Smart Agriculture Crop Disease Detection using DL for Enhanced Crop Health Monitoring

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Himanshu Jindal (201527)

Mehak Chauhan (201571)

Vanshika Bhardwaj (201186)

Under the guidance & supervision of

Dr. Ravindra Bhatt (Associate Professor)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidate's Declaration

I hereby declare that the work presented in this report entitled **Smart Agriculture Crop Disease Detection using Learning for Enhanced Crop Health Monitoring** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Ravindara Bhatt** (Associate Professor, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Himanshu Jindal
Roll No.: 201527

Student Name: Mehak Chauhan
Roll No.: 201571

Student Name: Vanshika Bhardwaj
Roll No.: 201186

This is to certify that the above statement made by the candidate is true to the best of our knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Ravindara Bhatt
Designation: Associate Professor
Department: CSE & IT

Acknowledgement

With immense gratitude, we extend our heartfelt thanks to the Almighty for His divine blessings, which have illuminated our path and enabled us to successfully complete the project – Smart Agriculture Crop Disease Detection using Learning for Enhanced Crop Health Monitoring.

Our sincere appreciation goes to our esteemed Supervisor, Dr. Ravindara Bhatt, Associate Professor in the Department of CSE at Jaypee University of Information Technology, Wakhnaghat. Dr. Bhatt's profound expertise in the realms of Data Science, and machine/deep learning has been instrumental in guiding us through this project. We are deeply indebted for her tireless support, patient mentorship, constructive criticism, and unwavering encouragement. Her keen interest in our work has truly been a driving force behind the project's completion.

We would also like to express our gratitude to Dr. Ravindara Bhatt from the Department of CSE for her valuable assistance, which significantly contributed to the successful conclusion of our project.

Our sincere thanks extend to all individuals, whether directly or indirectly, who played a role in the triumph of this project. We acknowledge the support of the entire staff, both teaching and non-teaching, whose timely assistance and facilitation greatly aided our endeavor.

In closing, we wish to recognize and appreciate the enduring support and patience of our parents. Their unwavering encouragement has been a source of strength throughout this journey.

With gratitude,
Himanshu Jindal
(201527)
Mehak Chauhan
(201571)
Vanshika Bhardwaj
(201186)

Table of Content

List of Figures

Abstract

Smart agriculture is revolutionizing traditional farming practices by integrating cutting-edge technologies to enhance crop productivity and mitigate agricultural challenges. Crop diseases are one such issue that has a big impact on production and food security. This initiative aims to improve crop health and boost agricultural productivity by utilizing machine learning approaches for effective crop disease identification and monitoring.

The suggested system gathers data from agricultural areas by using a variety of sensors and image-capture devices. Machine learning models are fed data from these sources, which include pictures of crops, environmental factors, and past disease trends. The gathered data is analyzed using Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or ensemble models to find any crop illnesses or anomalies.

The results can be visualized and interpreted by farmers and agricultural specialists thanks to the system's user-friendly interface. Real-time alerts and notifications are generated, giving immediate information regarding diseases that have been identified, suggested courses of action, and preventive measures. The experiment also highlights how important it is to have a large dataset in order to train machine learning models. To ensure the models' accuracy and robustness, the dataset is carefully selected to include a variety of photos showing both healthy crops and varied disease situations under various environmental conditions.

Farmers may proactively detect diseases in their early stages and treat them with precision and targeting by putting this Smart Agriculture Crop Disease Detection system into practice. As a result, this strategy reduces crop losses, maximizes the use of available resources, and supports sustainable farming methods.

To sum up, this project intends to transform traditional farming methods by incorporating machine learning approaches into agriculture, offering a clever framework for anticipatory crop disease detection, encouraging improved crop health monitoring, and eventually enhancing global food security.

Chapter 1: Introduction

1.1 Introduction

The foundation of human civilization, agriculture, faces constant pressures from shifting environmental circumstances and a world population that is rising at an exponential rate. The agricultural sector has several challenges due to the urgent need to feed the world's population, most notably crop diseases that have a substantial influence on crop productivity and quality. Many times, traditional disease monitoring and detection techniques are unable to provide prompt and precise identification, which results in significant losses for agriculture.

However, the incorporation of creative ideas into conventional agricultural methods in this age of technological developments has ushered in a new era of agriculture known as "Smart Agriculture". Innovations in this paradigm enhance crop productivity, reduce use of resources, and adequately solve agricultural problems by applying modern technologies like artificial intelligence, machine learning, and IoT (Internet of Things).

Smart agriculture involves employing machine learning algorithms to identify and combat crop diseases. The aim of this study is in a crucial area using the machine learning approach in converting crop health monitoring. This system uses data collected from different sources like sensors, drones and other capture devices to predict, identify, and control diseases in crops accurately and promptly.

This project has the integration of technology with agronomy as the center. These machine learning algorithms help to identify patterns in a set of data and distinguish among healthy and ailing crops depending upon some environmental facts, past diseased pattern as well as visible indicators. Farmers will be able to act promptly and sensibly against disease spreading through the detection process.

The project also highlights the significance of a complete dataset, which is a set of varied photos showing both healthy crops and a wide range of disease conditions under various environmental conditions. This dataset serves as the basis for the development of strong machine learning models that are proficient in the precise identification and categorization of crop diseases.

In the end, integrating Smart Agriculture methods for crop disease detection seeks to improve overall crop health, maximize resource utilization, and considerably increase global food security in addition to reducing the negative effects of diseases. Agriculture breaks through its traditional boundaries with this creative approach, opening the door to a farming ecosystem that is more efficient and sustainable.

1.2 Problem Statement

The main problems are caused by the shortcomings of traditional paper-based testing systems, The health and vitality of crops are key determinants of agricultural productivity and food security in the context of modern agriculture. However, the widespread spread of agricultural diseases continues to be a problem, resulting in significant financial losses and endangering the world's food supplies. Traditional illness detection techniques frequently involve physical labour, take a long time, and are prone to mistakes, which delay responses and cause damage. The proposed project aims to utilize the capabilities of deep learning technology to create a comprehensive crop disease detection system together with an integrated platform for Enhanced Crop Health Monitoring in order to address this crucial issue.

The creation and implementation of a cutting-edge deep learning model for the automated detection and categorization of agricultural diseases is the main goal of this research. The model will be created using the PyTorch framework architecture, enabling precise illness diagnosis from photos of plant leaves. The deep learning model will be trained to distinguish between healthy and disease-infected crops with high precision using a rigorously curated dataset and extra data collecting via web scraping.

The project plans to provide a comprehensive platform that offers farmers real-time information, predictive analytics, and a collaborative setting in addition to the deep learning model. With the help of this platform's user-friendly design, which was created using React and Tailwind CSS farmers will be able to easily upload pictures of their crops and quickly get illness diagnoses. The technology will also have the ability to trace IP addresses, making it easier to identify local disease outbreaks early and notify farmers in a timely manner. The system will also gradually forecast potential disease outbreaks using self-learning algorithms, allowing farmers to take preventative action.

The incorporation of a specific forum for farmers demonstrates the project's emphasis on community involvement and knowledge-sharing. This forum will promote cooperation by allowing farmers to share knowledge, perspectives, and suggestions pertaining to crop health and disease control. Additionally, the platform's ability to handle several languages would enable farmers from various regions to communicate with the system in their chosen regional languages of India, providing accessibility and inclusivity.

1.3 Objectives

The primary objectives of this project are outlined below:

1. The goal is to give farmers a proactive tool for early detection and accurate identification of crop illnesses by developing a scalable and reliable machine learning-based system for Smart Agriculture Crop Disease Detection.
2. To efficiently train and test machine learning models, gather and curate a broad dataset that includes photos of crops in various states of health as well as photographs of diseases under various environmental circumstances.
3. Use modern machine learning algorithms, such as ensemble models, recurrent neural networks (RNNs), and convolutional neural networks (CNNs), to analyse multi-source data, including sensor data, images, and environmental parameters, in order to accurately detect crop diseases.
4. Provide an easy-to-use interface that makes it possible for farmers and agricultural specialists to communicate with the system, view the diseases that have been identified, get alerts in real time, and see suggestions for the best course of action and preventive measures.
5. Assure the system's scalability and adaptability to a range of crops, diseases, and agricultural environments so that it can be widely implemented in a variety of agricultural settings and geographical locations.
6. Measure the precision, sensitivity, and specificity of disease identification to ascertain the dependability and efficacy of the system by subjecting it to rigorous testing in actual agricultural situations.

7. Provide thorough documentation, instructions, and support to farmers so they can successfully implement and use the Smart Agriculture Crop Disease Detection system, thereby facilitating knowledge transfer and the adoption of the developed technology.

1.4 Significance and Motivation of the Project Work

The importance and impetus behind the creation of a machine learning-based Smart Agriculture Crop Disease Detection system are numerous and crucial for the state of agriculture today. The main objective of this project is to bring about a proactive approach to crop health maintenance and monitoring, thereby revolutionizing traditional farming practices. The main driving force behind this system is its capacity to detect crop diseases early on, allowing for prompt interventions and reducing significant losses in crop productivity. It seeks to maintain crop productivity while minimizing financial losses for farmers by utilizing cutting-edge technologies.

The potential of this project to optimize resource utilization in agriculture is a crucial component of its significance. Conventional farming practices frequently utilize resources and pesticides carelessly, which deteriorates the environment. However, this initiative aims to reduce needless chemical usage by promoting more sustainable farming practices through accurate disease detection made possible by machine learning. Putting this system in place also gives farmers access to technology tools that give them real-time information about their crops. By boosting agricultural productivity, this information facilitates informed decision-making and enhances agricultural results, ultimately enhancing global food security.

1.5 Organization of Project Report

Project report "Smart Agriculture Crop Disease Detection using Learning for Enhanced Crop Health Monitoring " provides an in-depth review of the initiative from start to finish, organized in an intricate way. With a smooth and succinct narrative, each chapter develops methodically, guiding readers through the project's nuances and unique contributions.

Chapter 1: Introduction

The introductory chapter establishes the framework for the project by presenting the central theme, addressing recognized obstacles, delineating goals, and highlighting the importance and inspiration behind "Smart Agriculture Crop Disease Detection". Finally, the section offers a well-organized roadmap for reading the next few chapters.

Chapter 2: Literature Survey

This chapter delves into scholarly investigation and presents a succinct summary of pertinent literature while incorporating ideas from a variety of sources. It strategically highlights the most important gaps in the body of current knowledge in order to identify and explain the project's distinctive contributions.

Chapter 3: System Development

This crucial chapter describes the steps involved in the project's journey: requirements and analysis at the outset, design and architecture, data preparation, and finally project implementation using snippets of code and skillful problem-solving techniques.

Chapter 4: Testing

The critical testing phase is covered in detail in the fourth chapter, along with the tools that were selected, the nuances of the test cases, and the results. It guarantees a comprehensive analysis of the functionality and dependability of the project.

Chapter 5: Results and Evaluation

Project outcomes are the main focus of this chapter, which also offers a perceptive look at the results and a focused display of the results.

Chapter 6: Conclusions and Future Scope

The final chapter highlights the contributions of the project, summarizes the main conclusions, and notes its limitations. By outlining a wide range of possible future developments, it opens up the project's domain for future advancements and expansions.

Chapter 2: Literature Survey

2.1 Overview of Relevant Literature

Sharada P.Mohanty [16] evaluate the applicability of deep convolutional neural networks for the classification problem described above. We focus on two popular architectures, namely AlexNet (Krizhevsky et al., 2012), and GoogLeNet (Szegedy et al., 2015), which were designed in the context of the "Large Scale Visual Recognition Challenge" (ILSVRC) (Russakovsky et al., 2015) for the ImageNet dataset (Deng et al., 2009).

