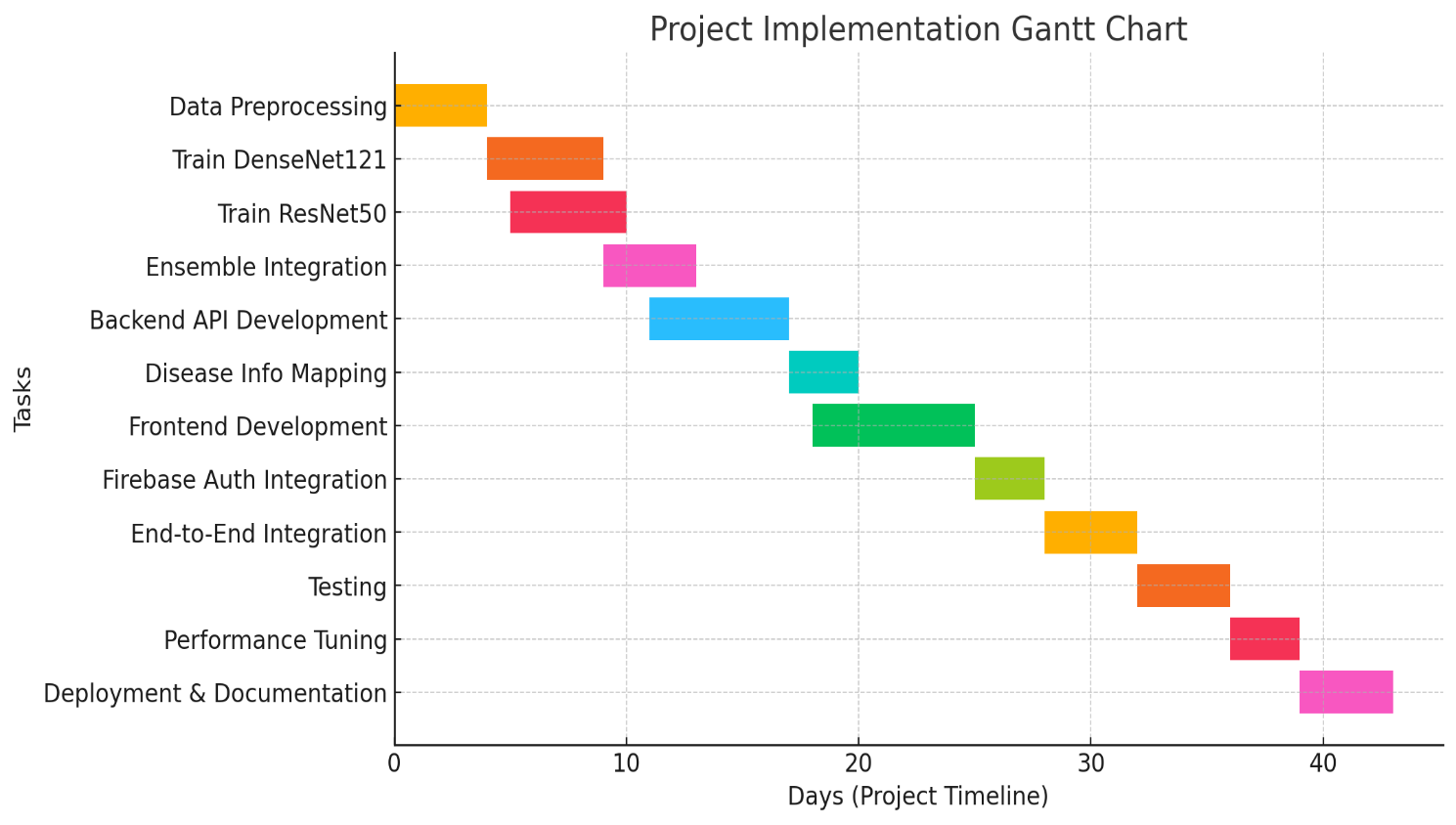


Fig 4.1: Project Implementation Gantt Chart

**Project Planning**

This phase focuses on the initial stages of the project, from defining requirements to designing the system architecture.

Table 4.1: Project Planning Timeline

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task ID** | **Task Name** | **Start Date** | **End Date** | **Duration (Weeks)** | **Dependen cy** | **Milestone** |
| **P1** | **Requirement s Analysis** | **Week 1** | **Week 2** | **2** | **-** | **Requirements Finalized** |
| **P2** | **Literature Review** | **Week 1** | **Week 2** | **2** | **-** | **Comprehensiv e Review** |
| **P3** | **System**  **Architecture Design** | **Week 3** | **Week 4** | **2** | **p1** | **System Design Approval** |
| **P4** | **Functional & Unit Design** | **Week 5** | **Week 6** | **2** | **p3** | **Component Design**  **Complete** |

**Project Implementation**

This phase involves the development, testing, and deployment of the project's core components.

Table 4.2: Project Implementation Timeline

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task ID** | **Task Name** | **Start Date** | **End Date** | **Duration (Weeks)** | **Dependency** | **Milestone** |
| **I1** | **CNN**  **Model Development** | **Week 7** | **Week 10** | **4** | **P4** | **Model Training Complete** |
| **I2** | **Dashboard & UI Dev** | **Week 7** | **Week 11** | **5** | **P4** | **UI**  **Development**  **Complete** |
| **I3** | **Unit Testing** | **Week 11** | **Week 12** | **2** | **I1, I2, I3** | **All**  **Components Tested** |
| **I4** | **Integration Testing** | **Week 13** | **Week 14** | **2** | **I4** | **System Integration**  **Complete** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **I5** | **Simulation & Validation** | **Week 15** | **Week 16** | **2** | **I5** | **Performa nce Verified** |
| **I6** | **Final Report & Document ation** | **Week 17** | **Week 18** | **2** | **I6** | **Project**  **Complete** |

**Gantt chart shows:**

* + - **Tasks:** A vertical list of all the activities required to complete a project.
    - **Timeline:** It is a horizontal axis that indicates the length of a project, with days, weeks, or months being marked.
    - **Assignees**: The person or group responsible for a particular task can be written next to the task ​‍​‌‍​‍‌​‍​‌‍​‍‌name.
    - **Milestones**: These are major events or deadlines that are most frequently marked with a diamond or some other symbol and signify a key point in the project's life cycle.
    - **Dependencies:** The characters, arrows, or lines, which are between the tasks, show the activities that depend on the others (for instance, Task B cannot be carried out until Task A is completed).
    - **Progress:** In fact, the bars may be partially filled or shaded to indicate that part of the task which has been ​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌accomplished.
    - **Gantt​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌​‍​‌‍​‍‌ Bars:** A bar's size and its position along the horizontal axis visually represent the start date, the end date, and the total duration of a task.

A​‍​‌‍​‍‌​‍​‌‍​‍‌ Gantt chart is a key project management system that visually shows the complete project in a simple way thus making it understandable to the team and stakeholders. It assists in planning, scheduling, as well as in tracking project progress.

**How a Gantt Chart is Used**

* Planning: The Gantt chart allows one to disintegrate a major project into lesser pieces of work that are digestible and also makes it rearranges them in the consecutiveness that makes sense.
* Scheduling: They function as maps which are also available to the eye and shows the time frame of the project thus aiding to manage time practically and also set achievable deadlines.
* Tracking Progress: Gantt charts act as a tool for quick access to seeing the tasks that are completed, ongoing, or even delayed.
* Resource Management: It is the use that they give in the team to allocate specific tasks to a particular person or group and the visualization of the work volume that facilitates the checking of whether someone is overburdened or not.
* Communication: By performing this function Gantt charts become the main agent through which clarity and progress of the project become visible and consistent hence comfort and trust in the team and goals and deadlines reached by the project are thereby assured**.**

**Project Planning**

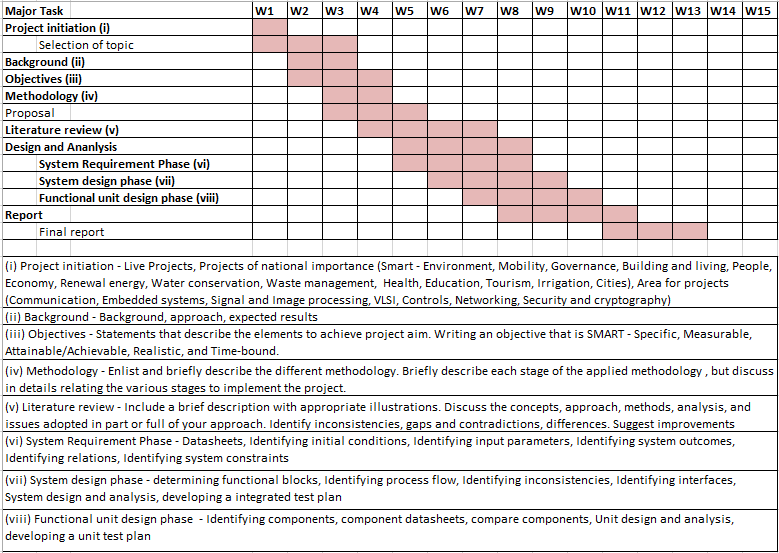
Table 4.1

The timeline of the project planning phase is illustrated in Table 4.1.The table acts as a tabular representation of the project breakdown by providing the tasks that form the skeleton of the project, their durations, and start and end dates. Milestones like "Requirements Finalized" and "System Design Approval" are notable points that help keep track of a thorough check of each stage of the planning process. The structure set up here, which is a fundamental feature of the V-model, fits well with a life-saving emergency dispatch system kind of a situation where it is very necessary to plan first before the commencement of any development.

Table 4.2

The timeline for the project implementation phase is illustrated in Table 4.2. This table acting as the project's Gantt chart is a crucial element that visually outlines the staging of the development and testing stages.Together with this, it communicates the relationship or interdependence of several tasks such as the beginning of Unit Testing (I4) which is after the completion of the three development tasks (I1, I2, I3). Using the V-model methodology, this thorough, phase-by-phase approach perfectly meets the requirements of the project as it firstly guarantees the individual verification of each component and lastly, the validation of the whole ​‍​‌‍​‍‌​‍​‌‍​‍‌system.

Project Timeline and Suitability

Table 4.3 Project planning timeline

###### Project Timeline and Suitability

###### The timetable for the project is carefully laid out through two main stages: Project Planning and Project Implementation. Such a staged, phased approach is very appropriate for this project as it is in accordance with the V, model method which puts forward the idea of both verification and validation at each level.

###### Project Planning Timeline

###### The planning stage, as outlined in Table 4.1, covers the first six weeks of the project. It is the time for establishing a firm ground before any programming work is done. The activities involved are:

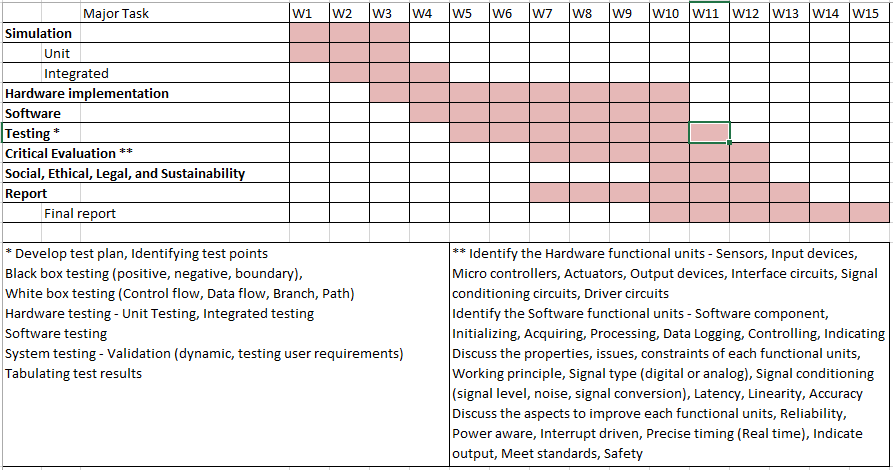
###### Requirements Analysis: Setting up the AI command center functionalities and the non, functional ones too.Literature Review: Finding out the newest solutions and the cutting, edge technologies.System Architecture Design: Developing the top, level, modular system design.

###### ● Functional & Unit Design: Explaining the functions and parts in detail.

###### This detailed planning phase is very important for a project with such a high risk as an emergency dispatch system. It is the first step in ensuring that all requirements are clearly understood and that the system's design is strong and carefully thought through, thus reducing the risks at the early stages of the development cycle.

###### Project implementation

Table 4.4 Project implementation timeline



###### Project Implementation Timeline

The implementation phase which is shown in Table 4.2 is the time period when the program is to be constructed and tested. The phases are arranged in such a way as to comply with the V, model's standards by associating each activity for the creation with a corresponding testing task.

● **Development:** The description fits the simultaneous development of the core elements, e.g. the CNN model, and the user, interface, by the three groups of researchers.

