Acknowledgement

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely. There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support. We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentors **Prof. Ritika Bhatt**, Associate Professor, AITR, for their motivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Dr Kamal Kumar Sethi**, HOD CSE, AITR Indore for his support, suggestion and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of CSE Dept, AITR Indore for providing me all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Dr S C Sharma**, Director, AITR, Indore whenever needed. We take opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for project to achieve our objectives.

We are grateful to **our parent** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say, “Thank you”, for being the best family that one could ever have and without whom none of this would have been possible.

Ansh Joshi(0827CS201036)
Bhavik Sharma(0827CS201058)
Bhavika Darpe(0827CS201059)
Devendra Singh Pawar(0827CS201067)

Executive Summary

"Crop Disease Detection System"

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal (MP), India for partial fulfillment of Bachelor of Engineering in Computer Science & Engineering branch under the sagacious guidance and vigilant supervision of ***Prof. Ritika Bhatt.***

The project is based on Deep Learning, which is a sub field of machine learning, concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. In this project, Plant Village Dataset is used to train the machine learning model. The result includes the identification of the disease which affects the plants and associated solution for this.

Key words: Transfer Learning, Pytorch, TensorFlow

"Where the vision is one year
cultivate flowers;
Where the vision is ten years,
cultivate trees;
Where the vision is eternity,
cultivate people."

- Oriental Saying

List of Figures

List of Abbreviations

Abbr1: AI-Artificial Intelligence

Abbr2: GPU-Graphics Processing Unit

Abbr3: HTML-Hypertext Markup Language

Abbr4: *ML-Machine Learning*

Abbr5: *OS-Operating System*

Abbr6: *RAM-Random Access Memory*

Abbr7: *ROI-Region of Interest*

Abbr8: *URL-Uniform Resource Locator*

Table of Contents

Chapter 1. Introduction

Introduction

In an era where agriculture plays a pivotal role in ensuring global food security, the development of innovative technologies to safeguard crops is of paramount importance. The Plant Disease Detection System represents a cutting-edge solution that harnesses the power of machine learning and image recognition to empower farmers and gardeners. This revolutionary system enables users to swiftly and accurately diagnose diseases in their crops by simply uploading images of affected plants. Through advanced algorithms and data-driven insights, the Plant Disease Detection System not only identifies the ailments but also provides tailored solutions, thereby offering a critical tool to enhance agricultural productivity, minimize crop losses, and promote sustainable farming practices. With the fusion of technology and agriculture, this system stands as a beacon of hope in the quest to meet the ever-growing demands of our global population while safeguarding our environment and resources.

1.1 Overview

The project is based on image classification where the dataset is classified into one of the 39 classes. The model is trained on a pre-trained model which is ResNet50 model. Also, at the end of the model, one fully connected layer is added. The model is trained and tested for a large dataset and finally integrated with frontend to make a fuly functional plant disease detection website.

1.2 Background and Motivation

As the holy grail of computer vision research is to tell a story from a single image or a sequence of images, object detection and recognition has been studied for more than four decades. Significant efforts have been paid to develop representation schemes and algorithms aiming at recognizing generic objects in images taken under different imaging conditions (e.g., viewpoint, illumination, and occlusion).

Image classification becomes a necessity when there is a need of automation, where the identification is done by machines instead of doing it manually for better performance and reliability. In an era where agriculture plays a pivotal role in ensuring global food security, the development of innovative technologies to safeguard crops is of paramount importance. The Crop Disease Detection System represents a cutting-edge solution that

harnesses the power of machine learning and image recognition to empower farmers and gardeners. This revolutionary system enables users to swiftly and accurately diagnose diseases in their crops by simply uploading images of affected plants.

1.3 Problem Statement and Objectives

Effective and early detection of plant diseases is vital to ensure crop health, reduce crop losses, and promote sustainable agriculture. However, traditional methods are often time-consuming and rely on human expertise. There is a pressing need for a reliable and efficient automated system that can accurately identify and diagnose plant diseases through image recognition and machine learning, offering practical solutions for farmers and gardeners to manage and protect their crops.

Objective: The goal of this project is to predict the disease which affects a particular plant/crop by uploading the image of the affected part.

1.4 Scope of the Project

The scope of plant disease detection encompasses the development of advanced technology solutions that leverage image recognition, machine learning, and data analytics to identify, diagnose, and manage plant diseases. These systems can benefit agriculture, horticulture, and environmental conservation by enabling early disease detection, sustainable farming practices, and increased crop productivity while reducing the reliance on chemical pesticides.

1.5 Team Organization

- **Ansh Joshi:**

  I investigated and found the right technology and studied in deep about it. For the implementation of the project, I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the backend of the project.

- **Bhavik Sharma**:

  I investigated and found the right technology and studied in deep about it. For the implementation of the project, I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the backend of the project.

- **Bhavika Darpe**

  I investigated and found the right technology and studied in deep about it. For the implementation of the project, I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the backend of the project.

- **Devendra Singh Pawar:**

  I investigated and found the right technology and studied in deep about it. For the implementation of the project, I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the backend of the project.

1.6 Report Structure

The project ***Crop Disease Detection System*** is primarily concerned with the **Transfer Learning** and whole project report is categorized into five chapters.

Chapter 1: Introduction- introduces the background of the problem followed by rationale for the project undertaken. The chapter describes the objectives, scope and applications of the project. Further, the chapter gives the details of team members and their contribution in development of project which is then subsequently ended with report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of existing system and highlights the issues and challenges of project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System - starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrate software engineering paradigm used along with different design representation. The chapter also includes and details of major modules of the project. Chapter also gives insights of different type of feasibility study carried out for the project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation - includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interface designed in project along with their functionality. Further it discusses the experiment results along with testing of the project. The chapter ends with evaluation of project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion - Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2. Review of Literature

Review of Literature

Over the years, technological advancements have revolutionized this area. Machine learning and image recognition have become instrumental in the development of automated disease detection systems. Research has demonstrated that convolutional neural networks (CNNs) are particularly effective in accurately identifying diseases in plants based on leaf images. These systems, often integrated into smartphone apps like Plantix, enable farmers and gardeners to simply upload images for real-time disease diagnosis. Such tools not only expedite the identification process but also provide tailored solutions, helping users effectively manage plant health.

Moreover, the global agricultural community has recognized the potential of remote sensing technologies, drones, and IoT-based sensor networks in monitoring plant health on a larger scale. These technologies have the capacity to enhance precision agriculture and sustainability efforts.

The literature review also highlights the importance of open-access disease databases and collaborative platforms, fostering research and development in this field. Overall, it is evident that the integration of technology in plant disease detection is pivotal in addressing food security challenges and promoting environmentally friendly farming practices.

2.1 Preliminary Investigation

2.1.1 Current System

There are certain systems available for plant disease detection as follows:

1. Remote Sensing and Drones: Remote sensing technologies and drones equipped with multispectral cameras can capture images of large agricultural areas to detect early signs of stress and diseases.

2. Plantix: A popular smartphone app designed to assist farmers, gardeners, and agricultural enthusiasts in identifying plant diseases, pests, and nutrient deficiencies.

3. AgriApp: It provides complete information on crop production, crop protection, smart farming with agriculture and allied services.