Omkar Kulkarni [10] proposes a deep learning-based model which is trained using public dataset containing images of healthy and diseased crop leaves. The model serves its objective by classifying images of leaves into diseased category based on the pattern of defect. Following testing on a trained model, it was discovered that the model trained with segmented images outperforms the model trained with color and grayscale images. It is observed that InceptionV3 model performs better than MobileNet in the task of crop detection.

LILI LI [2] has introduced the basic knowledge of deep learning and presented a comprehensive review of recent research work done in plant leaf disease recognition using deep learning. Provided sufficient data is available for training, deep learning techniques are capable of recognizing plant leaf diseases with high accuracy. Most of the DL frameworks proposed in the literature have good detection effects on their datasets, but the effects are not good on other datasets, that is the model has poor robustness. Therefore, better robustness DL models are needed to adapt the diverse disease datasets.

An in depth review entitled "A comprehensive review on detection of plant disease using machine learning and deep learning approaches" [1] covers the application of ML and DL techniques for plant disease recognition. The coverage encompasses different ML and DL approaches applied to the recognition and diagnosis of plant diseases from any collection of images or pictures. The authors provide a short history of the development of these techniques that started from traditional ML approaches and reached the current state-of-the-art neural networks, especially CNNs. It focuses on comparing different techniques and why their DL models work best because they can automatically extract more complicated features from images. This paper assesses the usefulness of transfer learning, data augmentation, and ensemble techniques for improving the reliability and precision of automated disease discovery platforms. It also addresses issues related to dataset quality, model interpretability, scalability, and generalization across multiple plant species and environmental conditions. In general, the comprehensive review is a good source that provides a summary of latest methodologies and also gives recommendations for improvement of machine learning and deep learning-based plant disease recognition systems.

A paper entitled " Tomato disease detection and classification by using deep learning" [3] investigates the usage of a deep learning approach for the detection and classification of tomato diseases. It explores the use of CNN and other deep learning architecture for analyzing tomato diseases image datasets. These models are reviewed to determine their ability to correctly identify and categorize all kinds of diseases, as well as for their role in early disease diagnosis and classification. This paper examines the advantages of deep learning approaches in dealing with complicated visual information, overcoming constraints, as well as suggestions for the future implementation of such methods for tomato disease control.

Dr. Vishwanath Karad [4] outlines several techniques for automated computerized plant disease detection that can be achieved by applying techniques for image segmentation and classification. Three diseases—Early Blight, Late Blight, and Bacterial Spot—were used to test the algorithms. The primary goal of the suggested strategy is to identify the diseases. The HSV alteration method worked incredibly well for locating the diseased portion of the leaf, and by extracting texture features, GLCM further increased the accuracy. Therefore, by analyzing the data, the suggested method produces minimally computationally complex and accurate plant leaf disease detection.

"Deep learning-based approach for disease identification in maize crop"[14] is an innovative utilization of deep learning methods in identifying and classifying maize diseases. The study seeks to develop and apply a strong system based on advancement in deep learning, especially CNN's, and effectively diagnose most of the common ailments affecting a maize plant. The study will make use of a large set of images on healthy and infected crops for training and fine-tuning deep learning algorithms to accurately classify the type of disease affecting crops. Deep learning approach could prove an effective and cost-efficient tool to address early diseases in maize, boosting total output and sustainable agriculture.

Adhao Asmita Sarangdhar [7] describes a new approach that employs the use of machine learning regression technique together with the IoT technology in developing a system capable of identifying as well as controlling disease incidences such as leaf disease on cotton leaves.

The study is aimed at building regressions as machine learning models for accurate detection and prediction of such diseases. The study's objective is to use sensors as well as data collection nodes placed within the agricultural fields for gathering actual environmental and plant health information in real time as a result of integrating them with IoT devices. The machine learning model makes use of this information and gives warnings to farmers before an outbreak occurs. This combined method can change how diseases are detected on cotton plants and interventions made earlier that can increase production and the longevity of cotton crops.

Chongake Bi [20] research report is a novel approach that uses MobileNet, a model made up of a lightweight convolutional neural network, for diagnosing and classifying apple leaf diseases. The objective of this study is to devise an effective and reliable system that can identify those diseases most likely to affect apple trees in farming. The study is aimed at developing lightweight model that takes advantage of the MobileNet-based architecture known for computational speed and feature extraction. The study uses mobile net, a convolutional network with low parameters that is trained and fine-tuned using a dataset of various images showing healthy and diseased apples' leaves for identifying apple leaf diseases. The paper demonstrates that such approach can provide handy device for fast disease detection in order to facilitate prompt actions and promote efficient cropping and apples production.

The research paper titled "Plant Leaves Classification: An efficient deep few-shot learning method using a Siamese network to classify plant leaves" [5]. In this study, the focus is to build a strong classification system that can classify different species of plants through their leaf images while having few labeled data. Small datasets can be used to teach a model since they share common image similarity metrics with other data sets. The study aims at creating a flexible leaf classification system that relies on the few-shot learning paradigm which enables training on new classes using only a small number of instances. This technique can be considered as one of the approaches with the capacity to solve the problems that are caused by unavailability of labelled data for plant identification using leaf image-based classification.

PENG JIANG [6] presents a new approach, using improved CNNs, which can detect some diseases in apple leaf at real time. The study deals with addressing an important requirement of quick disease discovery in apple orchards using deep learning approach to increase the precision as well as success rate of disease recognition. This study builds and designs special cancerous disease detecting CCN systems designed for apple leaves. The study proposes using improved CNN models to detect apple leaves' diseases in real-time, thus identifying certain apple leaf diseases at a high speed and with a high degree of accuracy. This technique has shown possibilities that can make disease management easier for apple growers through an early disease detection for effective intervention hence leading to healthy apple orchards.

2.2 Key Gaps in the Literature

It is evident that there are still gaps in research related to crop diseases detection by means of machine learning and deep learning. To start with, the concentrated attention only on a few types of crops mainly maize, cotton and apple highlights inexplicable knowledge on disease detection for other crops. Existing studies are not very comprehensive as far as many crops are concerned making it difficult to develop holistic understanding about identification mechanism for diseases. As well, most of the literature examines generic diseases peculiar to different crops, rather than focusing on uncommon illnesses that are rife in various agroecosystems. In addition, a detailed study of diseases attacking most plants would greatly improve multi-disciplinary disease detection methods for different crops.

The other notable lack is related to the use of IoT for comprehensive data collection within the domain of disease detection models. Although some researches briefly allude to them as data collectors, there is relatively scarce insightful discussion on those possibilities. However, investigations into use of IoT devices to acquire imagery in conjunction with environmental parameters, as well as real-time disease surveillance would significantly increase the reliability and effectiveness of such detection systems. Further investigation of this integration could alter disease management approaches by enriching data for training and validation of models thereby facilitating more refined disease identification and forecasting processes.

Besides, though previous research recognizes variability in environment as one problem area, more insight into the influence of models on different environments is needed. For practical application, it is important that we understand how the ML/DL models react to changing environmental conditions in the agriculture sector. Additional studies aiming at strengthening models of robustness and generalization within different environmental circumstances would bolster disease detection mechanisms, confirming its relevance and competence within numerous farming locations. Filling in the holes would lead to advancements towards more flexible, accurate, and useful tools for early detection and control of diseases in many different crops under different weather regimes.

Chapter 3: System Development

3.1 Requirements and Analysis

The project calls for the collection of extensive data, including a variety of photos showing both healthy and sick crops. Applying machine learning algorithms to multi-source data processing allows for precise disease detection in analysis. In order to ensure accessibility and usability for farmers and experts alike, user-friendly interfaces for real-time alerts and visualization are imperative.

Functional Requirements:
In order to accurately detect diseases, the project requires effective data collection from a variety of sources using machine learning algorithms. It also requires real-time alerts and an easy-to-use interface that gives users visualization and actionable insights.

Non-Functional Requirements:
Usability: assesses the ease of use of an interface, including memorability, learnability, and satisfaction. The user-friendliness of the software interface is prioritized, ensuring that users of all backgrounds can quickly navigate and comprehend the system.

Reliability: measures the system's ability to execute consistently on demand without degradation or failure across several platforms.

Integrity: In the context of data, integrity refers to consistently doing the right thing. It places a premium on data accuracy, consistency, and completeness, all of which are critical components of system security.

Performance: is an important non-functional criterion that includes characteristics such as resource requirements, reaction time, and transaction rate. These requirements determine how the system should work.

SDLC Methodology

The Software Development Life Cycle (SDLC) is a methodical procedure meant to ease the creation of high-quality, low-cost software in the shortest possible time frame. The SDLC's primary goal is to create great software that not only meets but exceeds all customer expectations and needs. This method of software development entails a structured plan divided into discrete stages or phases, each with its own set of operations and deliverables.

Following the SDLC closely increases development speed while reducing potential project risks and expenses associated with alternative production processes. The SDLC acts as a guiding framework for the efficient and successful development of software solutions while guaranteeing that the final product meets the needs of the customer.

Why is SDLC Required?

If deadlines are not reached, executing a project without a well-defined action plan can lead to disaster and, eventually, project failure. To ensure the seamless progression of the whole development cycle, every aspect, from resource allocation to deployment, must follow a defined pipeline. This requirement gave rise to the Software Development Life Cycle (SDLC), which, after achieving substantial success, became widely implemented. The SDLC is critical because it provides a systematic approach that is consistent with the whole development process. Its success is based on completing each phase to a high standard while fulfilling customer objectives for cost, time, and efficiency. The SDLC's major purpose is to ensure that the development cycle runs smoothly and that the product is of high quality.

How does the SDLC work?

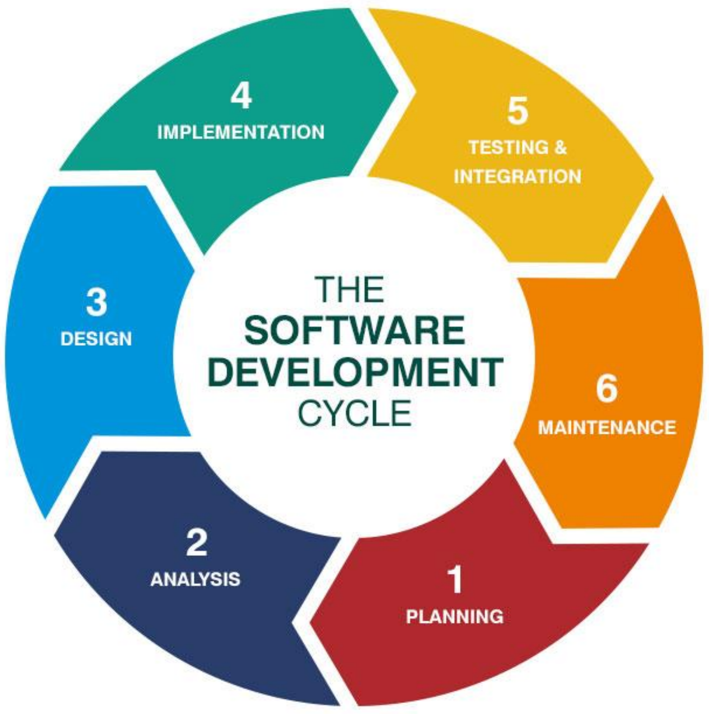


Figure 3.1: Steps Of SDLC

3.2 Project Design and Architecture

In the world of software engineering, multi-tier architecture, often known as n-tier architecture, is a client-server architecture that splits display, application processing, and data management functions deliberately. This separation is conceptual, ensuring a clear difference between the system's many parts. Consider an application that uses middleware to handle data requests, resulting in a multi-tier design that manages communication between a user and a database efficiently. The three-tier design is the most common type of multi-tier architecture, and it is widely used in a variety of software applications.