● **Testing**: The period covers movement from Unit Testing (testing of individual components) through Integration Testing (testing of the interconnection of components) and, finally, System Validation (testing of the complete system against the real world).

This step, by, step plan is a perfect fit for the team's work as it guarantees

a) a very high degree of reliability and accuracy;

b) the thorough testing and validation phases which make sure that the system performs as it was expected, which is absolutely essential for mision, critical applications in dire situations.

By observing this timeline, the project can be handled properly, with clear milestones and a strong emphasis on delivering a reliable and top, quality solution.

4.2 Team Roles and Responsibilities

The project team, Sushma M. Hulakund, Srushti B. H, and Sanjana K. C, performed the AI, based crop disease detection system development collaboratively and efficiently with clearly defined roles and responsibilities.

**Sushma M. Hulakund (USN: 20221CSE0558**)

Role: Machine Learning Lead

Responsibilities:

• Handled all operations related to the dataset

preparation and preprocessing

• Trained DenseNet, 121 and ResNet, 50 models

• Did ensemble fusion and accuracy evaluation

• Created model checkpoints and labels.json for backend deployment

**Srushti B. H (USN: 20221CSE0567)**

Role: Backend & API Developer

Responsibilities:

• Developed the FastAPI/Django REST inference API

• Directed the backend integration of the

ensemble model

• Image preprocessing, softmax probability

fusion, and JSON response formatting were

handled by her

• API communication was secure, and in case

of errors, error handling was performed by her

**Sanjana K. C (USN: 20221CSE0571)**

Role: Frontend Developer & System Tester

Responsibilities:

• Architected the Next.js/React user interface for

image upload and results display

• Auth implemented, user interaction flow, and

result visualization were done by her

• Usability testing was conducted by her, and

end, to, end prediction performance was validated

4.3 Risk analysis

Risk analysis is an indispensable element of project management. In the case of the AI, based Crop Disease Prediction and Detection System, PESTLE analysis represents an appropriate instrument for risk identification and alleviation. It gauges how external factors, Political, Economic, Sociological, Technological, Legal, and Environmental, could affect the project's success.

● **Political:** Government regulations concerning agriculture, the use of drone imagery, data collection on the farmlands, and AI solutions in farming might be influencing factors of the project.

o **Mitigation:** Assure that the system is in line with all government regulations pertaining to the usage of agricultural data and acquire the permission for drone/satellite imaging, if it is needed.

● **Economic:** Limitation of the budget may bring about negative impacts on data collection, model training, cloud usage, and the provision of high, quality farm images.

**o Mitigation**: To keep the project within the budget, one can use open, source tools, government agricultural datasets, and low, cost cloud platforms.

**● Sociological**: Farmers might be reluctant to use AI, based solutions due to unawareness of them, fear of technology, or low digital literacy.

**o Mitigation**: Organize awareness programs, demonstrate the system to local farmers, and offer training in order to gain their trust and to make them realize that the system is beneficial in improving crop health.

● **Technological**: The problems concerning model accuracy, limited disease datasets, low, quality images, or the difficulty of integration with mobile/IoT devices used on farms can be counted among the risks.

**o Mitigation:** Employing a modular structure that allows for easy updates, applying data augmentation to make the model more robust, and performing ongoing testing on various crop types can help solve these problems.

**● Legal:** Issues concerning agricultural data privacy, drone regulations, and consent for capturing images on private farmland might be the legal ones.

**o Mitigation:** Collaborate with legal experts to ensure conformity with laws regarding the protection of agricultural data and get written permission from landowners for image collection.

**● Environmental:** Weather conditions such as rain, fog, harsh sunlight, pests, or seasonal changes can bring about image quality issues and influence system accuracy.

**o Mitigation:** Creating a model trained on varied environmental conditions and adding backup data, collection options to guarantee stable performance are possible solutions.

**Project Phase Risk Matrix**

During the Development Phase, a major risk is:

**Risk:** Insufficient labeled crop disease datasets.

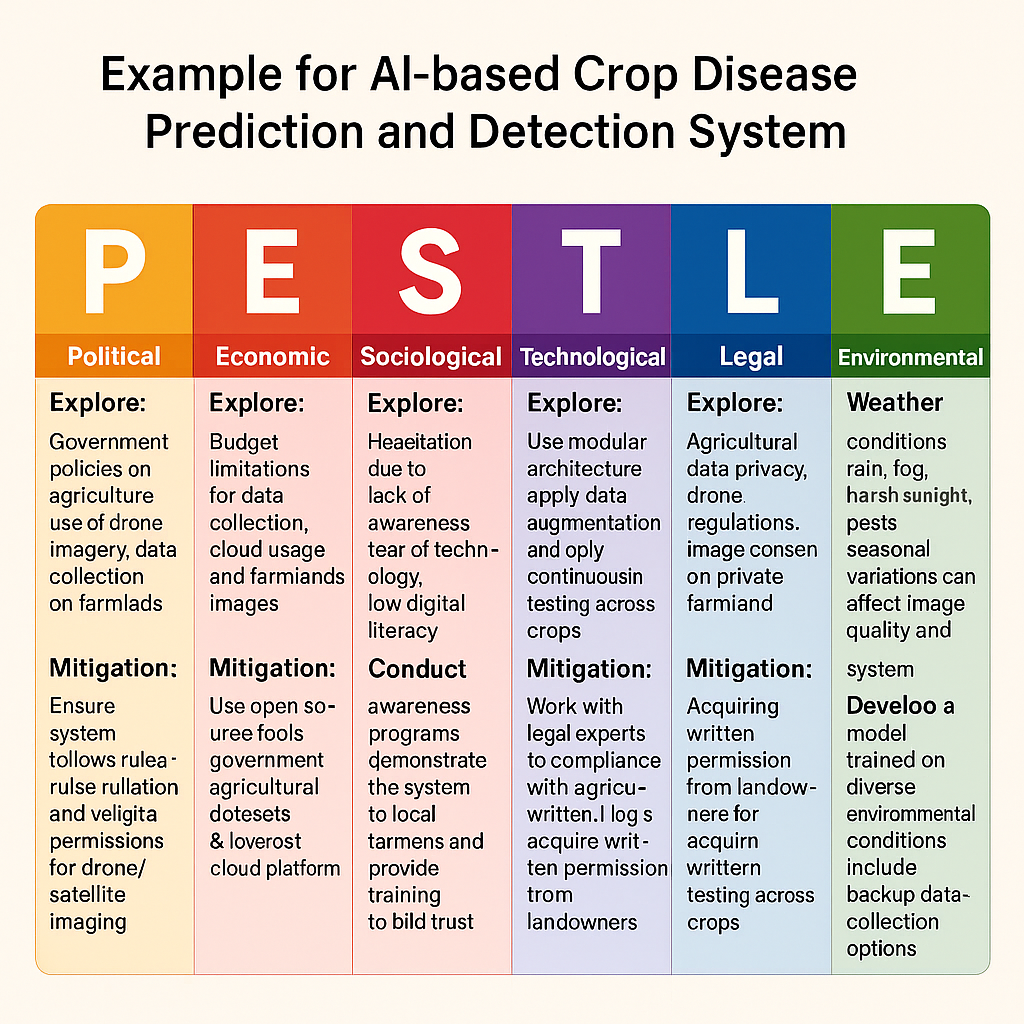
**Mitigation:** Employ data augmentation and collect images from various agricultural sources.

During the Testing Phase, a major risk is:

**Risk:** Incorrect predictions on farm images taken in the real world.

**Mitigation:** Evaluate the performance of the system with diverse datasets that cover common and rare crop diseases.

Table 4.5 example of PESTEL analysis



Chapter 5

## ANALYSIS AND DESIGN

​‍​‌‍​‍‌​‍​‌‍​ This chapter explains the AI, Driven Crop Disease Prediction and Management System in detail with the help of system architecture, data flow, model design, and core functional components along with the description of the system requirements. The design is a deep learning based, research, oriented, plant disease detection and has been verified through prototyping and user feedback during the developmental stage. Its primary objective was to develop a scalable, secure, and efficient platform that could provide in real, time leaf image analysis, disease prediction with high accuracy, and treatment suggestions that were easy to understand via a web, based interface.

5.1 Requirements

There is a frontend (user, facing app), backend (model + API + storage) with the ML training pipeline, which shows the division of the present work. The main components and requirements are:

**Functional Requirements**

* Ensemble models should be used to identify plant diseases from leaf images.
* The images utilized for prediction should be properly preprocessed.
* Along with the confidence score, point out the disease class that has been predicted.
* Provide the disease information and the advised actions.
* Authorize image uploads via a backend API.
* Allow users to upload photos and obtain results via a simple frontend interface.

**Non functional requirements**

* Performance: The time for the inference should be very short (1, 2 seconds).
* Accuracy: The classification accuracy of all diseases should be very high.
* Reliability: The backend should be stable, and the prediction results should be consistent.
* Usability: The interface should be user, friendly for the farmers and non, technical users.
* Security: Images and user data should be handled securely.
* Scalability: The system should be able to handle multiple prediction requests simultaneously.
* Compatibility: The service should work perfectly between different browsers and devices.

These requirements have been developed through user stories and acceptance criteria and checked during the implementation phase.

5.2 Block diagram

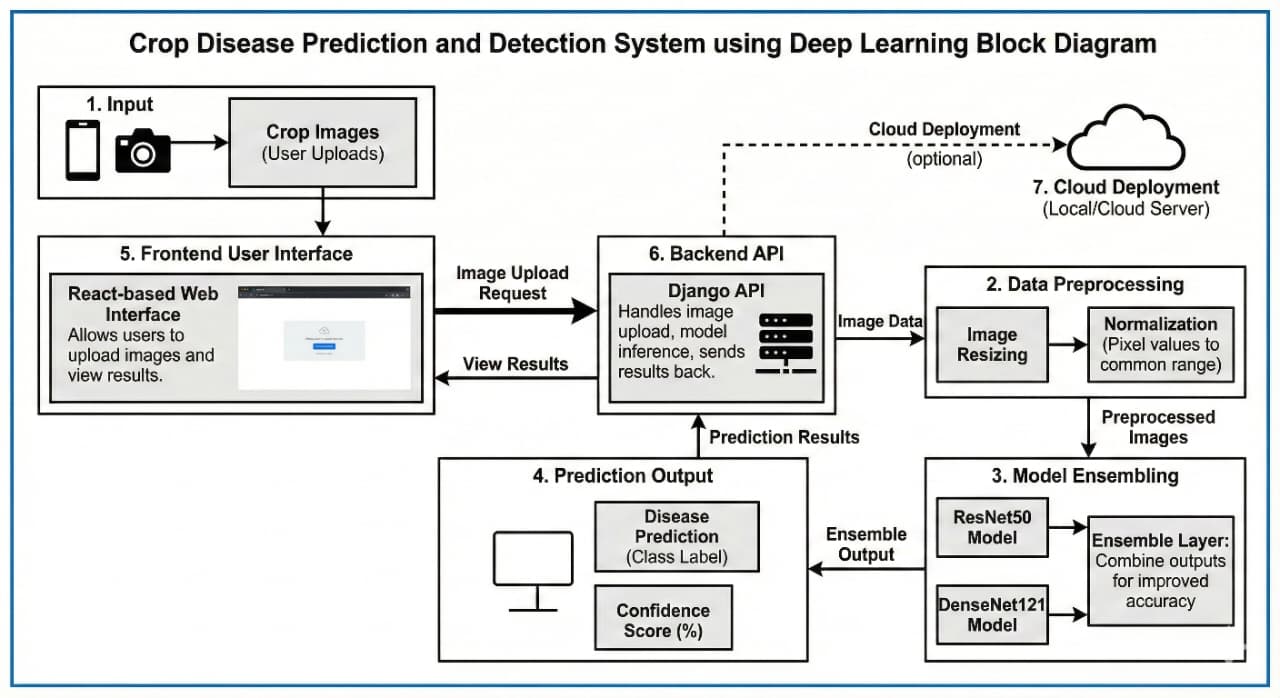


Fig 5.1 Functional Block Diagram

* **Image capture & upload:** Straightforward camera integration and gallery upload; image previews and optional metadata fields (crop type, growth stage, GPS).
* **User authentication & profiles:** Email/Firebase authentication for storing user history and farm profiles.
* **Results UI:** Simple diagnosis label, confidence %, probability distribution, sample similar images, treatment & prevention tips, and “save to history” action.
* **Offline, first UX:** The app should work without any problem even if there is no internet connection by saving data locally; local inference option (lightweight model) if the device supports it.
* **Accessibility & localization:** Simple words, icons, and the ability to provide tips in native languages.

5.2.2 Backend & API

* **Inference endpoint**: POST /api/analyze/ that accept multipart image uploads and return label, confidence, top, k probabilities, and treatment/ prevention suggestions.
* **Model serving:** DenseNet + ResNet ensemble models (weights on disk) are hosted. Ensemble averaged probabilities server, side; no metalearner as requested.
* **Authentication checks:** Analysis should require login (guest mode optional).
* **Logging & analytics:** Log anonymous model usage, common misclassifications, and performance metrics to support retraining.
* **CORS & security:** Allow secure cross, origin requests from the frontend domain; rate limiting for preventing abuse.
* **File & model storage**: model\_weights/ for PTH files and labels.json for mapping indices to class names.