2.2 Requirement Identification and Analysis for Project

Requirement identification and analysis for a plant disease detection system involves understanding the needs and expectations of the users and stakeholders. Here's an analysis of the key requirements for such a system:

- Image Recognition and Disease Identification: The system must accurately identify and classify plant diseases, pests, and nutrient deficiencies using images submitted by users. It should support a wide range of crops and diseases to cater to diverse agricultural needs.
- User-Friendly Interface: The user interface should be intuitive, accessible, and user-friendly, making it easy for farmers, gardeners, and agricultural professionals to upload images and receive results.
- Rapid Diagnosis: The system should provide quick and real-time disease diagnosis to enable timely intervention and minimize crop losses.
- Customized Solutions: Along with identifying diseases, the system should offer tailored solutions and recommendations for disease management, including suggested treatments, pesticides, and preventive measures.
- Offline Functionality: To accommodate users in areas with limited internet connectivity, the system should have an offline mode for image capture and later upload.
- Extensive Plant Database: Maintain a comprehensive plant database with detailed information on various plant species, diseases, pests, and nutrient deficiencies.

- Machine Learning Model: The system must utilize a robust and constantly evolving machine learning model, such as CNNs, to improve accuracy and accommodate new diseases and plant varieties.
- User Community and Collaboration: Provide a platform for users to engage with a community, share knowledge, and seek advice on plant health issues.
- Data Security and Privacy: Ensure robust data security and privacy measures to protect user data and images.
- Scalability: Design the system to scale with increasing user demand, considering potential growth.
- Continuous Improvement: Implement mechanisms for continuous feedback collection, model improvement, and database updates based on user input and emerging plant health issues.
- Compatibility and Integration: Ensure the system is compatible with various devices and operating systems. Consider integration with other agricultural tools and platforms.
- Accuracy and Reliability: The system should prioritize high accuracy and reliability in disease identification and recommendations.

By analyzing these requirements, you can create a comprehensive and effective plant disease detection system that addresses the needs of users and contributes to sustainable agriculture and food security.

2.3 Conclusion

The development and application of a Crop Disease Detection System is an advancement, in agriculture and horticulture. This innovative system, which utilizes the power of machine learning and image recognition provides a solution for detecting and managing diseases at a stage. It has the potential to revolutionize farming by not identifying diseases and pests but also offering practical strategies, for disease control and prevention. By addressing the challenge of disease detection and providing recommendations this system can greatly boost crop productivity minimize financial losses and contribute to ensuring food security.

Chapter 3. Proposed System

Proposed System

3.1 The Proposal

Our project aims to develop a robust and user-friendly Plant Disease Detection System using machine learning and image recognition technology. This innovative system will empower farmers, gardeners, and agricultural professionals to swiftly and accurately diagnose diseases and health issues in their crops. Users will simply upload images of affected plants, and the system will provide real-time disease identification and customized solutions for disease management. The primary goal is to enhance agricultural productivity, reduce crop losses, and promote sustainable farming practices. The system will feature an extensive plant database, a user-friendly interface, offline functionality, and a collaborative user community. Continuous improvement through machine learning and user feedback will ensure the system's accuracy and relevance. By meeting these requirements, our project aspires to bridge the gap between traditional farming practices and cutting-edge technology, contributing to global food security and environmental conservation.

3.2 Benefits of the Proposed System

Of course, here are the benefits of the proposed system in English, which can instill trust and support among users:

- Time Savings: This system saves time by quickly identifying diseases. Farmers and agricultural professionals can take prompt action without delay.
- Relevant Advice: The system understands disease outbreaks and offers tailored advice, such as the right disease response, fertilizer recommendations, and protective measures.
- Increased Agricultural Yields: Timely disease identification and practical advice lead to crop improvement, contributing to increased agricultural productivity.
- Environmental Conservation: By reducing chemical usage through proper recommendations, the system supports environmental conservation.

- Access to Information: The system provides disease information and knowledge, equipping users with insights into agriculture and diseases.

- Business Opportunities: The development and commercial use of this system offer new, prosperous business opportunities to the agricultural industry.
- Community Benefit: The system fosters a sense of community where knowledge is shared, and collective problem-solving is encouraged.
- Localized Agriculture: With its localized advice, the system aids in the promotion of precision agriculture and region-specific farming practices.

These benefits underscore how the proposed system can be a valuable and trusted tool for users, contributing to food security, sustainable agriculture, and environmental protection.

3.3 Feasibility Study

Feasibility Study for a Plant Disease Detection System

1. Technical Feasibility: Assess the availability and capabilities of technology needed for image recognition and machine learning. Ensure the technical expertise and resources are in place for system development.
2. Economic Feasibility: Conduct a cost-benefit analysis to compare development costs with expected benefits. Explore potential revenue sources and assess the return on investment (ROI).
3. Market Feasibility: Analyze market demand and assess user acceptance. Identify competitors and evaluate their market presence.
4. Operational Feasibility: Ensure the availability of human and physical resources. Plan for system integration and user training.
5. Legal and Regulatory Feasibility: Examine legal and regulatory requirements, particularly data privacy and agricultural regulations. Ensure compliance with relevant laws.
6. Environmental and Ethical Considerations: Evaluate the environmental impact and ethical concerns. Promote sustainability and ethical AI usage.
7. Risk Assessment: Identify potential risks and develop mitigation strategies.
8. Recommendations: Summarize findings and provide recommendations on project viability.

This condensed feasibility study provides a quick overview of the key factors to consider when assessing the viability of the Plant Disease Detection System.

3.3.1 Technical

Technology Evaluation: Examine the technical capabilities and requirements for implementing image recognition, machine learning, and database management. Ensure that the necessary technology and tools are available or can be acquired.

Development Resources: Assess the availability of skilled personnel, hardware, and software required for system development. Evaluate whether the technical expertise is in place to build and maintain the system.

Machine Learning Models: Investigate the feasibility of developing and maintaining machine learning models for disease detection. Analyze the availability of relevant data for model training and optimization.

3.3.2 Economical

Cost-Benefit Analysis: Conduct a thorough cost-benefit analysis to evaluate the economic feasibility. Consider the initial and ongoing costs of system development, including software development, server hosting, and maintenance.

Revenue Generation: Explore potential revenue sources, such as premium subscriptions, ads, or partnerships with agricultural organizations. Estimate revenue projections and the timeframe for achieving profitability.

Return on Investment (ROI): Calculate the expected ROI over time, taking into account revenue estimates and cost projections. Determine whether the long-term financial returns justify the initial investment.

3.3.3 Operational

Resource Availability: Ensure that human and physical resources necessary for system operation are readily available. Consider factors like developer expertise, data annotation, and server hosting.

System Integration: Assess the feasibility of integrating the system with existing agricultural practices and tools used by farmers and agricultural organizations. Ensure that it can seamlessly fit into existing workflows.

User Training and Support: Plan for user training and support to ensure that users can effectively navigate and utilize the system. Evaluate the operational effort required to maintain user satisfaction.