Three-tier Architecture

The three-tier architecture involves the development and maintenance of user interface, functional process logic or ‘business rule’ and storage for computer data access as separate and autonomous aspects. Traditionally, such modules are constructed into other platforms. This is an architectural model that describes a way of organizing software.

Besides the conventional merits associated with a modular software design, which includes clearly defined interfaces and modular components, the three-tier architecture is specifically crafted to enable independent updating or replacement of any tier. The flexibility of this situation allows for responsiveness towards changes in demand or technological shifts. Fundamentally, the three-tier design facilitates better software modularity, scalability, and flexibility by way of organization.

Presentation Tier

Finally, the presentation tier comprises the front-end aspect of an application. This is the superficial layer that interacts directly with users. The main role of the presentation tier is to provide information to customers while requesting from them. For web applications a host of technologies such as HTML, CSS and Javascript can be used while a host of other technologies like native code for desktop or mobile platform can be used to implement this tier.

Application Tier

The heart of the software is the application part. All the applications' business logic is managed by this processor. This includes tasks like checking of inputs of users, computation of results, and control of data. In most cases, applications in this tier interface with a presentation layer for obtaining data from the user that is required by the programme as well as showcasing the results of the program's processing. It can also connect to the data tier for purposes of retrieving or changing data.

Data Tier

The data tier is in charge of storing and maintaining the data used by the application. This comprises actions including data creation, reading, updating, and deletion. A database management system (DBMS) such as MySQL, PostgreSQL, or Oracle is generally used to implement the data tier.

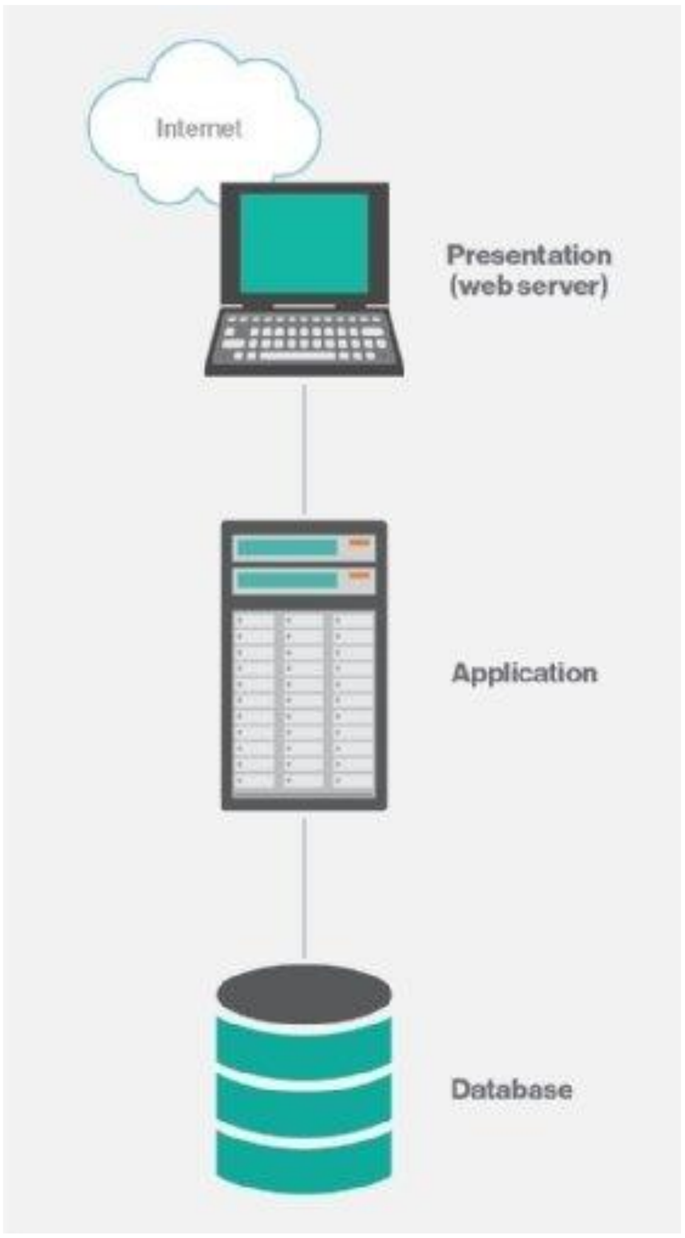
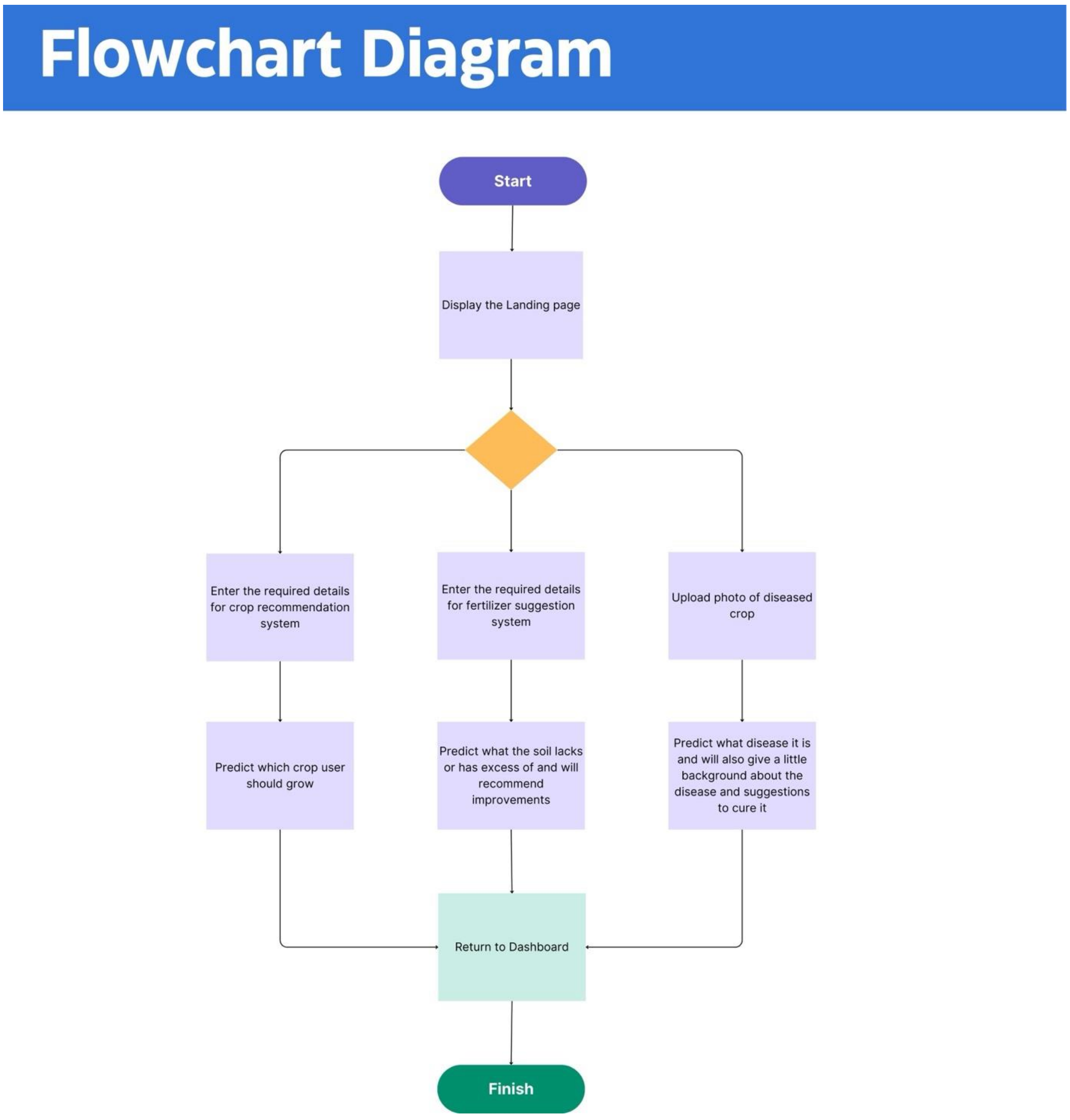


Figure 3.2: 3-tier Architecture

Flow Chart



Flowchart Diagram
Start
Display the Landing page
Enter the required details for crop recommendation system
Enter the required details for fertilizer suggestion system
Upload photo of diseased crop
Predict which crop user should grow
Predict what the soil lacks or has excess of and will recommend improvements
Predict what disease it is and will also give a little background about the disease and suggestions to cure it
Return to Dashboard
Finish

Figure 3.3: Flow chart of the process

3.3 Data Preparation

Data preparation for a Smart Agriculture Crop Disease Detection project involves several critical steps:

1. **Data Collection:** Assemble a varied dataset that includes pictures of crops in both healthy and diseased states. To guarantee dataset diversity, collect photos taken at various crop stages and environmental conditions.
2. **Data Cleaning:** Verify the dataset's consistency and dependability by performing quality checks. Eliminate any images that are duplicates, unnecessary, or of low quality as these could impair the model's performance.
8. **Data Annotation and Labeling:** By hand, annotate and label each image to indicate which diseases are present and in what quantities. In order for machine learning models to effectively learn and distinguish between healthy and diseased crops, proper labelling is essential for supervised learning.
9. **Data Augmentation:** Expand the dataset by incorporating methods to produce different image formats, such as flipping, rotating, scaling, or introducing noise. This procedure improves the robustness and generalisation of the model.
10. **Data Splitting:** Divide the dataset into training, validation, and testing sets to train the machine learning models, validate their performance, and assess their accuracy on unseen data.
11. **Feature Extraction:** To feed into the machine learning models for disease detection, extract pertinent features from the images, such as colour histograms, texture patterns, or other visual characteristics.
12. **Normalization and Preprocessing:** Adjust the data to a standard scale and carry out preprocessing operations such as resizing photos, converting to grayscale, or executing other modifications appropriate for the selected machine learning algorithms.
13. **Balancing the Dataset:** To avoid biassing the model in favour of the majority class, make sure that healthy and diseased samples are fairly represented.
14. **Handling Class Imbalance:** Use strategies like oversampling or undersampling, or make use of specific algorithms built to manage imbalanced datasets, to address class imbalance.

Waterfall Model

A Waterfall Model is one old, classical yet staged software development lifecycle process. It has a rigid sequential passage through specific stages with each stage being completed in turn before proceeding to the other one. The model consists of distinct, non-overlapping stages, namely:

1. **Requirements:** In the first stage of the project, specifications are elaborated through documentation of project prerequisites. Stakeholder's discussions define the scope, functionalities, constraints, and objective of the piece of software.
2. **Design:** The design phase starts subsequent to specifying requirements. Requirements are analyzed and then converted into the structure of the system architecture, software design, and technical specifications. The main focus at this stage is the design and architecture.
3. **Implementation:** It is at this stage that the actual coding or development of the software takes place. The design specification provides information that developers need to use in developing the software based on the set of requirements.
4. **Testing:** The software is thoroughly tested after its development in order to detect errors as well as other faults and defects. It includes such methods as unit testing, integration testing, system testing, and acceptance testing.
5. **Deployment:** The last step on the software development life cycle is when the software passes tests and is ready to be deployed in the production environment for end-users.
6. **Maintenance:** The last phase is about maintenance and support for the newly deployed software. This comprises taking into consideration users' feedback, rectifying errors, enhancing functionalities, and continuous maintenance.

The Waterfall Model has a rigid structure, which contains separate phases that are delivered one after the other. This methodology is well fitted for a project that is characterized with known and steady requirements but with minimum changes expected. On the other hand, this approach becomes rigid when a phase has been completed with no feedback to facilitate change. In spite of this, the Waterfall Model forms a basic principle of software development methodologies, impacting several iterative and adaptive techniques.