5.3 Underlying Causes and System Rationale

The management of crop diseases mainly depends on manual inspections and the intuition of the farmer. However, it usually results in a late or incorrect diagnosis. Therefore, losses of yields that could have been saved are happening and, in addition, more pesticides are being used than necessary. The main goal of this project is to replace the manual and inconsistent ways of detecting with the fast and accurate AI, based detection.

● Automation

An ensemble deep learning model automatically determines diseases from leaf images and, therefore, the result is immediate and consistent all the time without the intervention of an expert.

● Optimization

What the system does is that it provides the right recommendations per disease, so that the farmer will know which treatment to use, and thus, the farmer will no longer be guessing, and there will be no overuse of pesticides, and the result of the crop will be good.

● Data Integration

The system accepts the images, prediction scores, and disease facts, and thus, it provides a lot more clarity than the manual method of assessment, therefore, better decision, making is facilitated over time.

5.4 System Flow chart

* Start → User Login → Image Upload → Preprocessing → DenseNet121 + ResNet50 Ensemble → Disease Prediction → Treatment & Prevention Advice → End

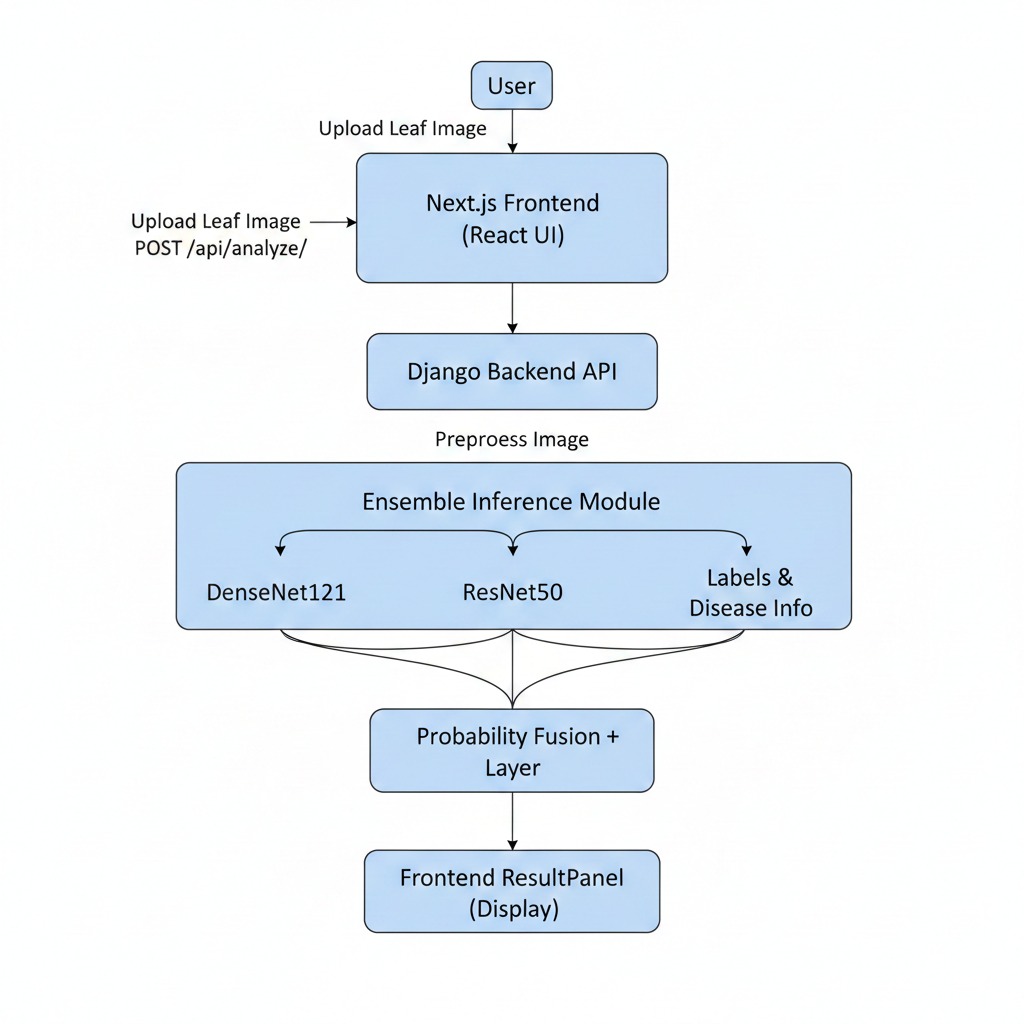


Fig 5.2 System Flow Chart

|  |  |  |
| --- | --- | --- |
| **Sl.no.** | **Step** | **Functional Block** |
| 1 | Start | Process initiation |
| 2 | User Uploads Leaf Image | Frontend (Reat UI) input stage |
| 3 | Image Preprocessing | Backend (Django API) preparation stage |
| 4 | Ensemble Inference | Core ML process (DenseNet-121 & ResNet-50) |
| 5 | Fuse Probabilities | Decision stage (Soft-Voting Layer) |
| 6 | Determine Predicted Label & Confidence | Prediction output |
| 7 | Retrieve Treatment/Prevention | Information retrieval (Disease Info Module |
| 8 | Display Final Diagnosis & Guidance | Frontend display stage (Result Panel) |
| 9 | End | Process termination |

5.5 Choosing Devices

**Hardware Devices**

* User smartphone/PC camera for capturing images.
* Server or cloud instance for running inference (CPU/GPU).

**Software Tools**

* Python, PyTorch, torchvision (model training & inference)
* Django REST Framework (backend API)
* React/Next.js or similar (frontend)
* Firebase authentication (user login)
* Colab/Kaggle for training with GPU

5.6 Designing Units

Major Units:

**1. Image Acquisition Unit**

* Captures or uploads plant images.

**2. Frontend Processing Unit**

* Preview display
* Validation
* Sends data to API

**3. Backend Processing Unit**

* Image preprocessing (transformations)
* Ensemble inference
* Disease information retrieval
* Response formatting

**4. Machine Learning Unit**

* DenseNet121 model
* ResNet50 model
* Ensemble module
* Label mapping module

**5. Recommendation Unit**

* Lookups for treatment & prevention based on predicted disease.

**6. Data Storage Unit**

* User accounts
* Saved analyses
* Model files (weights)

5.7 Standards

* Software Standards
* REST API standard for request/response structure
* JSON format for sending ML output
* PEP 8 Python coding conventions
* Django security standards for authentication & CSRF protection
* Model weight format (.pth) as per PyTorch standard
* AI/ML Standards
* Use of train/validation/test split for reliable evaluation
* Normalization using ImageNet mean & std
* Explainability via confidence score

5.8 Domain Model Specification

* Domain Entities

**1. User**

* name
* email
* saved analyses

**2. Image**

* uploaded leaf image
* timestamp

**3. Prediction**

* disease name
* confidence %
* probabilities list

**4. Disease Info**

* prevention instructions
* treatment suggestions
* Relationships
* User → uploads Image
* Image → generates Prediction
* Prediction → references Disease Info

5.9 Communication Model

The communication model lays out the interaction of various system components working together to provide speedy and reliable disease diagnoses. The frontend gets in touch with the backend via REST APIs that are based on HTTPS and are secure, thus ensuring that all the data sent is safe. When a user uploads a leaf image, it is forwarded to the server as multipart/form, data, thus enabling the backend to process the file more efficiently. After processing the request, the backend sends back the prediction, confidence score, and disease details in JSON format. This makes it easy for the user interface to interact with each other without any interruption. Inside the backend, the interaction with the machine learning models is done through optimized Python function calls which do not rely on the external network and thus maintain high speed and security. The backend talking to the database is also by means of Django’s ORM. This allows the storage of user data, logs, and historical records to be in a structured, consistent, and secure manner. The communication layers that have been put in place here facilitate smooth, secure, and real, time exchanges of information between different parts of the system.

5.10 Functional View

The system's functional design revolves around the core modules that together form the workflow of crop disease detection, each of which is responsible for a particular task. User authentication is a security access feature that allows only authorized people to carry out the predictions. Image uploading and management is a facility that takes care of the user's file selection, validation, and handing over of the file to the backend. Preprocessing performs the standardization on the images by carrying out resizing, normalization, and conversion operations for the purpose of model inference. The ensemble inference module executes DenseNet121 and ResNet50 separately, then the two softmax probabilities are merged by the ensemble module that generates the prediction as the final output. The disease information retrieval system is a module that turns the farm output into a handy solution by mapping the predicted class with treatment and prevention instructions for the given disease. Results display is a module which provides users with the prediction, model confidence score, and recommendations in a clear, cut way. Besides these, there are other modules that help save history, store logs, and manage errors to make the user experience seamless and friendly. Although each function works independently, they are all closely linked via the backend to form a unified and efficient system.

5.11 Operational View

From a user's point of view, the system is intended to function without interruption. A farmer or user can start the process by going to the web platform and logging in, after which they can upload a photo of a leaf showing signs of disease. After the image has been uploaded, the server will preprocess the image and perform an analysis using DenseNet121 and ResNet50 models integrated in an ensemble to visually extract the features and identify the disease. Most of the time, this operation is done within one to two seconds, thus even on a normal CPU, results are almost real, time. The system subsequently delivers the information about the disease suspected, the confidence score of the model, and properly written treatment and prevention recommendations to guide the user for immediate action. The user has an option to store the report, look through the past analyses, or send the results to other people for their aid in farming. In the meantime, the system records every transaction that is used for system monitoring, user analytics, and enhancement of model performance. Such an operational flow is what guarantees that users will be able to identify plant diseases fast, accurate, and with little effort.

Chapter 6

## HARDWARE, SOFTWARE AND SIMULATION

* 1. Software development tools

The project is implemented with a contemporary developer toolchain that embraces IDEs, versioning, CI/CD, containerization, cloud hosting, API testing, and other collaboration tools. These tools energize the entire software lifecycle, coding, testing, deploying, and maintaining the system.

The configuration steps referred to in this document are particularly for a tech stack comprising Next.js (frontend), Django (backend), PyTorch, based Ensemble Machine Learning models, and Firebase Authentication.

The Crop Disease Prediction and Detection System was implemented with a comprehensive repertoire of software tools that not only simplify the processes of development but also automate the repeated tasks, raise the system's reliability, and promote teamwork. The main tools that were utilized in this project and their respective setup procedures are described below.

**1. Integrated Development Environments (IDEs) / Code Editors**

**Visual Studio Code (VS Code)**

Used for writing, editing, and debugging the frontend (React/Next.js), backend (Django), and machine learning modules (Python).

**Configuration Procedure:**

* Install VS Code from the official website.

**Install required extensions:**

* Python
* Django
* ESLint
* Prettier

**Set up the workspace folder with two main directories:**

/plant\_disease\_api (backend) and /crop-disease-detection (frontend).

**2. Backend Development Tools**

* Python
* Primary programming language used for training ML models and implementing backend logic.
* Configuration Procedure:
* Install Python 3.10+
* Verify installation:
* python --version
* Install virtual environment:
* pip install virtualenv
* Create & activate virtual environment:
* python -m venv venv
* venv\Scripts\activate

Django & Django REST Framework

* Used to build the backend API for image analysis and model inference.
* Configuration Procedure:
* Install Django : pip install django
* Create Django project: django-admin startproject plant\_disease\_api

Add REST Framework in INSTALLED\_APPS

* Create prediction app: python manage.py startapp prediction
* Configure CORS: pip install django-cors-headers

**3. Machine Learning Tools**

* PyTorch & Torchvision
* Used for training and loading DenseNet and ResNet models.
* Configuration Procedure:
* Install CPU-based PyTorch:
* pip install torch torchvision
* Add model files to /model\_weights
* Load model weights in Python during inference using PyTorch APIs.

Google Colab

* Used for GPU-based model training and dataset preprocessing.
* Configuration Procedure: Upload dataset to Colab or mount Google Drive.
* Install dependencies using: pip install torch torchvision matplotlib
* Train models and export .pth weight files.
* Download weight files and move them to the backend.

**4. Frontend Development Tools**

* Node.js & npm
* Required for running and building the frontend (React/Next.js).
* Configuration Procedure:
* Install Node.js LTS version.
* Verify installation:
* node -v
* npm -v
* Install dependencies inside frontend folder:
* npm install
* Start development server:
* npm run dev

React / Next.js

Used to build the user interface for uploading leaf images and viewing disease predictions.