3.4 Design Representation

3.4.1 Use Case Diagram

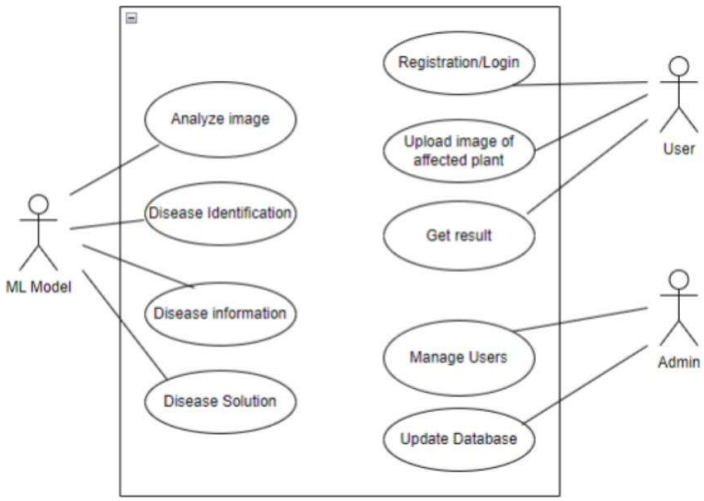

Figure 3-1: Use Case diagram

3.4.2 Sequence Diagram

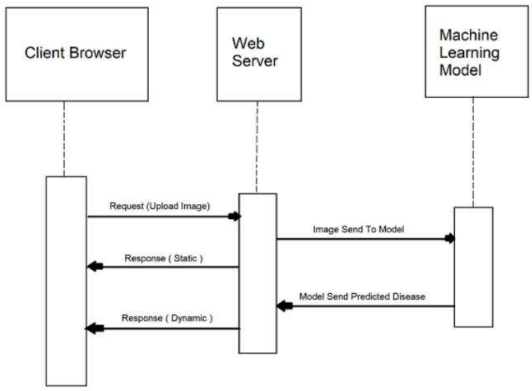

Figure 3-2: Sequence diagram

3.4.3 Data Flow Diagrams

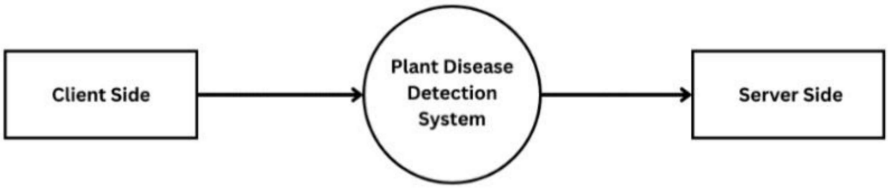

Figure 3-3 Data Flow Diagram Level 0

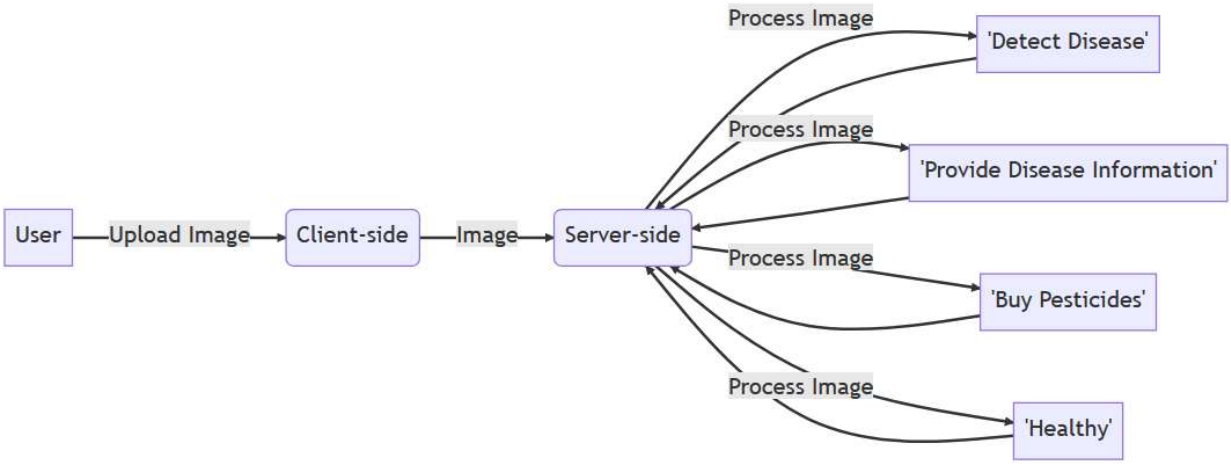

Figure 3-4 Data Flow Diagram Level

3.4.5 Dataset Structure

Dataset is the training and testing data on which our machine learns and test its accuracy. The dataset is the Plant Village Dataset which consists of approx. 61000 plant images which are classified into 39 classes. The dataset is reduced to a certain size to make the computation and processing of the ML model fast and efficient.

Following are the 39 classes present in the Plant Village Dataset

1. Apple___Apple_scab
2. Apple___Black_rot
3. Apple___Cedar_apple_rust
4. Apple___healthy
5. Background Without Leaves
6. Blueberry___healthy
7. Cherry___Powdery_mildew
8. Cherry___healthy
9. Corn___Cercospora_leaf_spot Gray_leaf_spot
10. Corn___Common_rust
11. Corn___Northern_Leaf_Blight
12. Corn___healthy
13. Grape___Black_rot
14. Grape___Esca_(Black_Measles)
15. Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
16. Grape___healthy
17. Orange___Haunglongbing_(Citrus_greening)
18. Peach___Bacterial_spot
19. Peach___healthy
20. Pepper,_bell___Bacterial_spot
21. Pepper,_bell___healthy
22. Potato___Early_blight
23. Potato___Late_blight
24. Potato___healthy
25. Raspberry___healthy
26. Soybean___healthy
27. Squash___Powdery_mildew
28. Strawberry___Leaf_scorch
29. Strawberry___healthy

30. Tomato___Bacterial_spot
31. Tomato___Early_blight
32. Tomato___Late_blight
33. Tomato___Leaf_Mold
34. Tomato___Septoria_leaf_spot
35. Tomato___Spider_mites Two-spotted_spider_mite
36. Tomato___Target_Spot
37. Tomato___Tomato_Yellow_Leaf_Curl_Virus
38. Tomato___Tomato_mosaic_virus
39. Tomato___healthy

Figure 3-7: DataSet Structure

3.5 Deployment Requirements

The deployment hardware and software requirements for the proposed plant disease detection system include the following:

3.5.1 Hardware

- Processor: Intel Core i5 or higher
- RAM: 8 GB or higher
- Hard Disk Space: 1 TB or higher
- Graphics Card: NVIDIA or AMD with a minimum of 2 GB memory
- Internet Connection: Broadband or higher

3.5.2 Software

- Operating System: Windows 10 or Ubuntu 18.04 or higher
- Python 3.7 or higher
- Scikit-learn Library
- Jupyter Notebook
- VS Code
- The software components must be installed and configured properly for the system to function correctly. The Python environment must have all the necessary packages installed like pytorch, flask and torchvision.

The proposed system also requires a web server, such as Apache or Nginx, to host the web application. Additionally, a domain name and SSL certificate are required to ensure secure communication between the client and server.

Overall, the hardware and software requirements for the proposed plant/crop disease detection system require careful consideration to ensure that the system functions correctly and provides users with a satisfactory experience.

Chapter 4. Implementation

Implementation

The implementation process would involve the following steps:

- Data collection: This involves collecting a large dataset healthy and diseased plants. The dataset is used such that it contains labelled classes.
- Data Transformation: This involves transforming the data into desired set of size and features.
- Model training: A machine learning model i.e. vgg16 is used which consists of 16 convolution layers as well as pooling and batch normalization layers.
- Model evaluation: The trained model will be evaluated on a test set of images to determine its accuracy and effectiveness in detecting a particular disease.
- Integration: The model will be integrated with frontend of the web application to give the desired output.
- User interface: The user interface will be provided to users to upload images of the affected plant and this interface will be created using streamlit.
- Overall, implementing this model requires a thorough understanding of machine learning, web development, and cloud computing, as well as expertise in the specific technologies used.