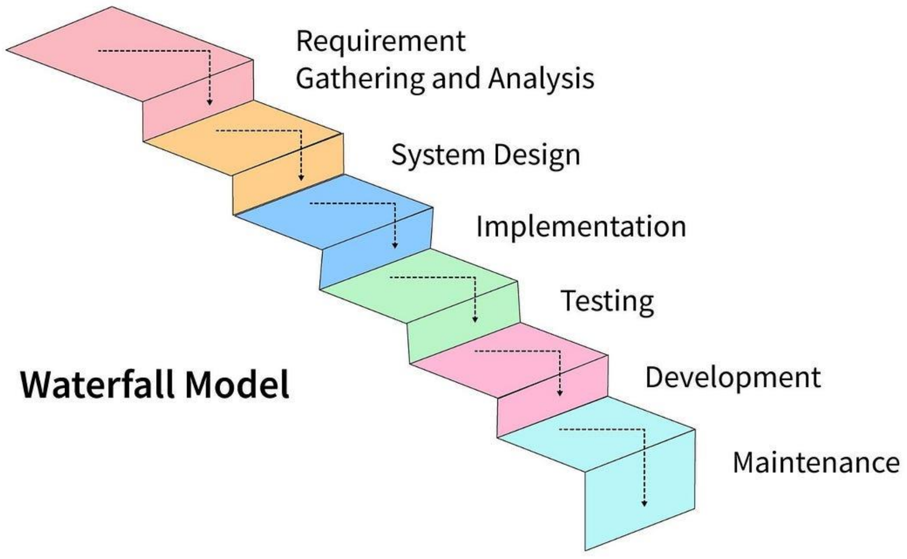


Figure 3.4: Waterfall Model

3.4 Implementation

This project involves a combination of advanced technologies, which enable numerous critical functions. For building an intuitive and responsive frontend, HTML, CSS and Javascript are used in order to provide a smooth user experience. Backend development uses Flask to ensure effective operation on the server side. Image processing involves manipulating and analyzing images of critical importance to disease detection and openCV is a vital part of this. This project utilizes PyTorch for the development of strong models that can effectively diagnose crop diseases. The comprehensive incorporation of varied technologies towards Smart Agriculture Crop Disease Detection makes a single comprehensive solution that is robust, scalable, and more accessible.

Algorithms

Crop Recommendation system

A crop recommendation system is a digital tool designed to assist farmers in optimizing their crop production by providing tailored suggestions based on various factors such as soil nutrient levels, climate conditions, and regional preferences. Let's break down the components and functionality of such a system:

1. **Soil Nutrient Values:** The system requires input of the nutrient values present in the soil. This typically includes the levels of Nitrogen (N), Phosphorus (P), and Potassium (K), often represented as the N-P-K ratio. These values are crucial as they determine the fertility of the soil and influence crop growth and yield. Farmers can obtain these values through soil testing kits or laboratory analysis.
2. **State and City Information:** Alongside the soil nutrients, it is also imperative to know where exactly in the country is the farmer located, at least the state and city are the basic requirements needed for the system to run seamlessly. This factor is pertinent because climatic conditions like temperature, humidity, and rainfall get affected to a large extent depending on the specific location of region. In this case, the system bases the crop advice on the historical data, that is, the same as the current local climate.
3. **Weather API Integration:** The system interfaced with a weather API which is used for either real or historical weather data retrieval for input location. These are the type of data that can be collected as temperature humidity, precipitation, and other climatic factors. Through examining the changes of weather, the system is able to propose the crops that suit for the present situation, making the growth process ideal while the risk of crop failure due to bad weather remains low.
4. **Crop Database:** The system maintains a comprehensive database of crops along with their specific requirements and characteristics. This includes information such as preferred soil type, ideal temperature range, water requirements, and nutrient needs. Based on the input soil nutrient values, location, and weather data, the system matches the requirements of various crops with the available conditions to generate personalized crop recommendations for the farmer.
5. **Recommendation Algorithm:** Behind the scenes, the system employs sophisticated algorithms to analyze the input data and generate accurate recommendations. This may involve machine learning techniques that learn from historical data to improve the accuracy of predictions over time. The algorithm takes into account not only the current soil and weather conditions but also factors like crop rotation, pest and disease resistance, and market demand to offer well-rounded recommendations.

Fertilizer Suggestion system

Fertilizer suggestions based on soil nutrient analysis and crop requirements, such systems empower farmers to make informed decisions that enhance soil health, optimize crop productivity, and promote sustainable agricultural practices. Let's break down the components and functionality of such a system:

1. **Input Parameters:** The system requires two main inputs: the nutrient contents of the soil and the type of crop the farmer intends to grow. Soil nutrient content can be obtained through soil testing kits or laboratory analysis, and it typically includes levels of Nitrogen (N), Phosphorus (P), Potassium (K), and sometimes secondary nutrients like Calcium (Ca), Magnesium (Mg), and Sulphur (S). The crop selection is crucial as different crops have varying nutrient requirements at different growth stages.
2. **Nutrient Analysis:** Once the soil nutrient contents and crop type are provided, the system analyzes the data to determine the existing nutrient levels in the soil. It compares these levels with the specific nutrient requirements of the chosen crop at various growth stages. Based on this analysis, the system identifies any deficiencies or excesses of key nutrients in the soil that could potentially affect crop growth and yield.
3. **Fertilizer Recommendations:** After identifying nutrient deficiencies or excesses, the system generates personalized fertilizer recommendations to address the soil's specific needs and optimize crop growth. For example, if the soil lacks Nitrogen, the system may suggest fertilizers high in Nitrogen such as urea or ammonium nitrate. If there's an excess of Phosphorus, it may recommend fertilizers with lower Phosphorus content or strategies to reduce Phosphorus runoff.
4. **Nutrient Balancing:** The system aims to achieve nutrient balance in the soil by providing recommendations that complement existing nutrient levels and meet the crop's requirements. It considers factors such as the availability of different fertilizer formulations, application methods, and timing to ensure efficient nutrient uptake by the crops while minimizing environmental impact and resource wastage.
5. **Feedback and Monitoring:** To ensure the effectiveness of the fertilizer recommendations, the system may incorporate feedback mechanisms and monitoring tools. Farmers can provide feedback on the performance of recommended fertilizers, crop health, and yield outcomes. Additionally, the system may offer features for ongoing soil testing and monitoring to track changes in nutrient levels over time and adjust

fertilizer applications accordingly, thereby enabling continuous optimization of soil fertility management practices.

Disease Detection system

A Disease Detection System for plant leaves is a digital tool designed to assist farmers and gardeners in identifying diseases affecting their crops and providing recommendations for prevention or treatment. Here's an expanded explanation of how such a system typically operates:

1. **Image Submission and Analysis:** Users upload an image of a plant leaf to initiate the analysis process. The system employs image recognition algorithms to scrutinize the leaf's characteristics, identifying patterns or anomalies that may indicate disease presence.
2. **Crop Identification and Health Assessment:** After analyzing the leaf image, the system determines the type of crop depicted and evaluates its health status. It distinguishes between healthy and diseased leaves, providing instant feedback on the leaf's condition.
3. **Disease Diagnosis and Cause Identification:** If the leaf is flagged as diseased, the system delves deeper to pinpoint the specific ailment affecting the crop. Drawing from an extensive database of known diseases, it identifies the root cause of the issue, whether it be fungal, bacterial, viral, or related to pests.
4. **Tailored Recommendations for Prevention and Treatment:** Leveraging its diagnostic findings, the system offers personalized recommendations tailored to address the identified disease. These suggestions encompass preventive measures to minimize future occurrences, as well as treatment options to mitigate the current ailment's impact on crop health.
5. **Supported Crops and Future Expansion:** The system currently supports a predefined set of crops for disease detection and diagnosis. However, it remains flexible for future expansion, with plans to incorporate additional crop types and diseases over time. This iterative approach ensures that the system continuously evolves to meet the diverse needs of users in the agricultural community.

Technologies Used

Flask

Flask is a lightweight and flexible approach to develop web applications written in python that are mostly based on their minimum but competent backends. It uses micro-framework approach, offering fundamental elements and parts that make up web application with flexibility to go for other libraries and extension depending on the project demands. Flask is a web application framework built on top of Werkzeug WSGI Toolkit and Jinja2 templating engine making it possible to create apps that support request handlers, route requests, and render html documents. The simplicity, easy learning curve as well as ability to develop RESTful API in flask makes flask one of the most preferred back-end development framework for light weight heavy duty type of projects. Flask can effectively perform server-side operations, manage routing and interact with other front-end components in the Smart Agriculture Crop Disease Detection project making a connection between them easy.

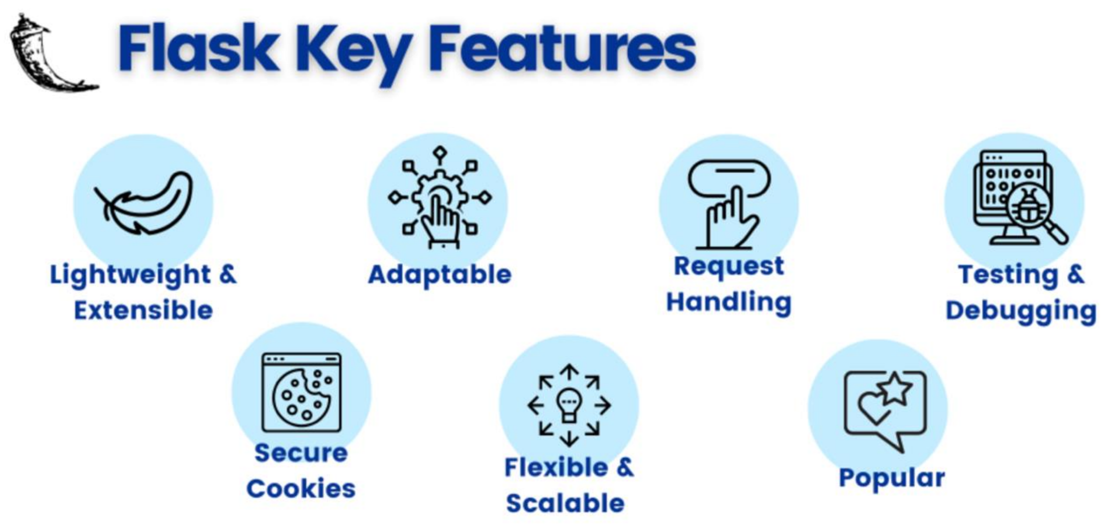


Figure 3.5: Key Features of Flask

OpenCV

It is a free and open platform for real time computer vision & machine learning (especially image & video processing in practice). It presents an end-to-end set of functions and algorithms for dealing with multiple computer vision tasks including; object detection, face recognition, image segmenting, features extraction etc. Open CV is programmed in C++ with a Python wrapper and a Java binding; therefore, it can be used by various types of developers and studies scholars. This includes image manipulation, transform, filter and geometric operations prefunctions as well as machine learning for classification, clustering and regression. Its flexibility, reliability, and ample documentation have made OpenCV a widely used framework in both research and practice areas such as robotic, medical, security, enhanced reality, and self-driving cars. OpenCV is important within the smart agriculture crop disease detection projects as it assists in analyzing crops diseases, and the categorization of a particular disease from captured images in the fields.

PyTorch

PyTorch is an open-source machine learning library created primarily by Facebook's AI Research Lab (FAIR). It is extensively employed in many machines learning applications, including as computer vision, reinforcement learning, and natural language processing. Particularly well-suited for deep learning and neural network research, PyTorch offers tensor computation (akin to NumPy) with robust GPU acceleration, automatic differentiation for configuring and training neural networks, and a dynamic computational graph.