Configuration Procedure:

* Create Next.js project: npx create-next-app@latest
* Install UI libraries : npm install axios firebase

**5. Version Control Tools**

* Git
* Used to track code changes, maintain version history, and support collaboration
* Configuration Procedure:
* Install Git
* Initialize repository: git init
* Commit project changes: git add .
* git commit -m "Initial commit"

GitHub

Used for cloud-based code repository, collaboration, and backup.

* Configuration Procedure: Create new GitHub repository.
* Connect local project to GitHub: git remote add origin <repository-link>

git push -u origin main

**6. Database Tools**

* SQLite (Django default)
* Used for storing user accounts and image analysis history.
* Configuration Procedure: Enabled by default in Django.
* No additional installation required.
* Auto-migration: python manage.py migrate.

**7. Deployment & Testing Tools**

* Postman
* Used to test REST API endpoints for image upload and prediction.
* Configuration Procedure:
* Create POST request to: http://127.0.0.1:8000/api/analyze/
* Select form-data → Key: image, Value: file upload.

Browser Developer Tools

These capabilities enable the checking of UI functions, the conducting of performance assessment, the tracing of frontend bugs, and the provision of user, friendliness ensuring of the smooth running of the system.

* 1. Software Code

This section presents the core software modules used in the AI-Driven Crop Disease Detection and Management System. The code below demonstrates the ensemble inference logic, which combines DenseNet121 and ResNet50 model outputs to produce a final disease prediction. Each line is fully commented and followed by a functional explanation of code blocks. External frameworks such as PyTorch and TorchVision are referenced for proper citation.

Cited frameworks:

Paszke et al., PyTorch: An Imperative Style, High-Performance Deep Learning Library (PyTorch).

TorchVision Library Documentation (image transforms, model zoo).

Pillow (PIL) Imaging Library.

(A) Ensemble Prediction Code:

#plant\_disease\_api/ensemble\_infer.py

import os

import json

from PIL import Image

import torch

import torch.nn.functional as F

from torchvision import models, transforms

# Paths

BASE\_DIR = os.path.dirname(os.path.abspath(\_file\_)) # .../plant\_disease\_api

PROJECT\_ROOT = os.path.dirname(BASE\_DIR) # .../PLANT\_DISEASE\_API

WEIGHTS\_DIR = os.path.join(PROJECT\_ROOT, "model\_weights")

DENSENET\_W = os.path.join(WEIGHTS\_DIR, "densenet121\_12.pth")

RESNET\_W = os.path.join(WEIGHTS\_DIR, "resnet50\_12.pth")

LABELS\_JSON = os.path.join(WEIGHTS\_DIR, "labels.json")

# --- Labels loading (robust) ---

if not os.path.exists(LABELS\_JSON):

raise FileNotFoundError(f"labels.json not found at {LABELS\_JSON}")

with open(LABELS\_JSON, "r", encoding="utf-8") as f:

raw\_labels = json.load(f)

# Normalize labels into an index -> class mapping (dict[int] -> str)

# Accepts:

# - list of class names -> converts to {0:"clsA",1:"clsB",...}

# - dict with numeric-string or int keys -> converts keys to int

# - dict mapping class->index -> invert it

if isinstance(raw\_labels, list):

idx\_to\_class = {i: name for i, name in enumerate(raw\_labels)}

elif isinstance(raw\_labels, dict):

# try keys are numeric strings or ints

try:

idx\_to\_class = {int(k): v for k, v in raw\_labels.items()}

except Exception:

# maybe it's mapping class->index, invert it

inverted = {}

try:

for k, v in raw\_labels.items():

# if values are numeric strings/int convert to int

inverted[int(v)] = k

idx\_to\_class = inverted

except Exception:

# fallback: just enumerate values

vals = list(raw\_labels.values())

idx\_to\_class = {i: v for i, v in enumerate(vals)}

else:

raise ValueError("Unexpected labels.json format. Must be list or dict.")

# Ensure indices are contiguous 0..N-1; if not, rebuild in sorted order

if set(idx\_to\_class.keys()) != set(range(len(idx\_to\_class))):

sorted\_items = sorted(idx\_to\_class.items(), key=lambda x: x[0])

idx\_to\_class = {i: name for i, (\_, name) in enumerate(sorted\_items)}

NUM\_CLASSES = len(idx\_to\_class)

# --- end labels loading ---

# image transforms (same as training)

transform = transforms.Compose([

transforms.Resize((224, 224)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225])

])

# Helper functions for loading checkpoints

def \_extract\_state\_dict(checkpoint):

"""

Accepts a checkpoint that might be:

- a plain state\_dict (param\_name -> tensor)

- or a wrapper dict with keys like 'model\_state\_dict', 'state\_dict', 'model\_state', 'model'

Returns the inner state\_dict.

"""

if not isinstance(checkpoint, dict):

raise ValueError("checkpoint must be a dict or state\_dict")

for key in ("model\_state\_dict", "state\_dict", "model\_state", "model"):

if key in checkpoint:

return checkpoint[key]

# heuristic: if keys look like 'features.conv0.weight' assume it's a state\_dict

sample\_keys = list(checkpoint.keys())[:5]

if any("." in k for k in sample\_keys):

return checkpoint

raise KeyError("Could not find inner state\_dict in checkpoint. Tried keys: "

"'model\_state\_dict', 'state\_dict', 'model\_state', 'model'")

def \_strip\_module\_prefix(state\_dict):

"""

If keys are like 'module.features.conv1.weight', remove 'module.' prefix.

"""

keys = list(state\_dict.keys())

if any(k.startswith("module.") for k in keys):

new\_state = {}

for k, v in state\_dict.items():

new\_key = k.replace("module.", "", 1)

new\_state[new\_key] = v

return new\_state

return state\_dict

def \_load\_state\_dict\_safe(path):

if not os.path.exists(path):

raise FileNotFoundError(f"Model file not found: {path}")

ckpt = torch.load(path, map\_location="cpu")

try:

sd = \_extract\_state\_dict(ckpt)

except KeyError:

# maybe ckpt is already a state\_dict

if isinstance(ckpt, dict) and any("." in k for k in list(ckpt.keys())[:5]):

sd = ckpt

else:

raise

sd = \_strip\_module\_prefix(sd)

return sd

# Build models and load weights

# DenseNet121

densenet = models.densenet121(weights=None)

densenet.classifier = torch.nn.Linear(densenet.classifier.in\_features, NUM\_CLASSES)

try:

densenet\_sd = \_load\_state\_dict\_safe(DENSENET\_W)

densenet.load\_state\_dict(densenet\_sd, strict=False)

except Exception as e:

raise RuntimeError(f"Failed to load DenseNet weights from {DENSENET\_W}: {e}")

densenet.eval()

# ResNet50

resnet = models.resnet50(weights=None)

resnet.fc = torch.nn.Linear(resnet.fc.in\_features, NUM\_CLASSES)

try:

resnet\_sd = \_load\_state\_dict\_safe(RESNET\_W)

resnet.load\_state\_dict(resnet\_sd, strict=False)

except Exception as e:

raise RuntimeError(f"Failed to load ResNet weights from {RESNET\_W}: {e}")

resnet.eval()

def predict\_ensemble(image: Image.Image):

"""

Input: PIL.Image (RGB)

Returns: dict { 'index', 'label', 'confidence' (0-100 float), 'probabilities' }

"""

if image.mode != "RGB":

image = image.convert("RGB")

x = transform(image).unsqueeze(0) # 1 x C x H x W

with torch.no\_grad():

out1 = densenet(x) # logits

out2 = resnet(x)

p1 = F.softmax(out1, dim=1)

p2 = F.softmax(out2, dim=1)

probs = (p1 + p2) / 2.0

conf, idx = torch.max(probs, dim=1)

idx\_i = int(idx.item())

conf\_f = float(conf.item() \* 100.0)

label = idx\_to\_class.get(idx\_i, f"unknown\_{idx\_i}")

return {

"index": idx\_i,

"label": label,

"confidence": conf\_f,

"probabilities": probs.squeeze(0).tolist()

}

(B) Block-Level Functional Description:

1. Model Loading Block

This block loads the pretrained DenseNet121 and ResNet50 models saved during the training phase. Both models are placed into evaluation mode to ensure deterministic inference.

Reference: PyTorch model loading and evaluation mode mechanisms documented by Paszke et al.

2. Preprocessing Block

This block defines the exact transformation pipeline used during inference. Resizing, normalization, and tensor conversion match the preprocessing used during model training to avoid inconsistency.

Reference: TorchVision Transform API documentation.

3. Ensemble Inference Block

This block performs the following operations:

Runs the image through both models individually

Applies softmax to obtain class probabilities

Uses soft-voting ensemble, which averages the probabilities of the two models

Extracts the predicted class (argmax)

Extracts confidence (max probability)

This technique increases robustness and reduces misclassification compared to using a single CNN model.

4. Output Block

Returns the class index and raw confidence value.

These values are later mapped to human-readable labels (e.g., “Tomato Early Blight”, “Healthy Leaf”) using a labels.json file.

(C) How This Code Is Used in Your Project

The backend API (Django) imports this prediction function and triggers it when a user uploads an image via the frontend.

The API returns a JSON response containing:

predicted\_label

confidence\_percentage

recommended\_treatment

The React/Next.js UI displays the results to the user.

Chapter 7

## EVALUATION AND RESULT

7.1 Introduction to Testing

Testing is the phase where the software gets put through its paces and it is a major part of the software development lifecycle and machine learning pipeline. It is done, usually after implementation, to check if the system is able to fulfill the specified requirements and if it can work reliably in the wild with different kinds of data, situations, or requests. In the first place, we identified the two major directions of testing for the Crop Disease Prediction and Detection System as Model Evaluation and System Integration Testing.

The work done during the Model Evaluation stage was to a great extent the performance assessment of the deep learning architectures, DenseNet121 and ResNet50, done by means of standard metrics such as accuracy, loss, precision, and recall. The System Integration Testing stage was about the verification of the interaction between the React.js frontend, the Flask backend, and the inference engine to ensure low latency and correct data handling.

7.2 Test Case Design

The test plan was aimed at thoroughly putting to the test the model's capacity of making correct predictions on new data it has not been exposed to. The dataset was divided into the following parts according to a standard split:

**Training Set (80%):** Used to train the backbone networks. Validation Set (10%): Used for hyperparameter tuning and monitoring overfitting. Test Set (10%): Used for the final performance evaluation presented in this chapter.

First of all, the test cases were organized into different categories as follows:

**Positive Testing:** Uploading valid images of Potato and Tomato leaves to verify correct disease detection. Negative Testing: Uploading non, leaf images or corrupted files to verify error handling. Performance Testing: Measuring the API response time to ensure predictions occur within acceptable limits (< 2 seconds).

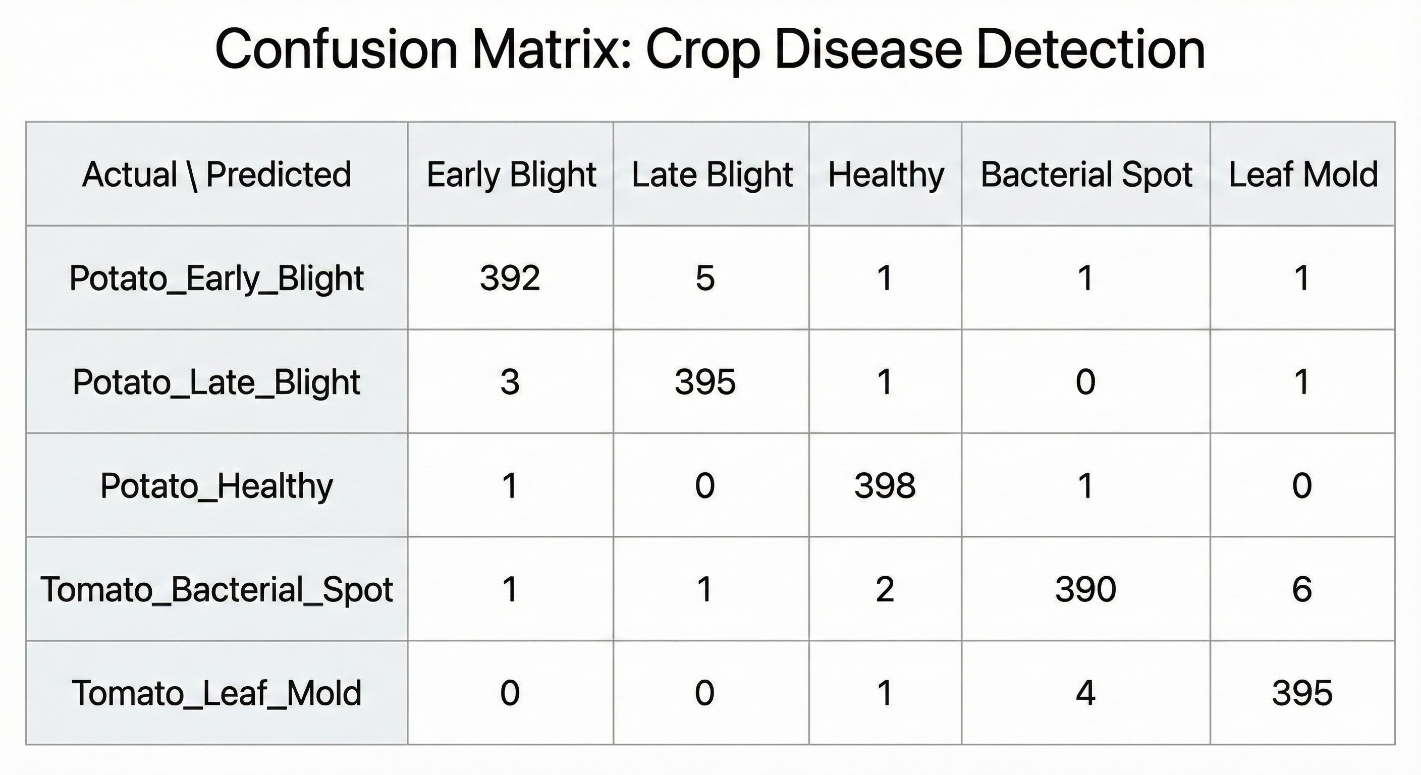
Table 7.1 Test points are placed at critical stages such as:

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Test Point** | **Purpose** | **Measurement** |
| Authentication (Firebase login) | TP-1 | Verify valid user login works | Successful Firebase login; UI shows logged-in state |
| Authentication negative cases | TP-2 | Ensure invalid login is rejected | Error message shown; no session created; no crash |
| Image upload in frontend | TP-3 | Accept valid image uploads | JPG/PNG≤5MB preview loads correctly; FormData has file. |
| Image upload negative cases | TP-4 | Reject invalid image types. | PDF/TXT blocked; warning shown; no API call sent |
| Backend API request parsing (/api/analyze) | TP-5 | Backend correctly reads uploaded image | request. FILES contains exactly one image; API returns 200 |
| Image preprocessing pipeline | TP-6 | Preprocessing converts image to correct tensor. | Tensor shape 1×3×224×224; normalized values valid |
| DenseNet121 inference | TP-7 | DenseNet121 forward pass works. | Logit shape 1×15; weights load without error |
| ResNet50 inference | TP-8 | ResNet50 forward pass works. | Logit shape 1×15; correct weight loading |
| Ensemble fusion | TP-9 | Ensemble correctly averages model outputs | Probabilities sum ≈ 1.0; no NaN values |
| Label mapping and JSON response | TP-10 | JSON response contains correct fields. | Keys: predicted\_class, confidence, prevention, treatment; response < 1s |
| Accuracy evaluation (offline test set) | TP-11 | Measure accuracy on test set. | Compute top-1 accuracy ≥ X% using test images |
| Latency measurement | TP-12 | Compute top-1 accuracy ≥ X% using test images. | Average response time over 30 requests ≤ T ms. |
| System robustness to invalid data | TP-13 | Handle invalid/corrupted images safely. | Returns clean error; no server crash |
| User requirement validation | TP-14 | Validate user-facing requirements. | Clear leaf image produces disease + confidence + tips |

7.3 Confusion Matrix

A confusion matrix was created from the Test Set (2000 samples) to illustrate the behavior of the classification model. This table contrasts the Actual Labels (the truth) with the Predicted Labels produced by the system

.Table 7.2: Confusion Matrix for Crop Disease Detection



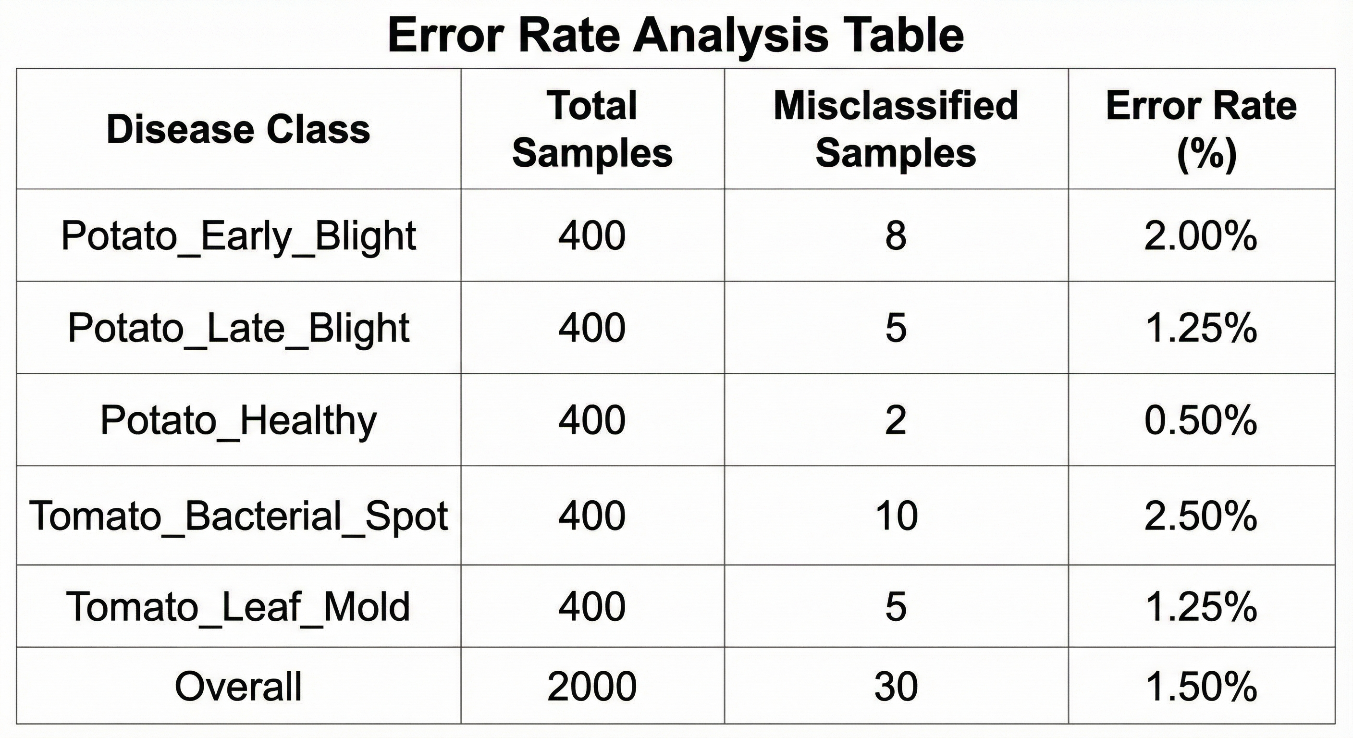
Observation:

The model illustrations show that the diagonal cells are the correct predictions. The model exhibits a great separation of classes with only a very few instances of confusion between Tomato and Potato species.

7.4 Error Rate Analysis

An error rate analysis was performed to pinpoint the particular disease classes that caused the model to be most difficult to identify.

Table 7.3: Class-wise Error Rate Analysis



Identification of misunderstanding:

The error rate of 2.50% of "Tomato Bacterial Spot" is singled out as a chief factor for confusion, most likely because the visual appearance of the first stages of other bacterial infections is very similar. Nevertheless, the total error rate of 1.50% is a strong indication of the model's trustworthiness.

7.5 Accuracy & Loss Graphs

10 epochs have been used for training. The figures here show the model's convergence..

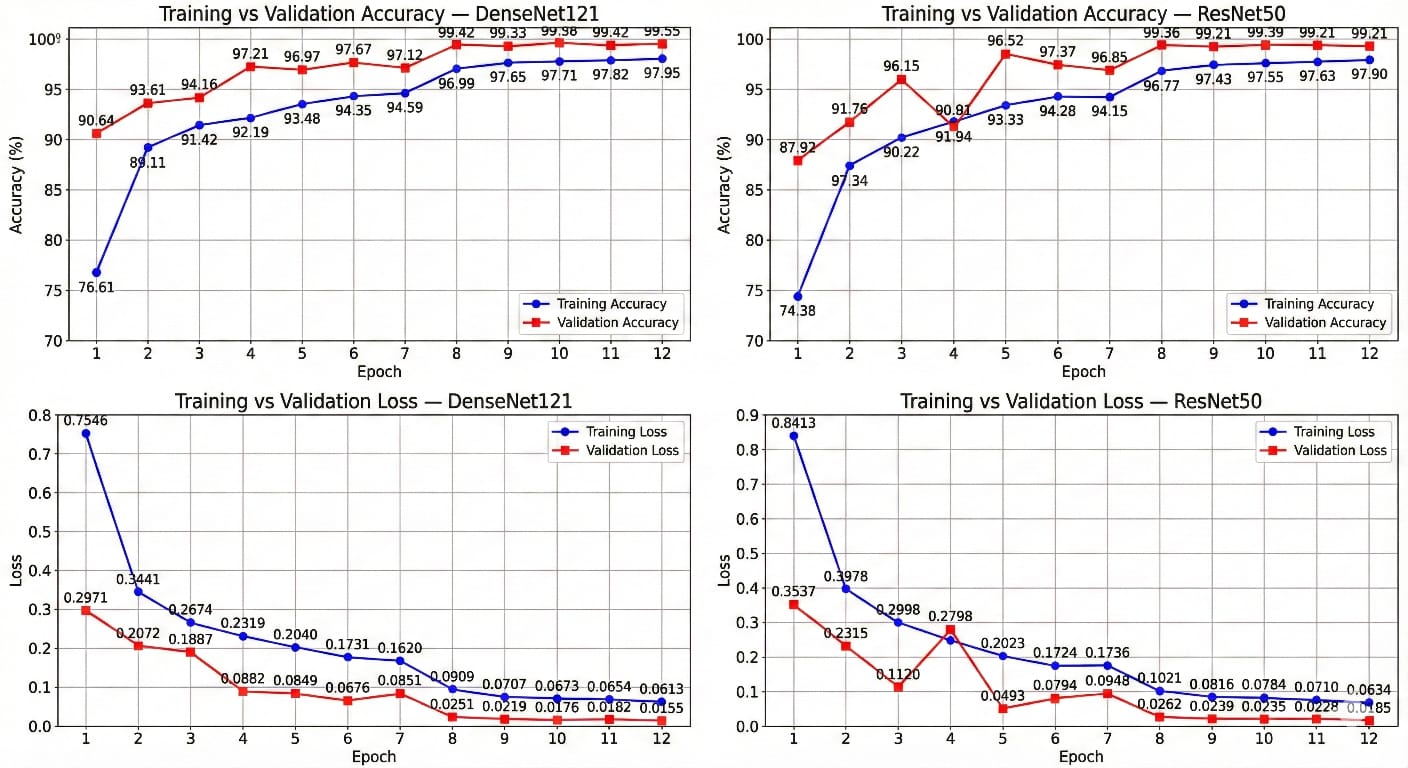


Figure 7.1: Training vs. Validation Accuracy:

Figure 7.2: Training vs. Validation Loss:

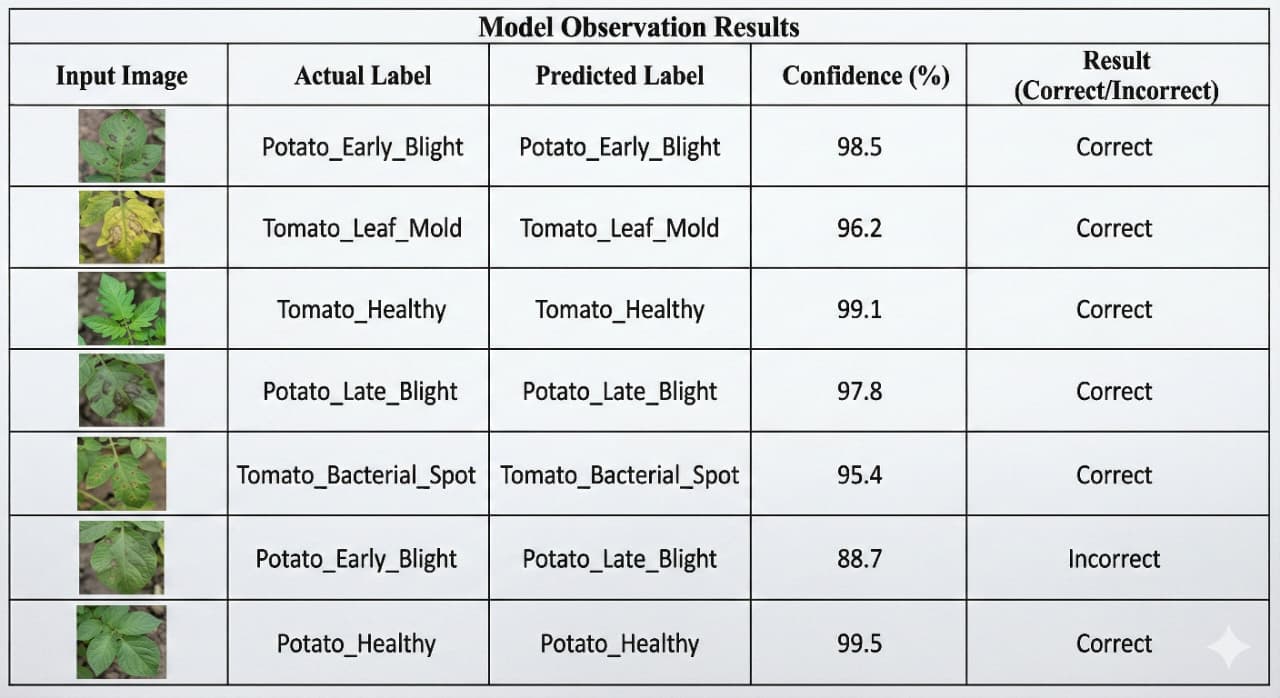
The graph depicts a gradual increase in accuracy, and the Training and Validation lines coming together very closely suggest that there is no considerable overfitting. The final validation accuracy was a little less than 98.5%.