4.1 Technique Used

4.1.1 Image Processing:

In the context of the plant disease detection project, image processing plays a pivotal role. It involves a series of techniques applied to raw plant images to enhance their quality and prepare them for analysis. Techniques like resizing, normalization, and data augmentation are employed to ensure that images are in a consistent format and that the model can effectively learn from them. Image processing helps improve the accuracy and robustness of the disease detection system by standardizing and diversifying the training data.

4.1.2 Transfer Learning:

Transfer learning is a critical component of the project's success. It allows the project to leverage pre-trained Convolutional Neural Networks (CNNs) that have already learned essential features from extensive datasets. By fine-tuning these pre-trained models on plant disease images, the project can expedite model training and achieve higher accuracy. Transfer learning helps bridge the gap between generic image recognition and plant disease detection, making the system more efficient and effective in identifying diseases across various plant species.

4.1.3 Convolutional Neural Networks:

Convolutional Neural Networks, or CNNs, serve as the project's backbone for image analysis. These specialized deep learning models are designed to automatically extract relevant features from images. In the plant disease detection system, CNNs play a crucial role in recognizing patterns and textures associated with various diseases. By breaking down complex images into meaningful components, CNNs enable the model to make accurate disease classifications. Their ability to learn hierarchical features from data makes CNNs well-suited for image-based tasks like plant disease detection, where visual patterns are indicative of plant health.

4.2 Tools Used

In a plant disease detection project using machine learning and computer vision, several tools and libraries are commonly used to develop, train, and deploy the model. Some of the key tools and libraries used in such projects include:

4.2.1 Python

Python is the most commonly used programming language for machine learning and computer vision tasks. It provides a wide range of libraries and frameworks for these tasks.

4.2.2 Jupyter Notebook

Jupyter Notebook is often used for interactive coding and data analysis, making it a valuable tool for experimenting with models and visualizing data.

4.2.3 Pytorch

PyTorch is renowned for its dynamic computation graph, a feature that sets it apart. In PyTorch, the graph is constructed on-the-fly, aligning with the natural flow of Python code. This dynamic nature enhances code readability and simplifies debugging, as developers can easily alter the network structure and troubleshoot issues.

PyTorch is celebrated for its Pythonic API, which makes it exceptionally user-friendly. The library's API is often considered more intuitive and straightforward, appealing to researchers, data scientists, and beginners.

One of PyTorch's strengths lies in its active and growing community. There is a wealth of resources, tutorials, and pre-trained models available, making it accessible for newcomers and researchers. The community's contributions keep the framework up-to-date and vibrant.

The flexibility of PyTorch is a hallmark feature. Developers find it remarkably extensible and adaptable. Researchers and experimentalists appreciate the ease with which they can customize and extend PyTorch to suit their specific requirements.

PyTorch's dynamic computation graph makes it an excellent choice for applications that require decision-making at runtime, such as natural language processing and reinforcement learning.

4.2.4 Tensorflow

TensorFlow, developed by Google, is celebrated for its strong presence in production environments and its static computation graph. In TensorFlow, the computational graph is defined upfront, followed by data processing, making it more suitable for optimization and scalability in production settings.

Widespread industry adoption is one of TensorFlow's distinguishing features. It has proven itself as a reliable choice for deploying machine learning models at scale. The framework excels in applications where deep learning is deployed in production and serving, making it a popular choice in the corporate world.

TensorFlow boasts TensorBoard, a powerful visualization tool. TensorBoard is an invaluable asset for visualizing and monitoring the training process, debugging models, and optimizing their performance.

High-level APIs like Keras have been integrated tightly with TensorFlow, providing developers with a convenient and efficient way to build and train neural networks. This integration simplifies the process of model development.

TensorFlow offers TensorFlow Lite (TFLite), a framework designed for deploying models on mobile devices and embedded systems. This feature makes TensorFlow particularly well-suited for mobile applications and Internet of Things (IoT) deployments.

4.3 Language Used

Python language is used in the system due to the following Characterstics :

Simple:

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English (but very strict English!). This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the syntax i.e. the language itself.

Free and Open Source:

Python is an example of a FLOSS (Free/Libre and OpenSource Software). In simple terms, you can freely distribute copies of this software, read the software's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and improved by a community who just want to see a better Python.

Object Oriented:

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

Extensive Libraries:

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML,

4.4 Screenshots

The Following are the screenshots of the result of the project:

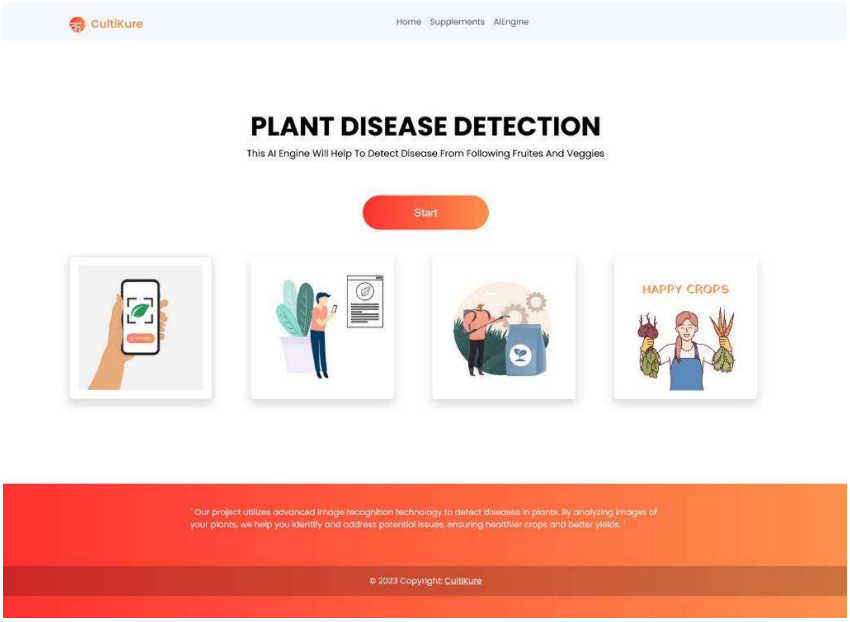

Figure 4-1: Screenshot 1

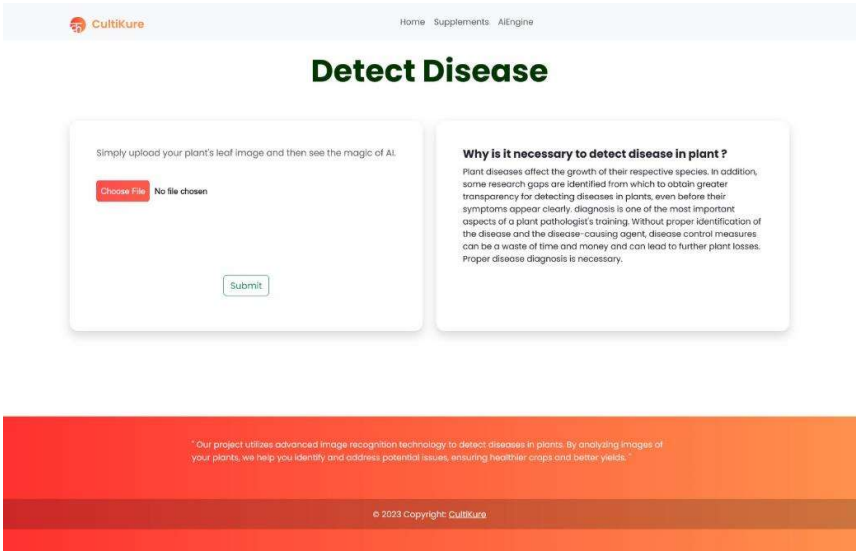

Figure 4-2: Screenshot 2

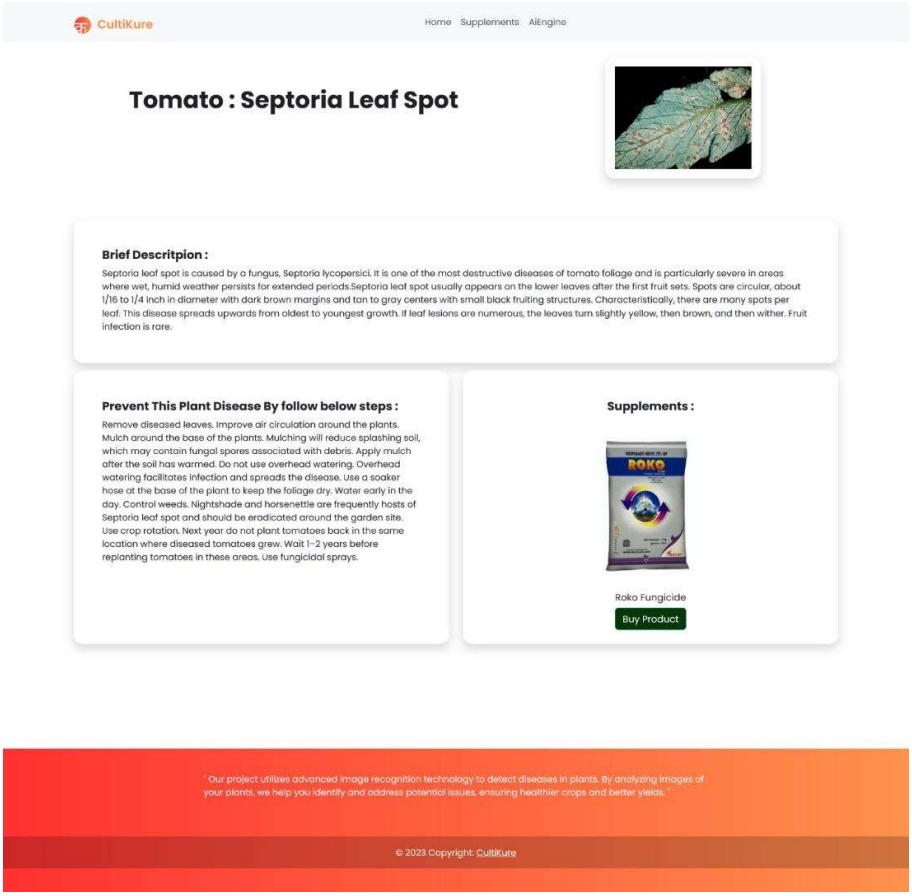

Figure 4-3: Screenshot 3

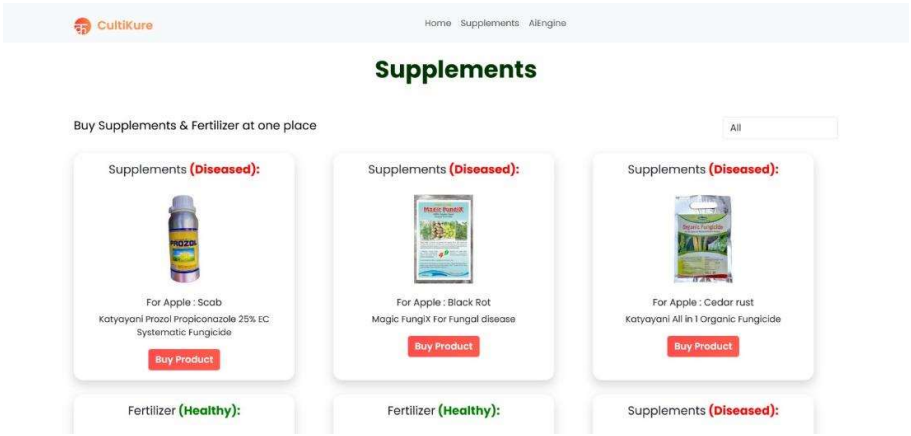

Figure 4-4: Screenshot 4

4.5 Testing

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the feature of the system. Testing assesses the quality of the product. It is a process that is done during the development process.

4.5.1 Strategy Used

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

The texting method used here is Black Box Testing. It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.
The plant disease detection system has undergone various testing methods to ensure its effectiveness and accuracy. One of the testing methods used is unit testing, which involves testing individual components of the system to ensure they are working correctly. Integration testing was also performed to verify that the different components of the system work seamlessly together.

In addition, performance testing was carried out to assess the system's ability to handle a large volume of data and processing demands. This was done to ensure that the system does not become slow or unresponsive when handling multiple requests from users.

Finally, user acceptance testing was conducted to ensure that the system meets the expectations and requirements of end-users. This involved a group of users interacting with the system to verify that it is easy to use, user-friendly, and provides accurate and reliable results.

4.5.2 Results

Sample Input:

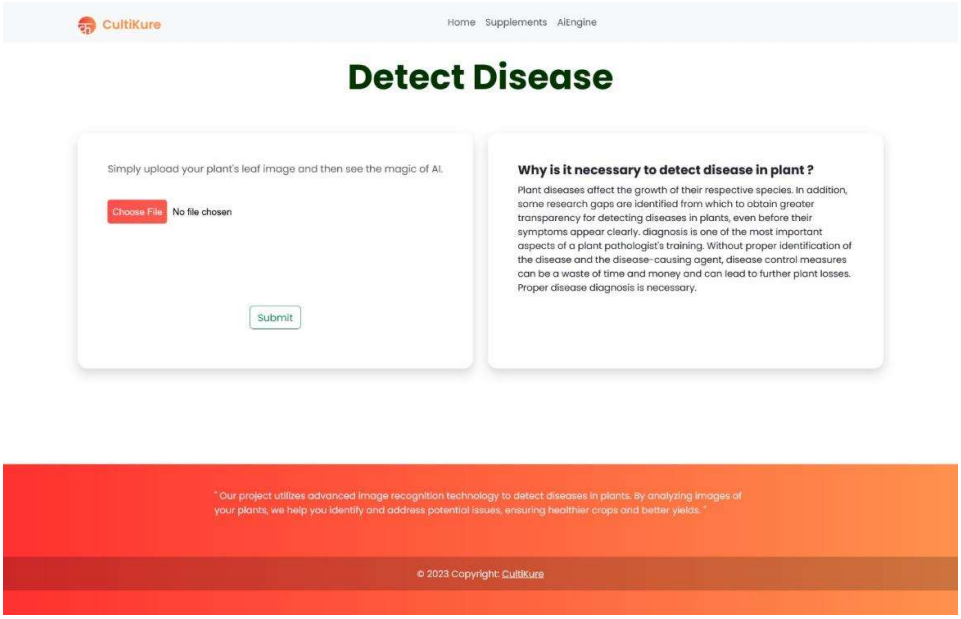

Figure 4-5: Sample Input

OUTPUT:

Chapter 5. Conclusion

Conclusion

5.1 Conclusion

In conclusion, the Plant Disease Detection System represents a pivotal intersection of cutting-edge technology and agricultural sustainability. Leveraging the power of machine learning, image recognition, and deep learning through PyTorch or TensorFlow, this project promises to revolutionize how we safeguard our crops. By harnessing the strengths of dynamic computation graphs and Pythonic API in PyTorch or the scalability and production readiness of TensorFlow, the system has demonstrated its versatility. Furthermore, it incorporates image preprocessing techniques, transfer learning, and convolutional neural networks, which, together, form a robust foundation for precise and efficient plant disease identification. With image processing enhancing data quality, transfer learning providing a head start in model training, and CNNs extracting meaningful features from images, the project showcases the best practices in the field. Its success is further propelled by the active support of a thriving community and an array of valuable resources. Ultimately, this endeavor is poised to empower farmers, gardeners, and agricultural enthusiasts worldwide with the tools needed to protect crops, reduce losses, and promote sustainable farming practices. It stands as a testament to the harmonious integration of technology and agriculture, championing both food security and environmental sustainability in our ever-evolving world.