Compared to comparable frameworks like TensorFlow, PyTorch's dynamic computational network offers greater flexibility in model development and debugging, making it one of its unique advantages. Because PyTorch allows you to define and modify the computational graph as you run code, experimenting with different architectures and concepts is made easier.

Additionally, a robust ecosystem of libraries and tools has been developed around PyTorch. Examples of these include torchtext for natural language processing, torchaudio for audio processing, and torchvision for computer vision tasks. Furthermore, PyTorch facilitates deployment to production systems by means of TorchServe and related technologies.

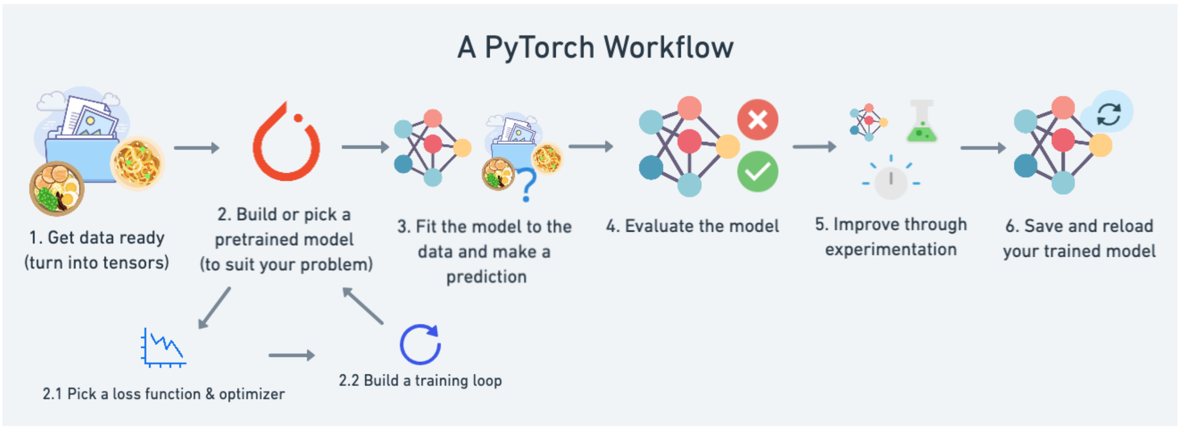

A PyTorch Workflow
1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2 Build a training loop

Figure 3.6: PyTorch Workflow

Numpy

NumPy, short for Numerical Python, is a fundamental library in the Python ecosystem for numerical computing. It provides powerful tools for working with arrays, matrices, and mathematical functions, making it essential for tasks ranging from simple data manipulation to complex scientific computations. NumPy's primary data structure is the ndarray (n-dimensional array), which allows for efficient storage and manipulation of large datasets. Its extensive collection of built-in functions enables fast and vectorized operations, enhancing performance compared to traditional Python lists. NumPy is also seamlessly integrated with other libraries such as SciPy, pandas, and Matplotlib, forming the backbone of the Python scientific computing stack. Its versatility, speed, and ease of use have made NumPy indispensable for scientists, engineers, and data analysts tackling a diverse array of computational challenges in fields like physics, biology, finance, and machine learning.

Pandas

Pandas is a Python library widely used for data manipulation and analysis. Built on top of NumPy, it offers data structures like Series and DataFrame that are intuitive and powerful, enabling users to work efficiently with structured data. Pandas excels in tasks such as data cleaning, transformation, and exploration, providing a plethora of functions and methods for indexing, filtering, grouping, and aggregating data. Its integration with other libraries like Matplotlib and Seaborn allows for seamless visualization of data, aiding in the interpretation and communication of insights. Whether handling small or large datasets, Pandas offers high performance and flexibility, making it a cornerstone tool for data scientists, analysts, and developers in various domains, including finance, healthcare, marketing, and beyond.

Matplotlib

Matplotlib is a comprehensive library in Python for creating static, interactive, and publication-quality visualizations. With its flexible and customizable interface, it enables users to generate a wide range of plots and charts, including line plots, scatter plots, histograms, bar charts, and more. Matplotlib's object-oriented approach allows for fine-grained control over every aspect of a plot, from the axes and labels to colors and styles, ensuring that users can create visuals that meet their specific needs and preferences. Whether used for exploratory data analysis, presentation-ready graphics, or embedding in web applications, Matplotlib's versatility and extensive documentation make it a go-to tool for data scientists, researchers, engineers, and educators seeking to convey insights from their data effectively.

Scikit-learn

Scikit-learn, often abbreviated as sklearn, is a widely-used machine learning library in Python, renowned for its simplicity, efficiency, and versatility. Built on top of other Python libraries such as NumPy, SciPy, and matplotlib, scikit-learn provides a unified interface for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and model selection. It offers a rich collection of algorithms and tools, ranging from classical methods like support vector machines and random forests to state-of-the-art techniques such as deep learning and gradient boosting. With its consistent API and extensive documentation, scikit-learn simplifies the process of building and deploying machine learning models, making it accessible to users at all skill levels, from beginners to seasoned practitioners. Its emphasis on reproducibility, model evaluation, and integration with other Python libraries makes scikit-learn an indispensable tool for researchers, data scientists, and developers working on a wide range of real-world problems.

HTML5

HTML5, or Hypertext Markup Language revision 5, is a client-side markup language that provides a platform-independent method of structuring and presenting material on the World Wide Web. Various HTML5 tags, such as those used for headings, paragraphs, divisions, and anchors, make it easier to organize data. Notably, HTML5 permits the embedding of scripting languages such as JavaScript and CSS, allowing users to create and structure information using elements, tags, and attributes.

CSS3

Cascading Style Sheets (CSS) is a language that defines the visual presentation features of web pages, including elements such as colors, layout, and fonts to improve the overall aesthetic appeal. CSS, which operates independently of HTML, can be combined with any XML-based markup language. There are three ways to include CSS in HTML files: inline CSS, internal CSS, and external CSS.

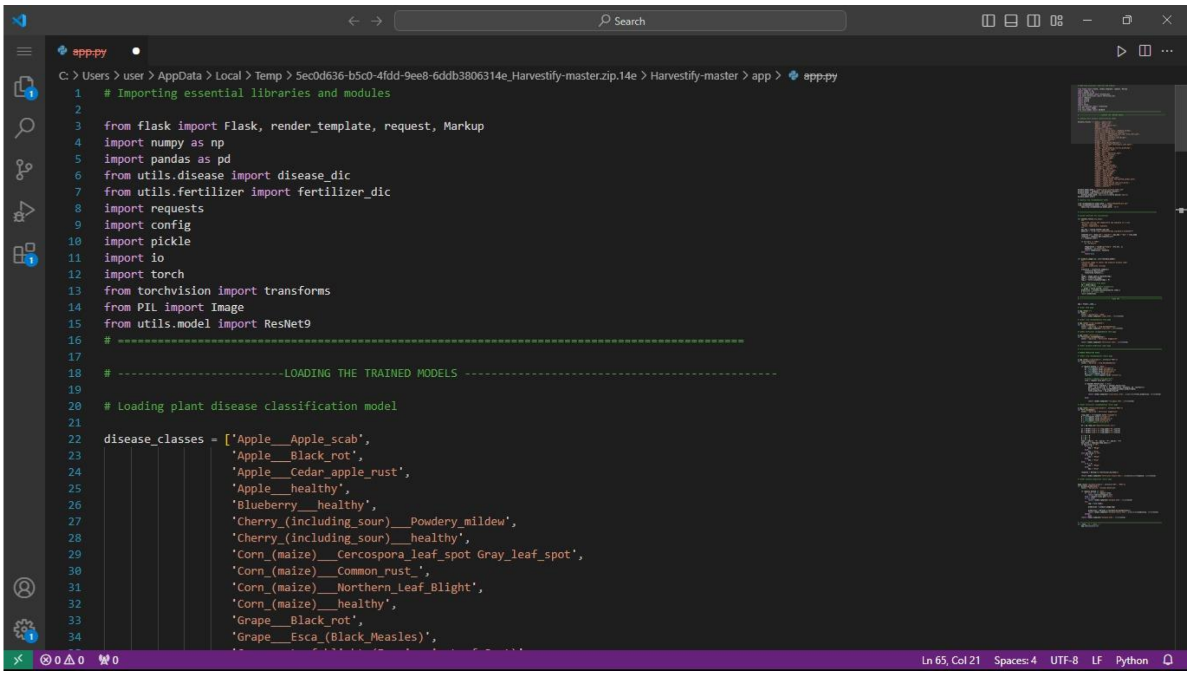
JavaScript

JavaScript: JavaScript, a cross-platform, object-oriented language, is useful for adding interactive functionality to web sites, such as elaborate animations, clickable buttons, and pop-up menus. Advanced JavaScript implementations, such as Node.js, extend website functionality without the requirement for file downloads. When used in a host environment, such as a web browser, JavaScript can be linked to native objects, giving the system control over them.

Bootstrap

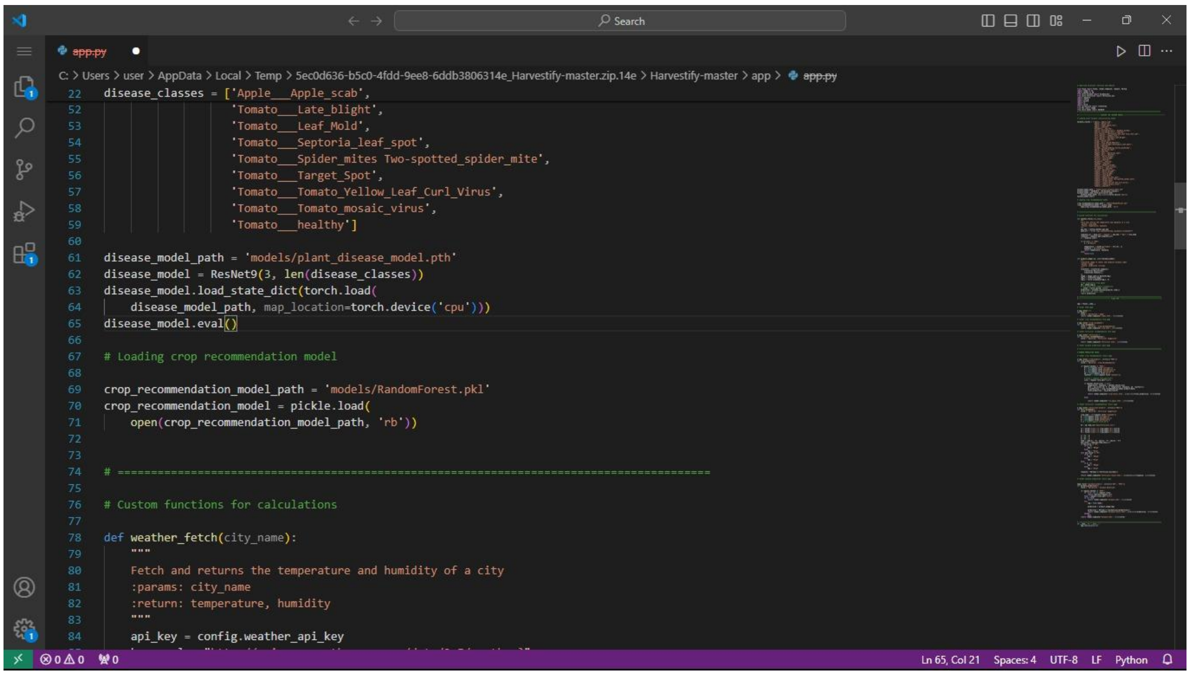
Bootstrap is a powerful front-end framework for developing responsive and mobile-first websites and web applications. Developed by Twitter, it provides a set of pre-designed HTML, CSS, and JavaScript components, as well as customizable themes, that streamline the process of building modern, visually appealing interfaces. Bootstrap's grid system allows for easy layout structuring, ensuring consistency across different screen sizes and devices. With its extensive documentation and large community support, Bootstrap simplifies web development tasks, accelerates prototyping, and facilitates the creation of intuitive user experiences. Whether you're a beginner or an experienced developer, Bootstrap empowers you to create professional-looking websites efficiently, making it a go-to choose for many web projects.