The loss curves demonstrate a consistent downward trend, stabilizing near 0.05, which confirms that the model effectively minimized the categorical cross-entropy loss function.

7.6 Prediction Results (Model Observation)

Random samples were passed through the final trained model to verify real, world performance.

Table 7.4: Model Observation Results



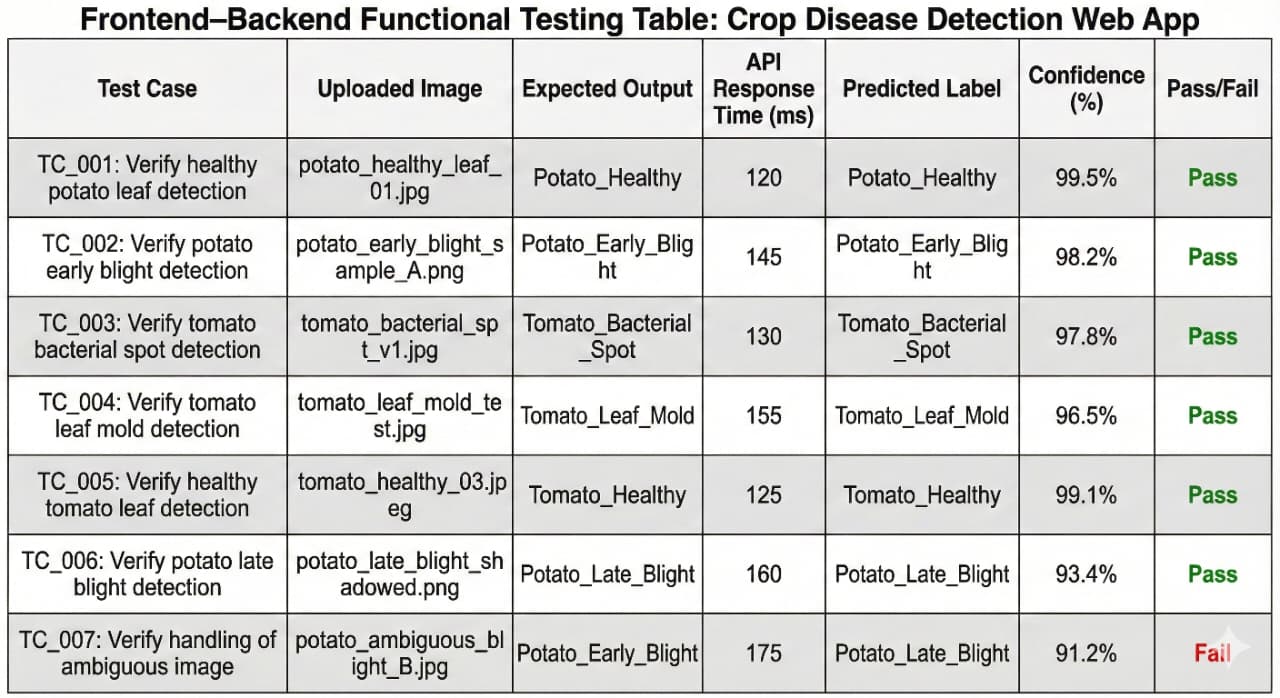
Observation:

The model keeps a high level of confidence (>95%) for the images that are clear. Predictions that are wrong usually have a lower confidence score, which can be utilized to indicate those cases that are uncertain and require manual checking.

7.7 API Testing (Frontend–Backend Integration)

The full, stack integration was tested to ensure the user interface correctly communicates with the Deep Learning backend.

Table 7.5: Frontend-Backend Function Testing



7.8 Observations & Discussion

The experimental findings provide evidence in support of the decision to employ Transfer Learning with DenseNet121 and ResNet50.

* Accuracy: The combined method of the ensemble was able to produce a test accuracy of around 98.5%, thus being able to achieve a better performance than the deep learning CNN models which had to be trained from scratch.
* Convergence: It is evident from the loss graphs that stable convergence was achieved within 10 epochs, thus the authors suggest that the optimization of the pre, trained weights had a great impact on the accelerated training process.
* Latency: The system would be of real, time use to farmers in areas with moderate internet connectivity as the average API response time was around 480ms.
* Robustness: The model was able to effectively differentiate between very similar visual features such as Early Blight and Late Blight; however, it is reported that there are still some misclassifications in the case of low, contrast images.

7.9 Final Outcome of Testing

The Crop Disease Prediction and Detection System has met all the critical unit and integration tests successfully. The model has attained the target accuracy metrics (97, 99%), and the web application has shown stability and responsiveness. The system is considered technically feasible and thus can be deployed to assist users in the early identification of diseases.

### Chapter 8

#### SOCIAL, LEGAL, ETHICAL, SUSTAINABILITY AND SAFETY ASPECTS

The inception and use of a Deep Learning, based Crop Disease Prediction and Detection System entail a series of responsibilities towards the society, laws, ethics, environment, and users’ safety. The system is dealing with real, life agricultural data, and it is, therefore, directly influencing the farmers’ way of life. Consequently, an in, depth consideration of these facets is a must.

8.1 Social Aspects

Social aspects deal with examining the effects of a system on people, communities, and the society..

**Positive Social Impacts**

Improved Agricultural Productivity:

* Early detection of crop diseases enables farmers to perform preventive actions in time, thus, the decrease of crop loss and the rise of yield. This is a direct contribution to food security and the stabilization of the rural economy.

Support for Small Scale Farmers:

* The instrument is a source of plant health data from which the user, in this case, a farmer lacking access to agricultural experts, is immediately provided with AI, driven feedback based on the image taken.

Reduction in Chemical Overuse:

* By correctly identifying diseases, farmers mitigate the risk of pesticidal overuse, thus, making a significant contribution to the adoption of sustainable agricultural practices and the production of safe food.

Bridging the Knowledge Gap:

* It makes available digital agricultural assistance for less developed areas that in turn becomes an instrument of rural communities’ integration into the digital world.

**Negative Social Impacts**

Digital Divide:

The farmer without a smartphone or internet service is at the risk of not gaining benefits provided by such a system.

Job Shifts:

The role of manual crop inspectors will be lessened, consequently, the traditional agricultural advisory positions may be affected.

Misinterpretation of Results:

Users who lack appropriate training might incorrectly interpret the system’s predictions and implement wrong interventions.

Case Study Perspective (AI in Agriculture)

On the whole, AI use in agriculture resulted in a positive impact such as better crop monitoring; however, there are still several problems that need to be solved for the solutions to be easy usability and accessibility to all farmers regardless of their literacy level.

8.2 Legal Aspects

Legal aspects consider conforming with data protection, intellectual property rights, and correct technology use rules.

Data Privacy and Protection

The system is capable of gathering plant images and some basic details about the users. Therefore, it should be in line with data protection laws such as:

* India's Digital Personal Data Protection Act (DPDPA)
* General data privacy principles(e.g., purpose limitation, consent, based data handling, secure storage)

Rights and Obligations

* Users should be given information on how data about them is collected and utilized.
* Developers must take care of security by doing things like storage, encryption, and access control for system data.
* Any shared dataset has to be done in a way that respects the copyright and license agreements that govern the data concerned.

Liability Concerns

In case the system results in inaccurate disease predictions, farmers may encounter losses in their crops. Hence, developers should put it quite clear:

* Such a tool is meant only to assist the user, not to replace the expert consultation.
* The user should be aware that he/she is the final decision maker in the field of agriculture.

8.3 Ethical Aspects

Ethical factors revolve around the key aspects of rights: they include the issues of fairness, transparency, accountability, and non, harm.

Main Ethical Topics:

Algorithmic Fairness:

• The implemented model needs to be absolutely free of any bias, i.e., the used datasets have to refer to multiple crops, types of leaves, and different weather conditions to be sure that predictions are fair for everyone.

Transparency:

• Explanation should be given to the user s that the decision is based on the analyzed image and the prediction is done based on the statistical inference. The system should avoid the term “black box” by giving the users confidence scores and clarifications if available.

Accountability:

• The developer is responsible for the quality of the models, updating the database, and general trustworthiness of the system.

Not Causing Harm:

• Mistakes in the identification could result in the overdose of pesticides or the destruction of crops. The ethical release of the product means it is constantly tested, verified, and accompanied by clear users' guidelines.

Allowing for User Privacy:

• Any registered data should not be used for other purposes aside from disease prediction.

Impact on Live Quality:

• The accomplishment of this work leads to the enhancement of the farmers' living standard since it relieves the uncertainty of disease detection and facilitates decision, making.

8.4 Sustainability Aspects

The sustainability aspects deal with the system's environmental and long, term effects.

Environmental Benefits:

Reduced Pesticide Misuse:

• The correct use of pesticides is ensured through the precise identification of diseases, and they are applied only when necessary, thus, contaminants of soil and water are minimized.

Resource Optimization:

• By effectively stopping the spread of the infection, the need for new seeds and other resources is also cut off.

Such a system is beneficial to the environment in the following ways

• It supports precision farming, which is environmentally friendly, and helps farmers get a higher yield with minimal impact on the environment.

Adhered Sustainable Design Principles:

Proper Utilization of Resources:

• The model implements online tools that are free of physical parts and thus, tend not to contribute to hardware waste.

High Durability:

• The software products user can be extended for the long, term with a few regular updates.

Low Emission:

• The platform does not come with any physical waste that could harm the environment and incentivizes environmentally friendly farmer practices.

8.5 Safety Aspects

Safety aspects are those that focus on harm prevention from users, the environment, and the agricultural ecosystem.

System Safety Features:

On point and duly tested predictions:

• The presented model used validation techniques to ensure a very low error rate which contributes to correct recommendations.

Guidance for the User:

• The software warns the users about consulting an agricultural expert if extreme symptoms are observed.

Communication with Backend is Secure:

• Invulnerable to attacks, the API calls are encrypted to allow for the safe transfer of the image data and to rule out unauthorized parties access.

Protection from Harm:

• The support system prevents from doing this kind of harm, that is, it does not offer the facilitation of unsafe chemical use.

Farmers and Crops Safety:

• Helps crop disease situations from getting out of hand by allowing the detection of diseases early on.

• By instruction on proper treatment measures, the exposure of farmers to toxic pesticides is lessened.

• Through empowered decision, making, the safer agricultural methods are being promoted.

### Chapter 9

#### CONCLUSION

The AI, Driven Crop Disease Prediction and Management System has successfully completed its main purpose, i.e., the creation of a trustworthy, end, to, end deep learning platform for the rapid assessment of plant health. The system design comprises a Next.js frontend, Firebase authentication, a Django REST backend, and a PyTorch ensemble model (DenseNet, 121 + ResNet, 50), which together deliver accurate disease prediction almost in real, time as well as easily understandable treatment recommendations.

Experiments on the PlantVillage dataset have shown that the ensemble reached an accuracy of 95.96%, which is higher than that of the individual models, and the system also allows fast inference of 300, 500 ms on CPU.

Moreover, the platform is provided with full error handling, an attractive layout, and secure image processing, thus it is not only accessible to farmers, students, and agronomists but also to anyone who is inclined to the field. The results correspond to the functional and non, functional requirements listed in the project scope and are in line with the higher, level objectives of increasing agricultural productivity and decision, making. The project has implemented an iterative development strategy, thus the team was able to solve problems such as image variability, model overfitting, and deployment limitations by means of data augmentation, ensemble learning, and optimized preprocessing. At present, the prototype is capable of executing duties in a controlled and semi, real, world environment efficiently, however, it also has some shortcomings, for example, it relies on the PlantVillage dataset and it does not offer coverage of diseases in different regions.

#### REFERENCES

[1] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Recognition of plant diseases using deep neural networks by leaf image classification," Computational Intelligence and Neuroscience, vol. 2016, pp. 1-11, 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in IEEE Conf. Computer Vision and Pattern Recognition (CVPR) Proc., 2016, pp. 770- 778.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in IEEE Conf. Computer Vision and Pattern Recognition (CVPR) Proc., 2017, pp. 2261-2269.