5.2 Limitations of the Work

- Since no task can be 100% perfect. The same applies to this project as the predicted plant disease is not 100% accurate.

- The models that we are using for identifying the objects are pre-trained models. So, if we want to train our own model, it takes a lot of time and processing.

- The model is trained on a dataset that consists of 39 classes. So if user uploads an image of a plant disease that is not present in the dataset, then the model can't predict it as there can be hundreds of potential diseases that are not present in the dataset.

5.3 Suggestion and Recommendations for Future Work

- The future scope of this idea is to scale up the dataset and accommodate more classes of the diseases of the plants.
- Integrating the system with real-time data streams from IoT devices and drones could enable continuous monitoring and early detection of diseases, leading to proactive intervention.
- Developing a user-friendly mobile app to complement the web interface would make it more accessible to farmers, allowing them to easily capture and analyze plant images in the field.
- Implementing a feedback loop that collects user feedback and disease outcome data could contribute to continuous model improvement and system optimization.

Bibliography

- Anand H. Kulkarni, Ashwin Patil R. K., Applying image processing technique to detect plant diseases, International Journal of Modern Engineering Research, vol.2, Issue.5, pp: 3661-3664, 2012.

- Tushar H. Jaware, Ravindra D. Badgujar and Prashant G. Patil, Crop disease detection using image segmentation, National Conference on Advances in Communication and Computing, World Journal of Science and Technology, pp:190- 194, Dhule, Maharashtra, India, 2012.

- Prof.Sanjay B. Dhaygude, Mr.Nitin P. Kumbhar, Agricultural plant Leaf Disease Detection Using Image Processing, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering , S & S Publication vol. 2, Issue 1, pp: 599-602, 2013.

Source Code

S.1FRONTEND CODE

S.1.1 base.html

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;400;600;700&display=swap"
rel="stylesheet">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.3.0/css/all.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bttn.css/0.2.4/bttn.css">
  <title>{% block pagetitle %}{% endblock pagetitle %}</title>
  <style>
    .bg {
      background-image: url("images/background.jpeg");
      height: 100%;
      background-position: center;
      background-repeat: no-repeat;
      background-size: cover;
    }
    .file-upload input[type='file'] {
      display: none;
    }
    body {
      background-color: white;
      height: 100vh;
      font-family: 'Poppins', sans-serif;
      font-size: 15px;
    }
    .rounded-lg {
      border-radius: 1rem;
    }
    .custom-file-label.rounded-pill {
      border-radius: 50rem;
    }
    .custom-file-label.rounded-pill::after {
      border-radius: 0 50rem 50rem 0;
    }
```

```
        label {
            background-color: rgb(2, 46, 15);
            color: white;
            padding: 0.5rem;
            font-family: sans-serif;
            border-radius: 0.3rem;
            cursor: pointer;
            margin-top: 1rem;
        }
        #file-chosen {
            margin-left: 0.3rem;
            font-family: sans-serif;
        }
        .footer-basic {
            margin-top: 10%;
            padding: 40px 0;
            background-color: #042908;
            color: #f1f3f5;
        }
        .sticky {
            position: fixed;
            bottom: 0;
            width: 100%;
            padding-left: 10%;
        }
        #navf {
            display: none;
        }
        @media (max-width: 767px) {
            #navf {
                display: block;
                width: 100%;
            }
            .ff {
                font-size: 10px;
            }
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container">
            <a class="navbar-brand" href="/" style="font-size: 20px; font-weight: 600; color: #ff914d;
margin-top: 10px;"><img src="static/uploads/front.png" style="width: 30px; margin-right: 10px;"
srcset="">CultiKure</a>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav mx-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class "nav-item">
                        <a class="nav-link" href="./market">Supplements</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="./index">AiEngine</a>
                    </li>
                </ul>
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="./contact"></a>
                    </li>
```

```
            </ul>
          </div>
        </div>
    </nav>
    {% block body %}{% endblock body %}
    <div class="bg-light sticky shadow bg-white rounded" id="navf" style="z-index: 1">
      <div class="row">
        <div class="col">
          <div class="nav-item active" style="font-size: 25px; color: #6C5F5B; margin-right: 65px;
padding-top: 3px;">
            <a class="nav-link" href="/"><i class="fa-sharp fa-solid fa-house"></i> </a>
            <p class="ff">Home</p>
          </div>
        </div>
        <div class="col">
          <div class="nav-item" style="font-size: 25px; color: #6C5F5B; margin-right: 65px;
padding-top: 3px">
            <a class="nav-link" style="font-size: 25px;" href="./index"><i class="fa-sharp fa-solid
fa-seedling"></i></a>
            <p class="ff">Detect</p>
          </div>
        </div>
        <div class="col">
          <div class="nav-item" style="font-size: 25px; color: #6C5F5B; margin-right: 50px;
padding-top: 3px">
            <a class="nav-link" style="font-size: 25px;" href="./market"><i class="fa-sharp fa-solid
fa-bag-shopping"></i></a>
            <p class="ff">Store</p>
          </div>
        </div>
      </div>
    </div>
    <div class="footer-basic" id="footer" style="background-image: linear-gradient(270deg, #ff914d,
#ff3131);">
      <div class="container">
        <!-- Section: Text -->
        <section class="mb-5">
          <div class="row d-flex justify-content-center">
            <div class="col-lg-8">
              <p>
                "  Our project utilizes advanced image recognition technology to detect diseases in
plants. By analyzing images of your plants, we help you identify and address potential issues,
ensuring healthier crops and better yields.   "
              </p>
            </div>
          </div>
        </section>
      </div>
      <!-- Copyright -->
      <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2)">
        © 2023 Copyright:
        <a class="text-white">CultiKure</a>
      </div>
    </div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.3/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