Code Snippets



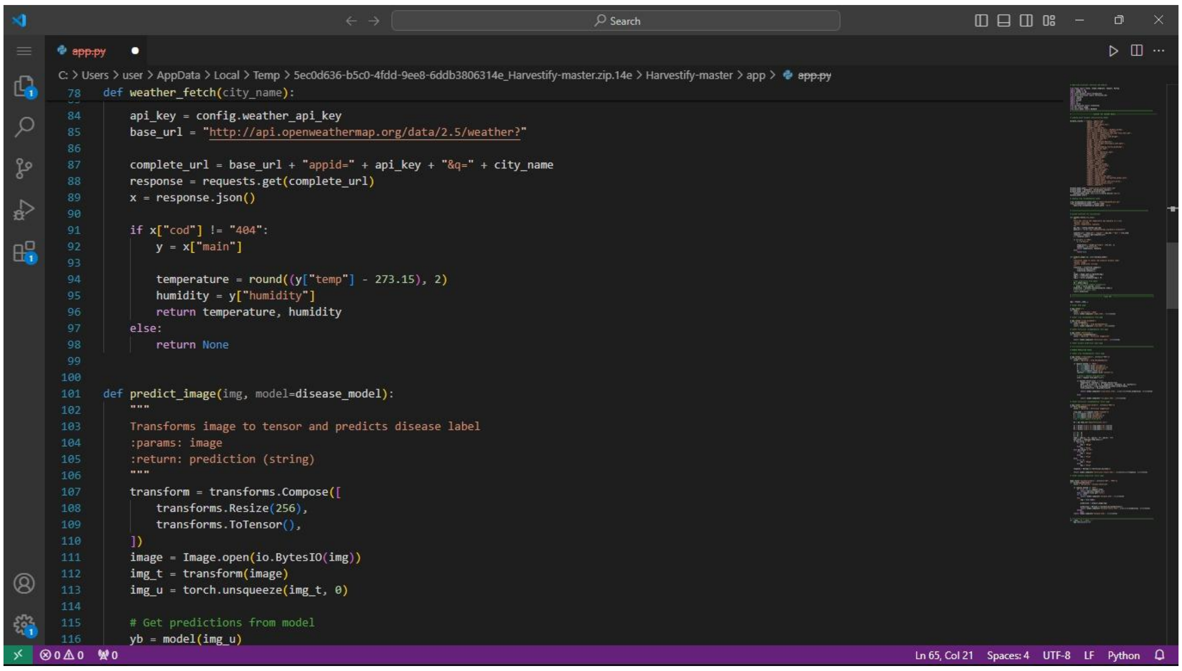
app.py
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py
Ln 65, Col 21 Spaces: 4 UTF-8 LF Python

```python
# Importing essential libraries and modules

from flask import Flask, render_template, request, Markup
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
# ===========================================================================================

# ------------------------LOADING THE TRAINED MODELS -----------------------------------------------

# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple___Cedar_apple_rust',
                   'Apple___healthy',
                   'Blueberry___healthy',
                   'Cherry_(including_sour)___Powdery_mildew',
                   'Cherry_(including_sour)___healthy',
                   'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
                   'Corn_(maize)___Common_rust_',
                   'Corn_(maize)___Northern_Leaf_Blight',
                   'Corn_(maize)___healthy',
                   'Grape___Black_rot',
                   'Grape___Esca_(Black_Measles)',
```



app.py
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py
Ln 65, Col 21 Spaces: 4 UTF-8 LF Python

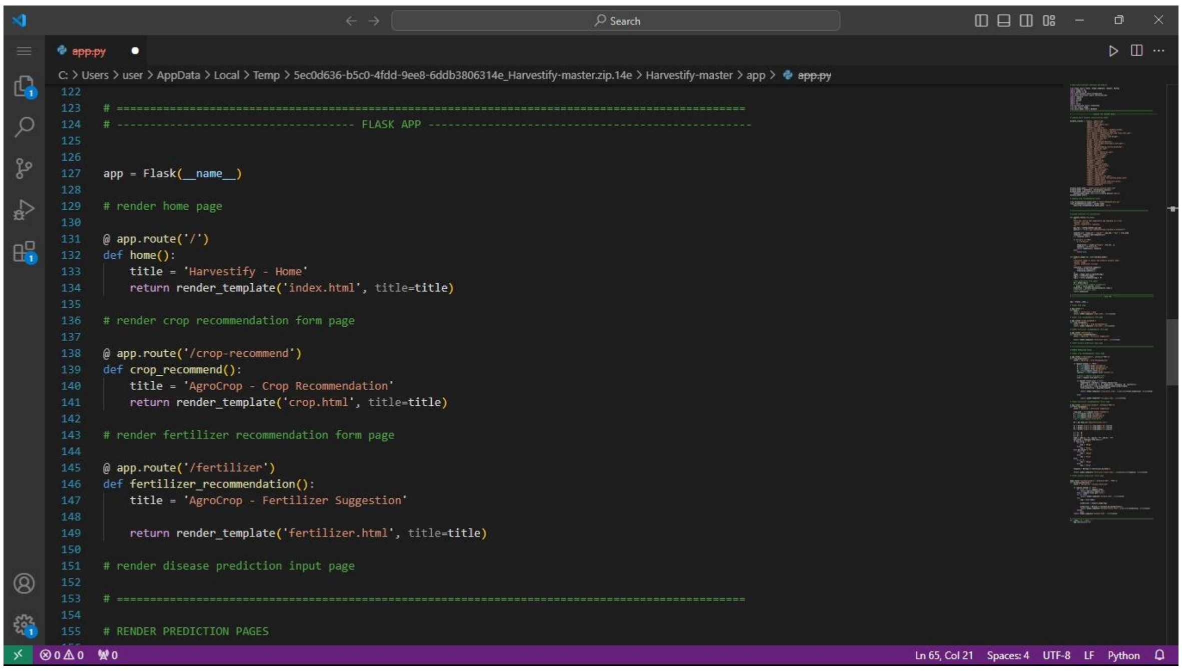
```python
disease_classes = ['Apple___Apple_scab',
                   'Tomato___Late_blight',
                   'Tomato___Leaf_Mold',
                   'Tomato___Septoria_leaf_spot',
                   'Tomato___Spider_mites Two-spotted_spider_mite',
                   'Tomato___Target_Spot',
                   'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
                   'Tomato___Tomato_mosaic_virus',
                   'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load(
    disease_model_path, map_location=torch.device('cpu')))
disease_model.eval()

# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load(
    open(crop_recommendation_model_path, 'rb'))


# ========================================================================================

# Custom functions for calculations

def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
    :params: city_name
    :return: temperature, humidity
    """
    api_key = config.weather_api_key
```



C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py

```python
def weather_fetch(city_name):

    api_key = config.weather_api_key
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    complete_url = base_url + "appid=" + api_key + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]

        temperature = round((y["temp"] - 273.15), 2)
        humidity = y["humidity"]
        return temperature, humidity
    else:
        return None


def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
    """
    transform = transforms.Compose([
        transforms.Resize(256),
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

    # Get predictions from model
    yb = model(img_u)
```

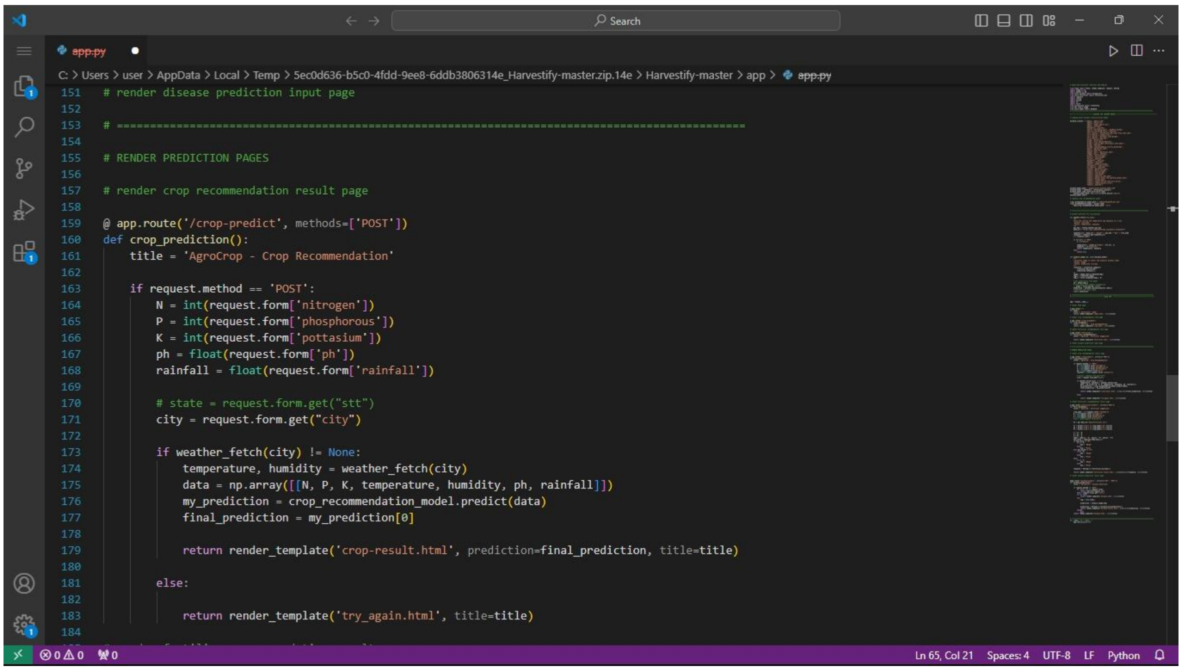
Ln 65, Col 21   Spaces: 4   UTF-8   LF   Python



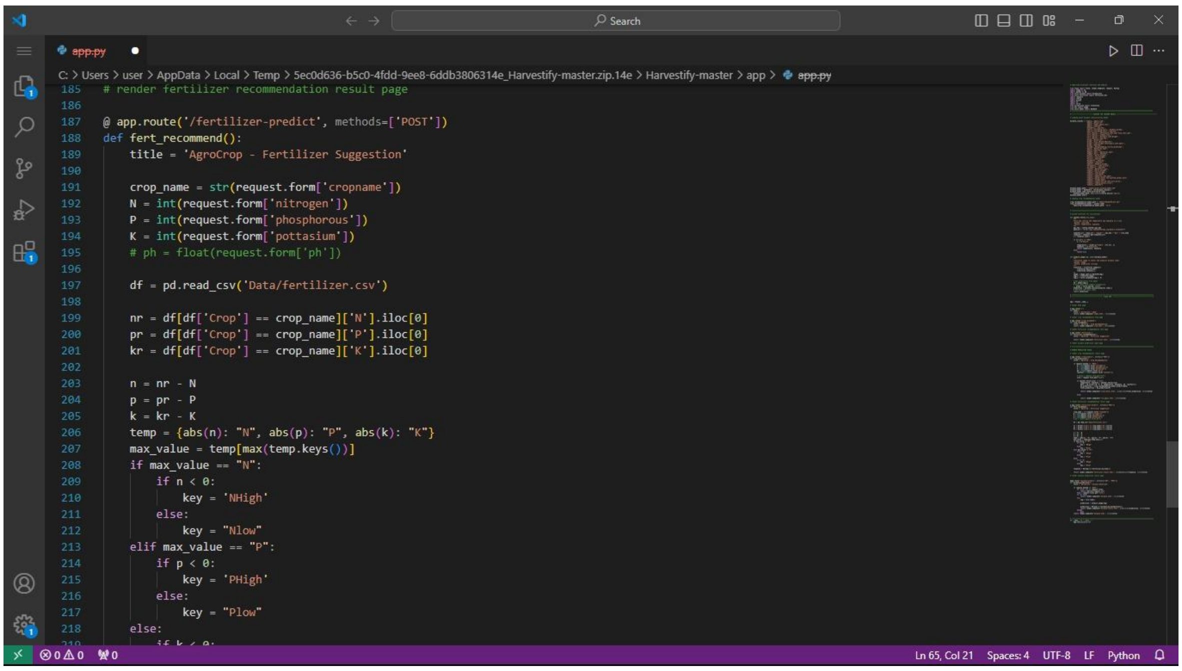
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py

```python

# ===============================================================================================
# ------------------------------------ FLASK APP ------------------------------------------------


app = Flask(__name__)

# render home page

@ app.route('/')
def home():
    title = 'Harvestify - Home'
    return render_template('index.html', title=title)

# render crop recommendation form page

@ app.route('/crop-recommend')
def crop_recommend():
    title = 'AgroCrop - Crop Recommendation'
    return render_template('crop.html', title=title)

# render fertilizer recommendation form page

@ app.route('/fertilizer')
def fertilizer_recommendation():
    title = 'AgroCrop - Fertilizer Suggestion'

    return render_template('fertilizer.html', title=title)

# render disease prediction input page

# ===============================================================================================

# RENDER PREDICTION PAGES
```