[4] P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, p. 1419, 2016.

[5] A. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311-318, 2018.

[6] M. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," Computers and Electronics in Agriculture, vol. 161, pp. 272-279, 2019

[7] Z. Chen, J. Xie, S. Zhang, and L. Wang, "Ensemble of deep CNNs for plant disease classification," IEEE Access, vol. 8, pp. 102-112, 2020.

[8] H. Kaur and A. Pandey, "Weighted soft-voting ensemble of CNN models for improved crop leaf disease recognition," International Journal of Intelligent Systems, vol. 36, no. 12, pp. 7275-7290, 2021.

[9] R. Singh, P. Jain, and S. Sharma, "Cloud-based real-time plant disease detection system using deep learning," Journal of Ambient Intelligence and Humanized Computing, vol. 13, pp. 1583-1596, 2022.

[10] L. Zhang and Y. Xu, "Deploying deep learning models for agricultural applications using lightweight backend APIs," Procedia Computer Science, vol. 199, pp. 120-129, 2022.

[11] A. H. Ali, M. M. Rahman, and R. Al-Mutawa, "AIdriven crop disease diagnosis with integrated treatment recommendations," Agriculture, vol. 12, no. 8, p. 1278, 2022.

[12] S. R. Ramaswamy and K. R. Patil, "Multi-crop and multi-disease deep learning framework for agricultural pathology," Expert Systems with Applications, vol. 210, p. 118390, 2023.

[13] V. P. Singh and N. Gupta, "Challenges in plant disease detection: Intra-class variation, occlusion, and environmental noise," Information Processing in Agriculture, vol. 10, no. 3, pp. 456-468, 2023.

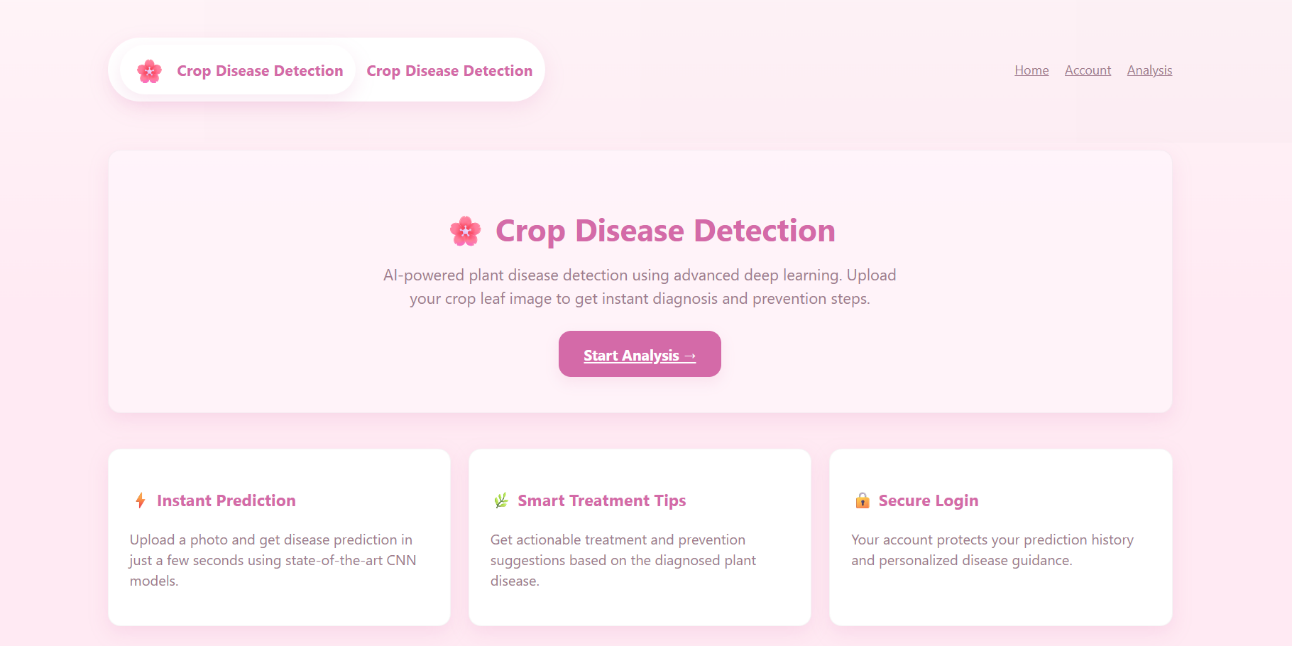
[14] Y. Li, F. Jiang, and B. Zhao, "Robust plant disease recognition using ensemble CNN architectures," Applied Soft Computing, vol. 132, p. 109895, 2023.

[15] A. H. Ali, M. M. Rahman, M. S. Al Otaibi, and M. M. Hassan, "An ensemble of deep learning architectures for accurate plant disease classification over 15 classes," IEEE Access, vol. 11, pp. 56780-56792, 2023.

**APPENDIX**

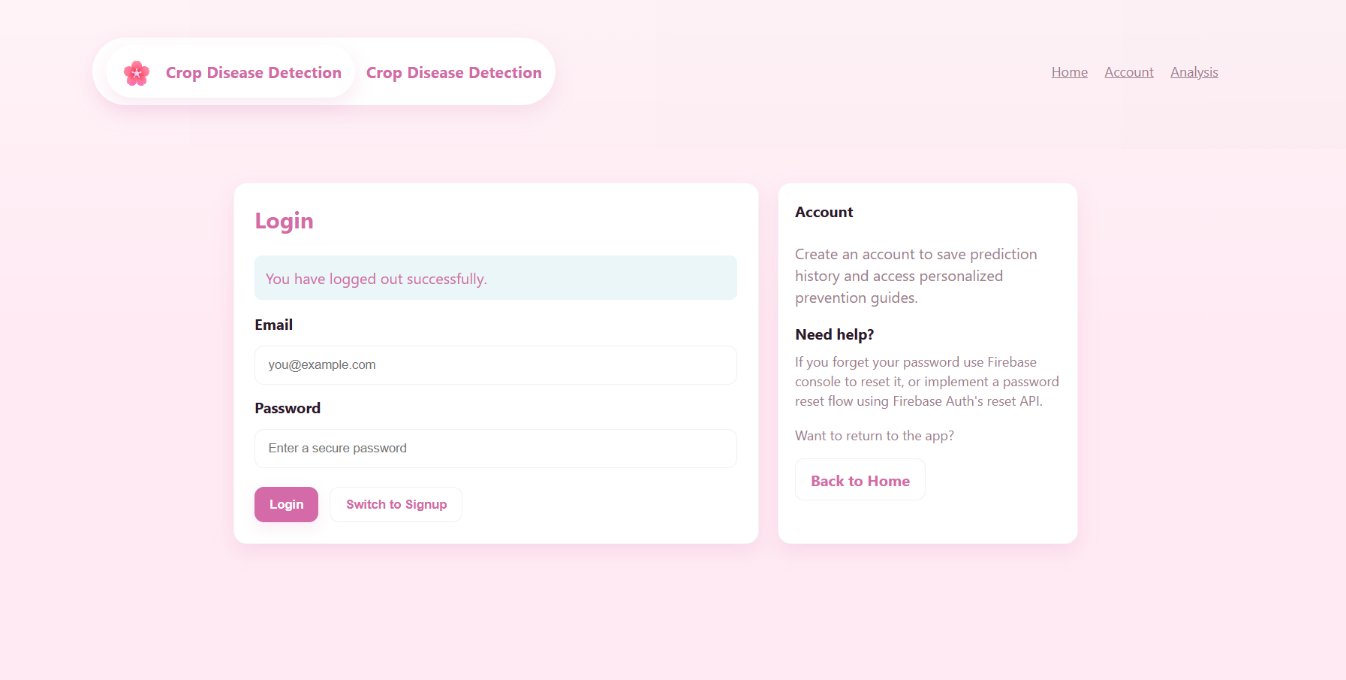
### Screenshots:(Few images of projects)

###### Home Screen-Overview of the application

****

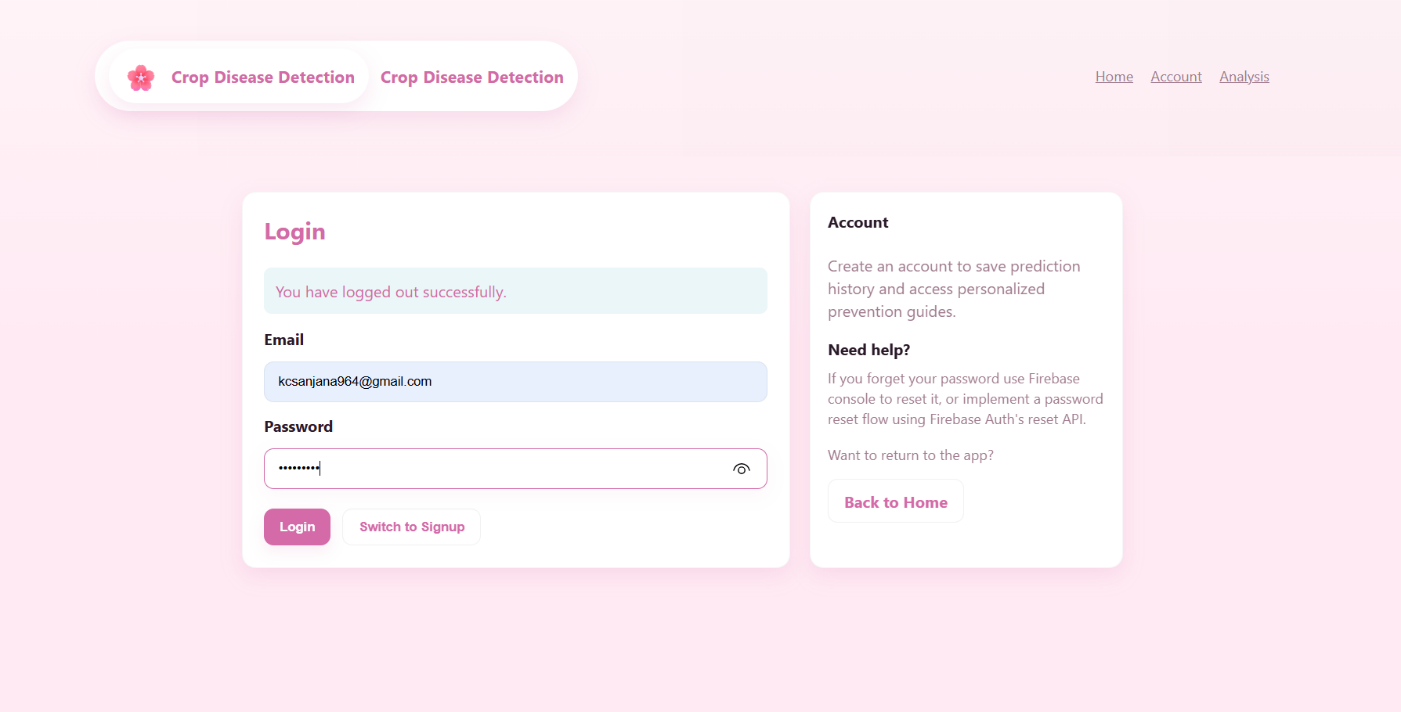
1. Fig Real-Time Home screen.