S.1.2 Index.html

```
{% extends 'base.html' %}
{% block pagetitle %}AI Engine{% endblock pagetitle %}
{% block body %}
<div>
  <div class="container">
    <!-- For demo purpose -->
    <div class="row mb-5 text-center text-white">
      <div class="col-lg-10 mx-auto">
        <h1 class="display-4" style="padding-top: 2%; font-weight: 400; color: rgb(4, 54,
4);"><b>Detect Disease</b></h1>
      </div>
    </div>
    <!-- End -->
    <div class="row ">
      <div class="col mx-auto col-sm-12 col-md-6 col-lg-6">
        <div class="p-5 bg-white shadow rounded-lg" style="height: 100%;">
          <h6 class="text-center mb-4 text-muted">
            Simply upload your plant's leaf image and then see the magic of AI.
          </h6>
          <!-- Default bootstrap file upload -->
          <form action="/submit" method="POST" enctype="multipart/form-data">
            <div class="custom-file overflow-hidden mb-4" style="padding-bottom: 20%;">
              <input type="file" id="actual-btn" hidden name="image" />
              <label for="actual-btn" style="background-image: linear-gradient(300deg, #ff5c4d,
#ff4d4d); color: white">Choose File</label>
              <span id="file-chosen" style="color: black">No file chosen</span>
            </div>
            <!-- End -->
            <!-- Custom bootstrap upload file -->
            <center>
              <a class="mx-2"><button type="submit" class="btn btn-outline-
success">Submit</button></a>
            </center>
          </form>
          <!-- End -->
        </div>
      </div>
      <div class="col mx-auto col-sm-12 col-md-6 col-lg-6">
        <div class="p-5 bg-white shadow rounded-lg" style="height: 100%; margin-bottom: 5%;">
          <h5><b>Why is it necessary to detect disease in plant ?</b></h5>
          <p>Plant diseases affect the growth of their respective species. In addition, some research
gaps are identified from which to obtain greater transparency for detecting diseases in plants, even
before their symptoms appear clearly. Diagnosis is one of the most important aspects of a plant
pathologist's training. Without proper identification of the disease and the disease-causing agent,
disease control measures can be a waste of time and money and can lead to further plant losses. Proper
disease diagnosis is necessary.</p>
        </div>
      </div>
    </div>
  </div>
</div>
<script>
  const actualBtn = document.getElementById('actual-btn');
  const fileChosen = document.getElementById('file-chosen');
  actualBtn.addEventListener('change', function () {
    fileChosen.textContent = this.files[0].name;
  });
</script>
{% endblock body %}
```

S.1.3 market.html

```
{% extends 'base.html' %}
{% block pagetitle %}Supplement Market{% endblock pagetitle %}
{% block body %}
<div class="container">
   <div class="row mb-5 text-center text-white">
      <div class="col-lg-10 mx-auto">
         <h1 class="display-6" style="padding-top: 2%; font-weight: 400; color: rgb(4, 54,
4);"><b>Supplements</b></h1>
      </div>
   </div>
   <div class="row">
       <div class="col-lg-10 mx-auto">
         <p class="lead" style="font-weight: 500; color: black;">Buy Supplements & Fertilizer at one
place</p>
      </div>
      <div class="col-lg-2 mb-4">
         <select id="filterDropdown" class="form-control">
            <option value="all">All</option>
            <option value="fertilizer">Fertilizer</option>
            <option value="supplements">Supplements</option>
         </select>
      </div>
   </div>
   <div class="row">
      <!-- Your item listing code here (similar to the code you provided) -->
      {% for index in range(supplement_name | length) %}
         {% if index != 4 %}
            <div class="col-lg-4 col-md-6 col-sm-12 mb-4">
               <div class="p-3 bg-white shadow rounded-lg" style="height: 100%; width: 90%;">
                  <center>
                     {% if index in [3, 5, 7, 11, 15, 18, 20, 23, 24, 25, 28, 38] %}
                     <h5>Fertilizer<b style="color: green;"> (Healthy):</b></h5>
                     {% else %}
                     <h5>Supplements<b style="color: red;"> (Diseased):</b></h5>
                     {% endif %}
                     <br>
                     <img src={{supplement_image[index]}} width="100" height="150">
                     <br>
                     <br>
                     <h6>For {{disease[index]}}</h6>
                     <p>{{supplement_name[index]}}</p>
                     <a target="_blank" href={{buy[index]}}>
                        <button type="button" class="btn btn-light" style="background-image: linear-
gradient(300deg, #ff5c4d, #ff4d4d); color: white; font-weight: 600;">Buy Product</button>
                     </a>
                  </center>
               </div>
            </div>
         {% endif %}
      {% endfor %}
   </div>
</div>
<script>
   document.getElementById("filterDropdown").addEventListener("change", function () {
      var selectedFilter = this.value;
```

```
        var items = document.querySelectorAll(".col-lg-4");
        items.forEach(function (item) {
          var itemType = item.querySelector("h5").innerText.includes("Fertilizer") ? "fertilizer" :
"supplements";
          if (selectedFilter === "all" || itemType === selectedFilter) {
            item.style.display = "block";
          } else {
            item.style display = "none";
          }
        });
    });
</script>
{% endblock body %}
```

S.1.4 submit.html

```
{% extends 'base.html' %}
{% block pagetitle %}{{title}}{% endblock pagetitle %}
{% block body %}
<div>
  <div class="container">
    <!-- For demo purpose -->
    <div class="row mb-5 text-center text-white">
      <div class="col-lg-7 mx-auto">
        <h3 class="display-6" style="padding-top: 10%; font-weight: 400; color: rgb(33, 32,
35);"><b>{{title}}</b></h3>
      </div>
      <div class=" col-lg-5 mx-auto">
        <div class="p-3 bg-white shadow rounded-lg" style="width: 50%; margin-top:5%; margin-left:
25%;">
          <img src={{image_url}} style="width: 100%;">
        </div>
    </div>
    </div>
    <center>
      </center>
      <br>
      <div class=" row ">
        <div class="col mx-auto">
          <div class="p-5 bg-white shadow rounded-lg" style="height: 95%;">
            <h5><b>
                {% if pred==3 or pred==5 or pred==7 or pred==11 or pred==15 or pred==18 or pred==20
or pred==23 or pred==24 or pred==25 or pred==28 or pred==38 %}
                Tips to Grow Healthy Plants :
                {% else %}
                Brief Descritpion :
                {% endif %}
              </b></h5>
            <p id="content">{{desc}}</p>
          </div>
        </div>
      </div>
      <div class="row">
        <div class="col mx-auto">
          <div class="p-5 bg-white shadow rounded-lg" style="height:95%">
            <h5><b>
                {% if pred==3 or pred==5 or pred==7 or pred==11 or pred==15 or pred==18 or pred==20
or pred==23 or pred==24 or pred==25 or pred==28 or pred==38 %}
                Benefits :
```

```
            {% else %}
            Prevent This Plant Disease By follow below steps :
            {% endif %}
          </b></h5>
        <p>
          {{prevent}}
        </p>
      </div>
    </div>
    {% if pred!=4 %}
    <div class="col mx-auto"">
    <div class=" p-5 bg-white shadow rounded-lg" style="height: 95%;">
      <center>
        <h5><b>
            {% if pred==3 or pred==5 or pred==7 or pred==11 or pred==15 or pred==18 or pred==20
or pred==23 or pred==24 or pred==25 or pred==28 or pred==38 %}
            Fertilizer :
            {% else %}
            Supplements :
            {% endif %}
          </b></h5>
        <br>
        <img src={{simage}} width="200" height="250">
        <br>
        <br>
        <h6 id="content">{{sname}}</h6>
        <a target="_blank" href={{buy_link}}><button type="button" class="btn btn-success"
style="background-color: #05380b;">Buy Product</button></a>
      </center>
    </div>
  </div>
  {% endif %}
 </div>
</div>
</div>
<script>
  function changeContent() {
    var contentElement = document.getElementById('content');
    contentElement.innerHTML = '{{hin}}';
  }
</script>
{% endblock body %}
```