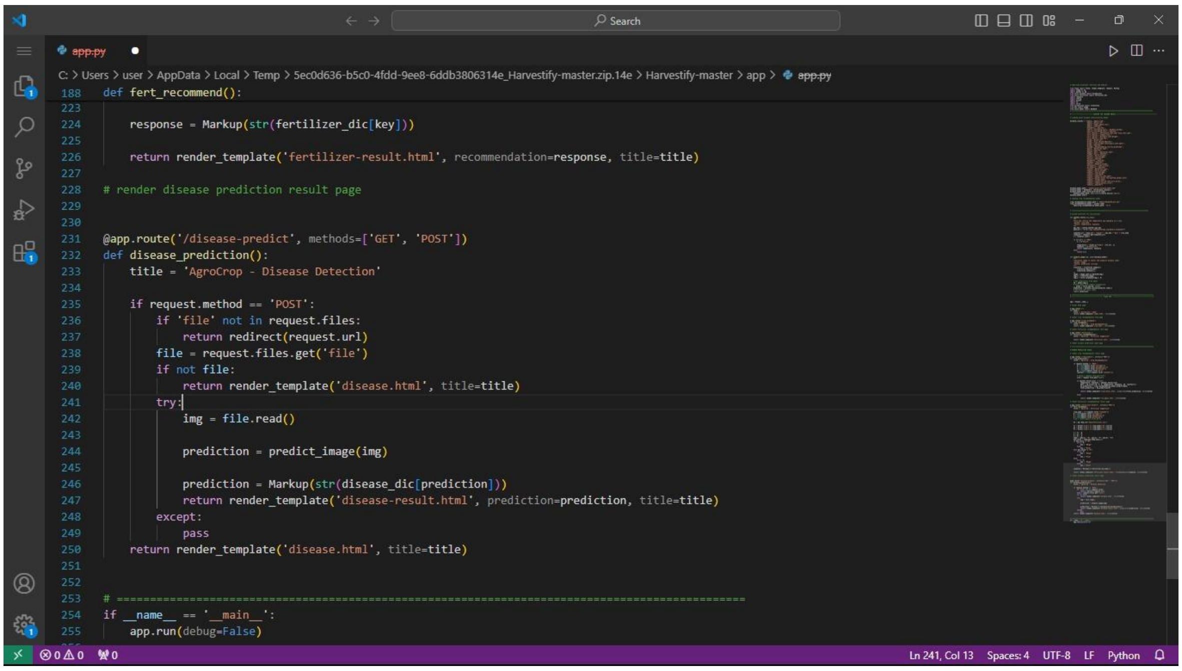
Ln 65, Col 21   Spaces: 4   UTF-8   LF   Python


app.py
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py
Search
Ln 65, Col 21 Spaces: 4 UTF-8 LF Python

```python
# render disease prediction input page

# ===============================================================================================

# RENDER PREDICTION PAGES

# render crop recommendation result page

@ app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'AgroCrop - Crop Recommendation'

    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
        K = int(request.form['pottasium'])
        ph = float(request.form['ph'])
        rainfall = float(request.form['rainfall'])

        # state = request.form.get("stt")
        city = request.form.get("city")

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)
            data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
            my_prediction = crop_recommendation_model.predict(data)
            final_prediction = my_prediction[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)

        else:

            return render_template('try_again.html', title=title)

```


app.py
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py
Search
Ln 65, Col 21 Spaces: 4 UTF-8 LF Python

```python
# render fertilizer recommendation result page

@ app.route('/fertilizer-predict', methods=['POST'])
def fert_recommend():
    title = 'AgroCrop - Fertilizer Suggestion'

    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])

    df = pd.read_csv('Data/fertilizer.csv')

    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
    pr = df[df['Crop'] == crop_name]['P'].iloc[0]
    kr = df[df['Crop'] == crop_name]['K'].iloc[0]

    n = nr - N
    p = pr - P
    k = kr - K
    temp = {abs(n): "N", abs(p): "P", abs(k): "K"}
    max_value = temp[max(temp.keys())]
    if max_value == "N":
        if n < 0:
            key = 'NHigh'
        else:
            key = "Nlow"
    elif max_value == "P":
        if p < 0:
            key = 'PHigh'
        else:
            key = "Plow"
    else:
```


app.py
C: > Users > user > AppData > Local > Temp > 5ec0d636-b5c0-4fdd-9ee8-6ddb3806314e_Harvestify-master.zip.14e > Harvestify-master > app > app.py
Ln 241, Col 13 Spaces: 4 UTF-8 LF Python

```python
def fert_recommend():

    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response, title=title)

# render disease prediction result page


@app.route('/disease-predict', methods=['GET', 'POST'])
def disease_prediction():
    title = 'AgroCrop - Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)
        file = request.files.get('file')
        if not file:
            return render_template('disease.html', title=title)
        try:
            img = file.read()

            prediction = predict_image(img)

            prediction = Markup(str(disease_dic[prediction]))
            return render_template('disease-result.html', prediction=prediction, title=title)
        except:
            pass
    return render_template('disease.html', title=title)


# ===============================================================================================
if __name__ == '__main__':
    app.run(debug=False)
```

3.5 Key Challenges

Some critical issues that are faced by Smart Agriculture Crop Disease detection project during implementation include the following. The first and most crucial obstacle has been obtaining a complete and representative dataset of many good pictures of various crop illnesses under distinct environment conditions. The development of machine learning-based models that can correctly diagnose different types of crop diseases in various contexts is still one of the fundamental problems. Another major challenge is real-time disease detection in dynamic agricultural settings that must be accurate even under changing environmental circumstances. Furthermore, combining different technologies and processes alongside building a simple interface for farmers and experts is further complicated. Furthermore, challenges include data collection, preprocessing, scalability, and ensuring reliable performance during different times of day that affect image quality and presentation of diseases under varying environmental conditions. To address these challenges necessitates an interdisciplinary practice of machine learning, agriculture, programming, and familiarity with specific barriers in terms of agricultural practice and technical integration.

Collecting, annotating, and verifying accuracy and relevancy is the main complication that the project confronts with. In the preprocessing phase, challenges crop up regarding cleaning and transforming the dataset ready for model training with integrity intact. There is also a requirement for computational resources, particularly for deep-learning based models that are associated with infrastructure challenges such as the availability of equipment necessary, scalability, and feasibility of deployment under resource limitations at the farm level. A significant complication also stems from the environmental variability such as different types of illumination, weather changes in the field, and other specific elements that could significantly affect the quality of the photograph and signs of disease. Modeling is very important in such unstable environments. However, building a successful algorithm should be sturdy enough to account for this. Finally, the ability for the structure to accommodate additional information and new improvements as well maintaining consistency of its functionality during future stages becomes an open case question in terms of constructing and launching of the project.

Chapter 4: Testing

4.1 Testing Strategy

The testing strategy for Smart Agriculture Crop Disease Detection Project involves several important issues that are needed to ensure that the system is reliable, accurate and effective. Unit testing is based on testing of separate units, which are evaluated individually so that proper functioning of all parts can be ascertained and actual results confirmed. Subsequently, integration testing checks the interoperability and communication between separate sub-systems or segments within an integrated system by validating that information exchanges are efficient across different pieces of a holistic unit.
Secondly, validation testing is vital as it confirms that the system conforms to stipulated standards and users' demands. This entail ascertaining whether the system is able to correctly diagnose and identify crop infections according to certain parameters set. Some of these metrics are response time, processing speed, resource utilization but all this within various conditions in order to ensure that the system will be adequate for scaling. Moreover, accuracy and stress test in machine learning model confirm its capability correctly to diagnose diseases when applied different dataset and situation.

User Acceptance Testing (UAT) tests the usability, functionality, and user interface of a system by involving end users, such as farmers or agricultural specialists. It guarantees that the system satisfies user needs, is easy to use, and offers insightful information in actual agricultural situations. In order to safeguard sensitive data, security testing evaluates possible weaknesses, puts secure data handling procedures into place, and guarantees data privacy and authentication methods.

Unit Testing

The process of evaluating individual units of software to ensure that they perform as expected is known as unit testing. This level of testing is usually done by developers and is frequently automated. Unit testing is an important aspect of the software development lifecycle (SDLC) since it aids in the early detection and correction of issues.

Unit testing is intended to:

- Check that each piece of software works as it should.
- Identify and fix bugs as early as possible in the development process.
- Improve the software's quality.
- Make the software easier to maintain.
- The Advantages of Unit Testing

Unit testing has numerous advantages, including:

- Lower development expenses
- enhanced software quality
- Lower maintenance expenses
- increased developer assurance

Integration Testing

The process of evaluating individual parts of software that have been integrated together is known as integration testing. This level of testing is often carried out by testers and is frequently manual. Integration testing is crucial because it aids in the detection of defects that emerge when software units interact with one another.

Integration testing's goal is to:

- Identify faults that emerge when software components interact with one another. Ascertain that the integrated software fits the overall system requirements.
- Improve the integrated software's quality.
- The Advantages of Integration Testing

Integration testing has numerous advantages, including:

- Lower development expenses
- enhanced software quality
- Lower maintenance expenses
- increased trust in the integrated software

Integration Testing Techniques

Many techniques are available for integration testing, such as:

- Testing from the top down
- Bottom-up research
- Sandwich evaluation

System Testing

System testing means that the process of testing an entire, integrated system. Testers usually do this type of testing that involves physical action. System testing is important as it helps detect errors that emerge when the system combats each other or the environment.

The purpose of system testing is to:

Identify any issues that arise when the system interacts with other systems or the environment. Check to see if the system fulfills the overall system requirements. Improve the system's quality.

The Advantages of System Testing:

- Lower development expenses
- enhanced software quality
- Lower maintenance expenses
- Improved system confidence System Testing Techniques

There are several system testing methodologies that can be utilized, including:

- Testing in the dark
- White-box evaluation
- Performance Testing

Acceptance Testing

Acceptance testing is the process of determining whether a system is ready for delivery. Typically, the customer or end-user does this level of testing. Acceptance testing is vital since it ensures that the system satisfies the needs and expectations of the client.