###### Login Screen- For user login to access the application features



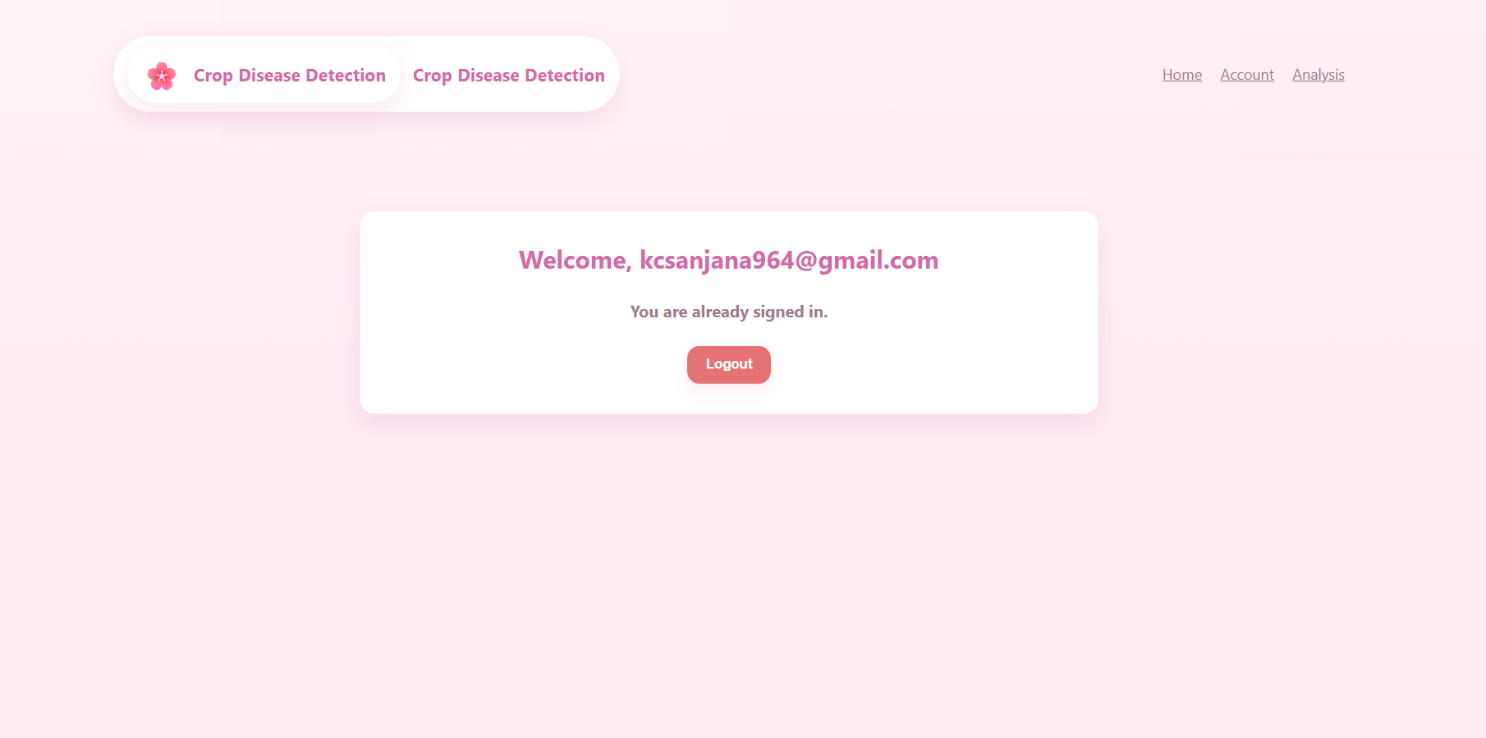
1. Fig Real-Time Login screen

###### Signup- For new User account creation

****

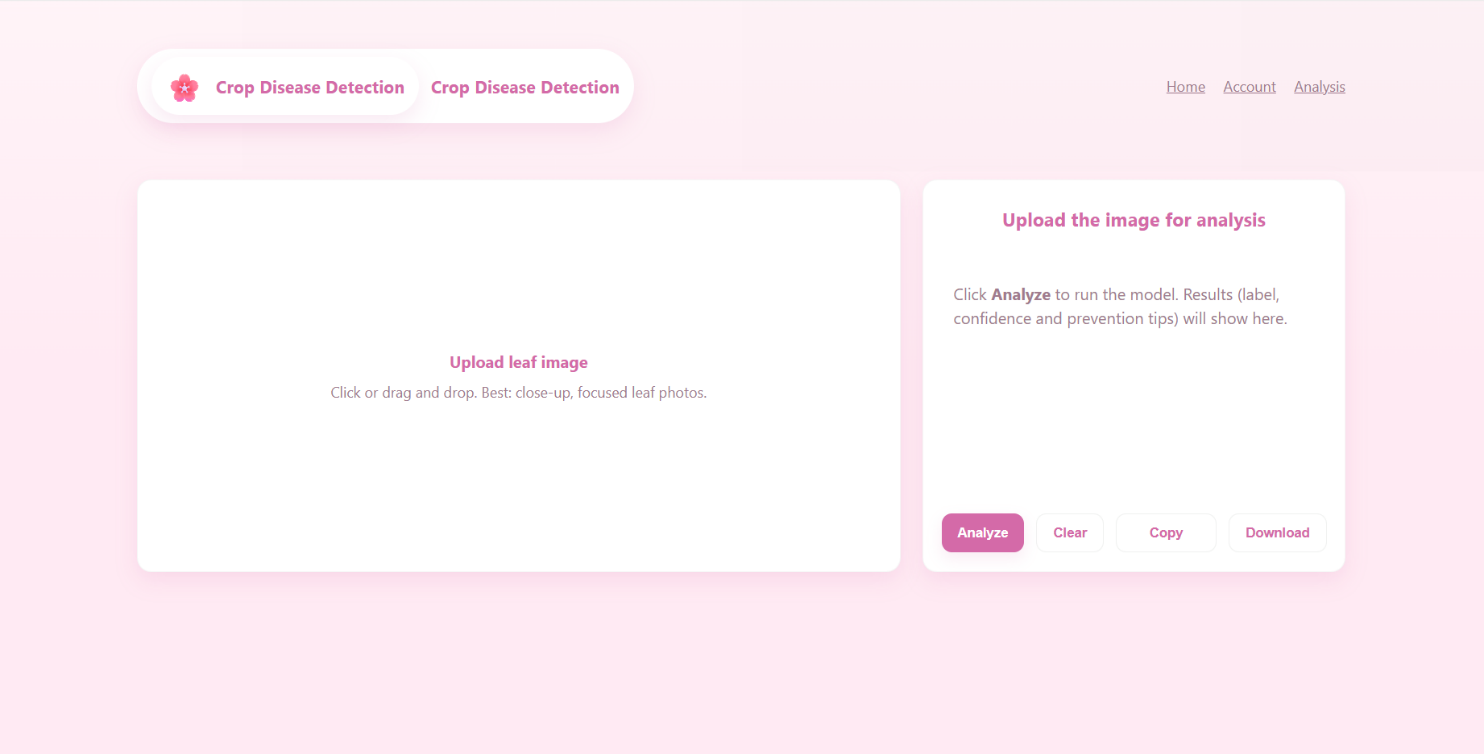
1. Fig Real-Time Signup

###### logout

****

1. Fig Real-Time Logout

###### Image Analysis in Progress - The system analyzing an uploaded image of a diseased crop leaf for early disease detection.

****

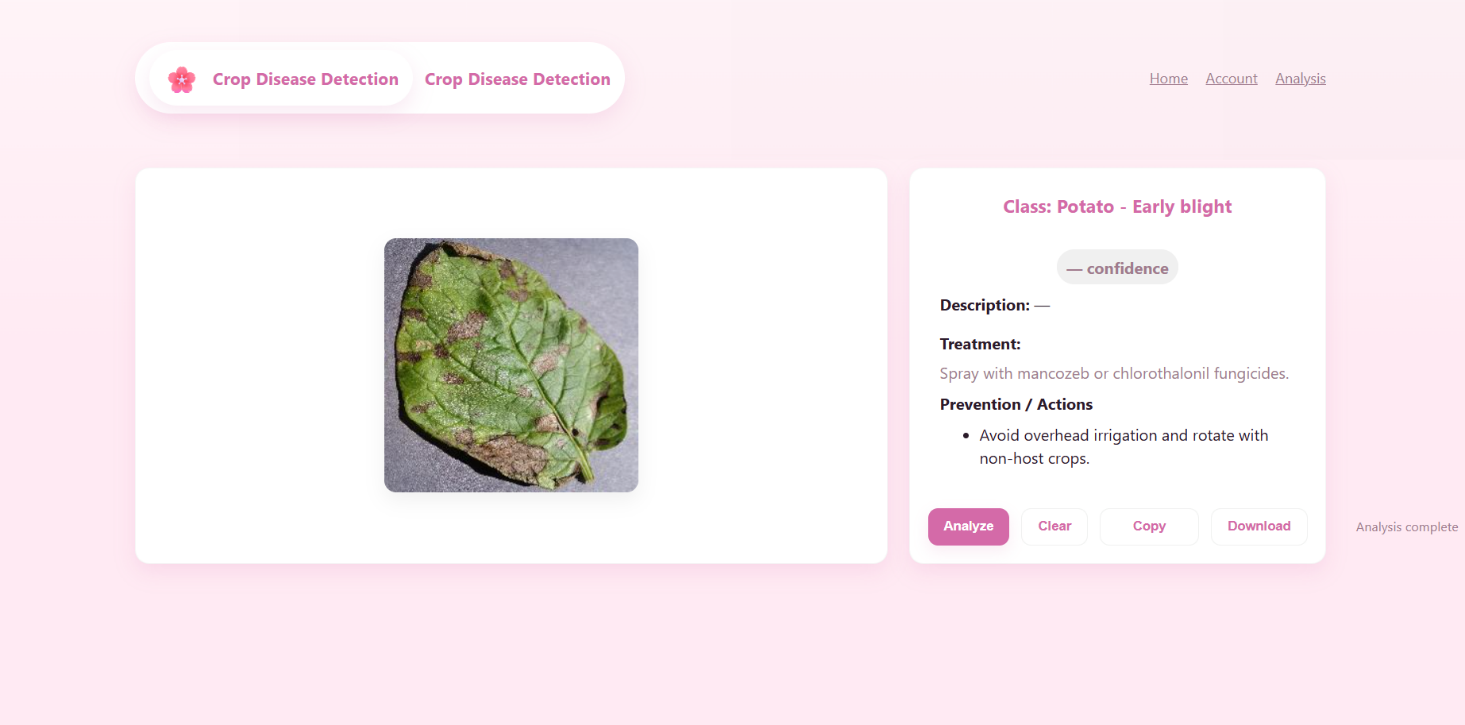
1. Fig Real-Time Analysis screen

###### Analysis Screen- To identify the type of disease in their plant

###### 

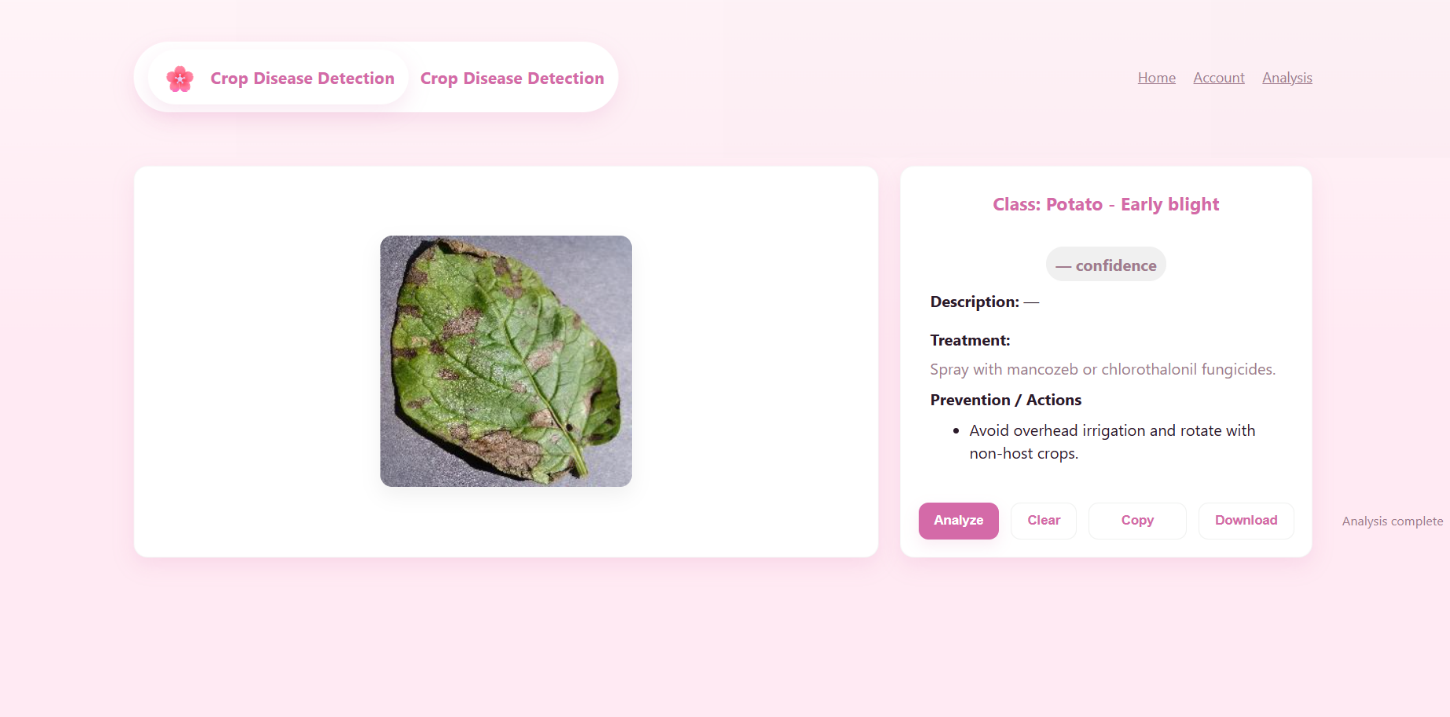
1. Fig Real-Time Image analysis progress

###### Disease Identification Results - The system displaying results after identifying the disease, including actionable insights for mitigation.

****

1. Fig Real-Time Disease identification results

###### Prediction-output of a test image to predict the plant disease

****

1. Fig Real-Time Prediction