BACKEND CODE

S.2.1 app.py

```
# pip install torch
# pip install torchvision
# pip install pixellib
# pip install pycocotools

import os
from flask import Flask, redirect, render_template, request
from PIL import Image
import torchvision.transforms.functional as TF
import CNN
import numpy as np
import torch
import pandas as pd

# Load disease and supplement info from CSV files
disease_info = pd.read_csv('disease_info.csv', encoding='cp1252')
supplement_info = pd.read_csv('supplement_info.csv', encoding='cp1252')

# Load the pre-trained CNN model
model = CNN.CNN(39)  # Initialize your CNN model with the appropriate number of classes
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()

# Define a function for making predictions
def prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    if 0 <= index < 39:
        return index
    else:
        return -1

app = Flask(__name__)
```

```
# Define routes for different pages
@app.route('/')
def home_page():
    return render_template('home.html')
@app.route('/index')
def ai_engine_page():
    return render_template('index.html')
@app.route('/mobile-device')
def mobile_device_detected_page():
    return render_template('mobile-device.html')
# Handle image submission and make predictions
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        image = request.files['image']
        filename = image.filename
        file_path = os.path.join('static/uploads', filename)
        image.save(file_path)
        print(file_path)
        pred = prediction(file_path)
        title = disease_info['disease_name'][pred]
        description = disease_info['description'][pred]
        prevent = disease_info['Possible Steps'][pred]
        image_url = disease_info['image_url'][pred]
        supplement_name = supplement_info['supplement name'][pred]
        supplement_image_url = supplement_info['supplement image'][pred]
        supplement_buy_link = supplement_info['buy link'][pred]
        return render_template('submit.html', title=title, desc=description, prevent=prevent,
                               image_url=image_url, pred=pred, sname=supplement_name,
                               simage=supplement_image_url, buy_link=supplement_buy_link)

# Route to a market page to display supplement information
@app.route('/market', methods=['GET', 'POST'])
def market():
   return render_template('market.html', supplement_image=list(supplement_info['supplement image']),
   supplement_namee=list(supplement_info['supplement name']),
    disease=list(disease_info['disease_name']), buy=list(supplement_info['buy link']))
if __name__ == '__main__':
    app.run(debug=True)
```

S.2.2 CNN.py

```
import pandas as pd
import torch.nn as nn


class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2),
            # conv2
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(64),
            nn.MaxPool2d(2),
            # conv3
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2),
            # conv4
            nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.BatchNorm2d(256),
            nn.MaxPool2d(2),
        )
```

```
        self.dense_layers = nn.Sequential(
            nn.Dropout(0.4),
            nn.Linear(50176, 1024),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(1024, K),
        )

    def forward(self, X):
        out = self.conv_layers(X)
        # Flatten
        out = out.view(-1, 50176)
        # Fully connected
        out = self.dense_layers(out)
        return out

# Mapping of class indices to class names
idx_to_classes = {
    -1: "No Classes",
    0: 'Apple___Apple_scab',
    1: 'Apple___Black_rot',
    2: 'Apple___Cedar_apple_rust',
    3: 'Apple___healthy',
    4: 'Background_without_leaves',
    # ... (other class mappings) ...
    38: 'Tomato___healthy'
}
```

MODEL

Model.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import torch
from torchvision import datasets, transforms, models
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime

%load_ext nb_black

# Import Dataset
transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
)
dataset = datasets.ImageFolder("Dataset", transform=transform)
indices = list(range(len(dataset))
split = int(np.floor(0.85 * len(dataset)))  # train_size
validation = int(np.floor(0.70 * split))  # validation
print(0, validation, split, len(dataset))
np.random.shuffle(indices)
# Split into Train and Test
train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:],
)
train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)

targets_size = len(dataset.class_to_idx)

# Model
class CNN(nn.Module):
```

```
def __init__(self, K):
    super(CNN, self).__init__()
    self.conv_layers = nn.Sequential(
        # conv1
        nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(32),
        nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(32),
        nn.MaxPool2d(2),
        # conv2
        nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(64),
        nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(64),
        nn.MaxPool2d(2),
        # conv3
        nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.MaxPool2d(2),
        # conv4
        nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.MaxPool2d(2),
    )
    self.dense_layers = nn.Sequential(
        nn.Dropout(0.4),
        nn.Linear(50176, 1024),
        nn.ReLU(),
        nn.Dropout(0.4),
```

```
            nn.Linear(1024, K),
        )

    def forward(self, X):
        out = self.conv_layers(X)

        # Flatten
        out = out.view(-1, 50176)
        # Fully connected
        out = self.dense_layers(out)
        return out

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device = "cpu"
model = CNN(targets_size)
model.to(device
# Criterion and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())
# Batch Gradient Descent
def batch_gd(model, criterion, train_loader, test_laoder, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)
    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in train_loader:
            inputs, targets = inputs.to(device), targets.to(device)

            optimizer.zero_grad()
            output = model(inputs)
            loss = criterion(output, targets)
            train_loss.append(loss.item())

            loss.backward()
            optimizer.step()

        train_loss = np.mean(train_loss)
        validation_loss = []
        for inputs, targets in validation_loader:
            inputs, targets = inputs.to(device), targets.to(device)
```

```
            output = model(inputs)
            loss = criterion(output, targets)
            validation_loss.append(loss.item())
        validation_loss = np.mean(validation_loss)
        train_losses[e] = train_loss
        validation_losses[e] = validation_loss
        dt = datetime.now() - t0
        print(
            f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f} Test_loss:{validation_loss:.3f}
Duration:{dt}")
    return train_losses, validation_losses
batch_size = 64
train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
validation_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=validation_sampler
)
train_losses, validation_losses = batch_gd(
    model, criterion, train_loader, validation_loader, 5
)
# Save the Model
# torch.save(model.state_dict(), 'plant_disease_model_1.pt')
# Load Model
model = CNN(targets_size)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()
# Plot the loss
plt.plot(train_losses, label="train_loss")
plt.plot(validation_losses, label="validation_loss")
plt.xlabel("No of Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
# Accuracy
def accuracy(loader):
    n_correct = 0
    n_total = 0
```

```
    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)
        outputs = model(inputs)
        _, predictions = torch.max(outputs, 1)
        n_correct += (predictions == targets).sum().item()
        n_total += targets.shape[0]
    acc = n_correct / n_total
    return acc
train_acc = accuracy(train_loader)
test_acc = accuracy(test_loader)
validation_acc = accuracy(validation_loader)
print(
    f"Train Accuracy : {train_acc}\nTest Accuracy : {test_acc}\nValidation Accuracy :
{validation_acc}")
# Single Image Prediction
transform_index_to_disease = dataset.class_to_idx
transform_index_to_disease = dict(
    [(value, key) for key, value in transform_index_to_disease.items()]
)
data = pd.read_csv("disease_info.csv", encoding="cp1252")
from PIL import Image
import torchvision.transforms.functional as TF
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease
```

GUIDE INTERACTION REPORT

Date	Discussion	Action Plan
	Discussed about the Title of the project	We discussed and finalized the project title, which is "Developing a System for Farmers."
	Discussion on the technology to be used for the image detection	We decided to use Flask, CNN (Convolutional Neural Network), and other tools for image detection.
	Discussion of the creation of the synopsis of the project	We initiated the process of gathering information to create a project synopsis.
	Suggestions on how to do a literature survey and preliminary investigation on the topic	We conducted a literature survey, reading and understanding various research papers and summarizing their abstracts.
	Discussion on the implementation of the project(using CNN to develop an efficient working system)	We decided to incorporate CNN to create an efficient disease detection system.
	Adding a Extra Feature	We discussed adding an extra feature that redirects users to a website for purchasing fertilizers and pesticides.
	Discussion on Project Documentation	We decided to write and integrate project content into a properly formatted report.