Acceptance testing is performed to:

- Determine whether the system satisfies the needs and expectations of the customer.
- Check to see if the system fulfills the overall system requirements.
- Improve the system's quality.

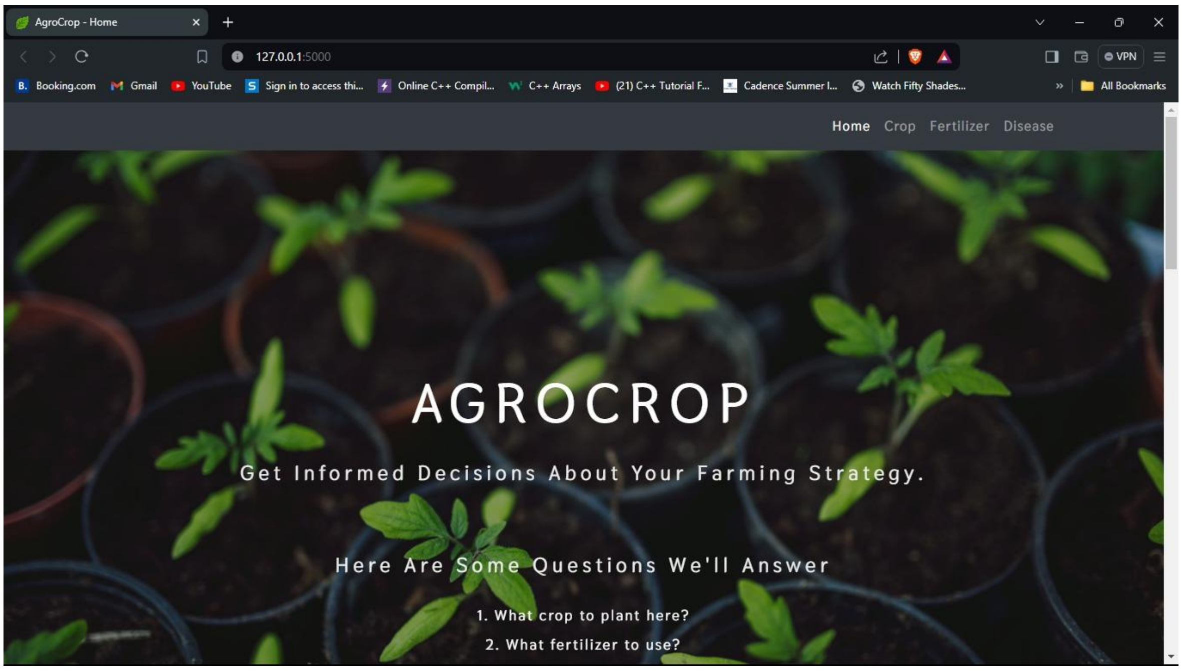
Acceptance testing has numerous advantages, including:

- Customer satisfaction has increased.
- Lower development expenses
- enhanced software quality
- Lower maintenance expenses
- Acceptance Techniques for Testing

There are various acceptability testing methodologies that can be utilized, including:

- User acceptance testing (UAT)
- Beta testing
- Pilot testing

Chapter 5: Results



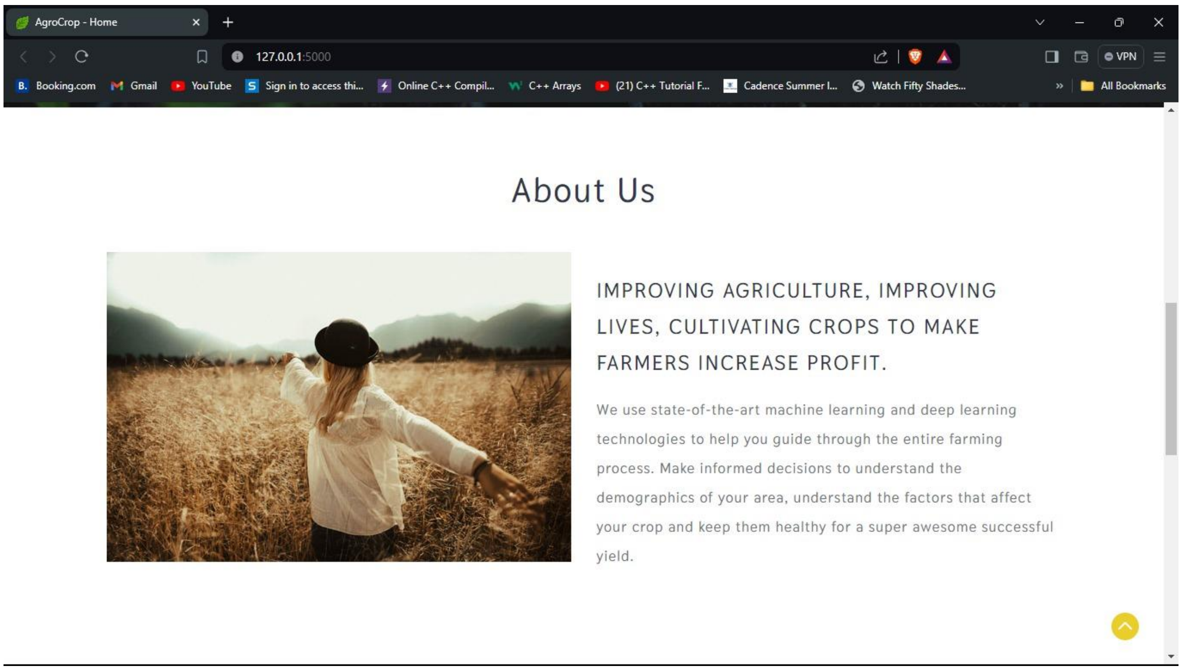
AGROCROP

Get Informed Decisions About Your Farming Strategy.

Here Are Some Questions We'll Answer

1. What crop to plant here?
2. What fertilizer to use?

Figure 5.1: Landing page of website -1

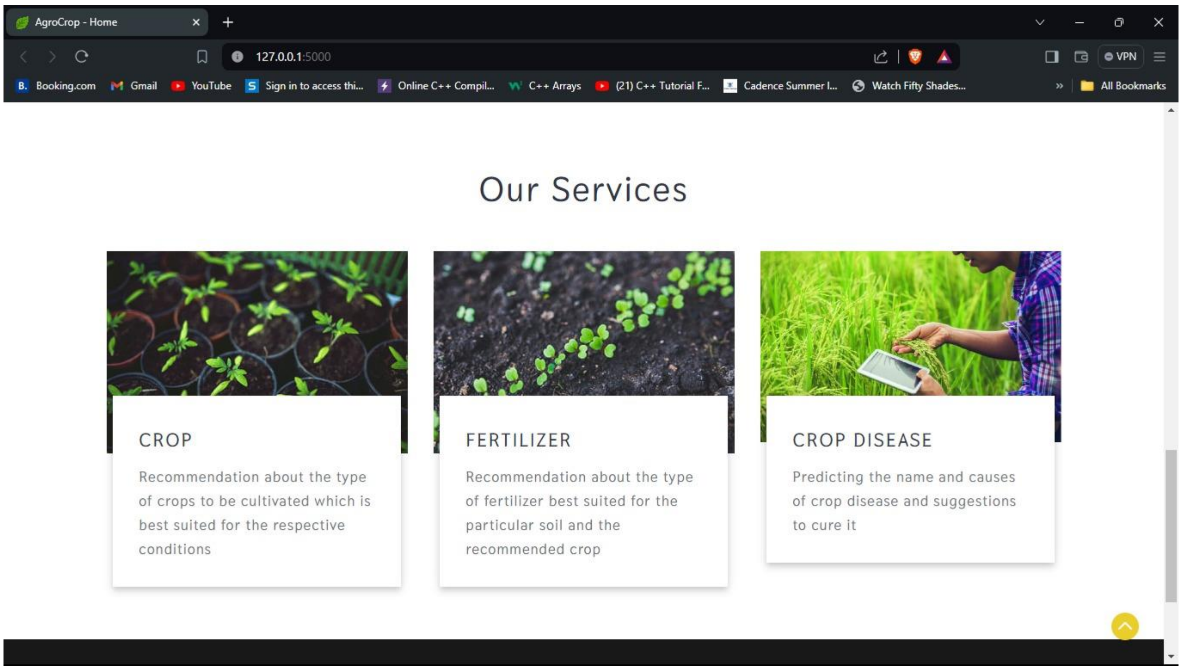


About Us

IMPROVING AGRICULTURE, IMPROVING LIVES, CULTIVATING CROPS TO MAKE FARMERS INCREASE PROFIT.

We use state-of-the-art machine learning and deep learning technologies to help you guide through the entire farming process. Make informed decisions to understand the demographics of your area, understand the factors that affect your crop and keep them healthy for a super awesome successful yield.

Figure 5.2: Landing page of website -2



Our Services

CROP

Recommendation about the type of crops to be cultivated which is best suited for the respective conditions

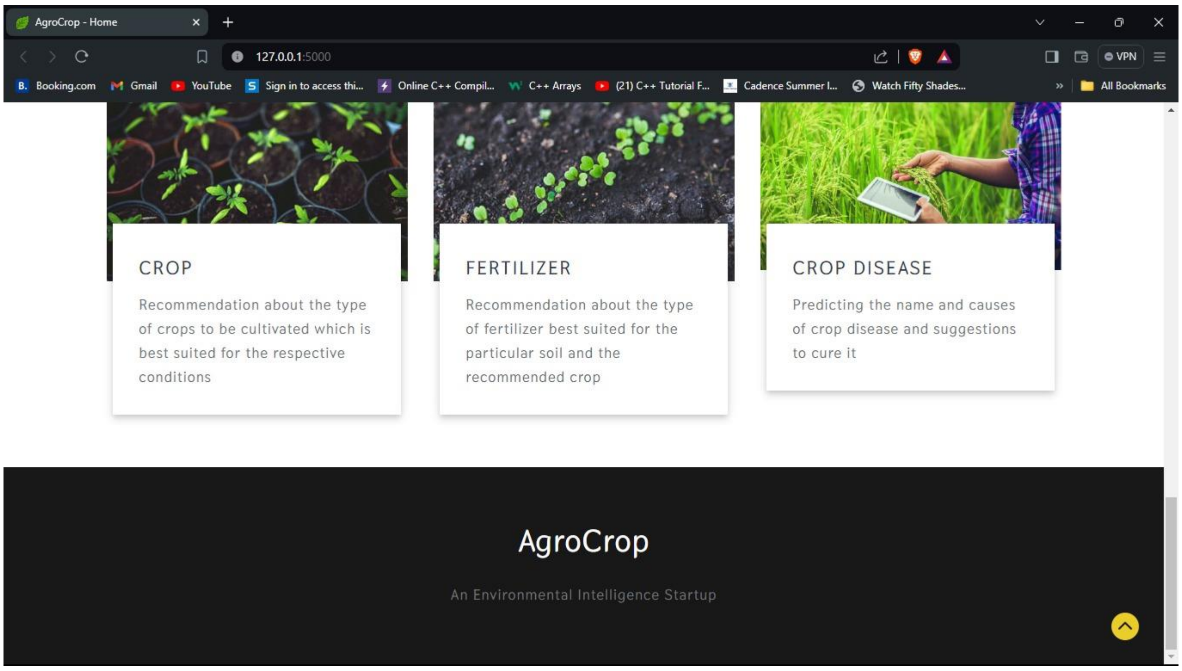
FERTILIZER

Recommendation about the type of fertilizer best suited for the particular soil and the recommended crop

CROP DISEASE

Predicting the name and causes of crop disease and suggestions to cure it

Figure 5.3: Landing page of website -3



CROP

Recommendation about the type of crops to be cultivated which is best suited for the respective conditions

FERTILIZER

Recommendation about the type of fertilizer best suited for the particular soil and the recommended crop

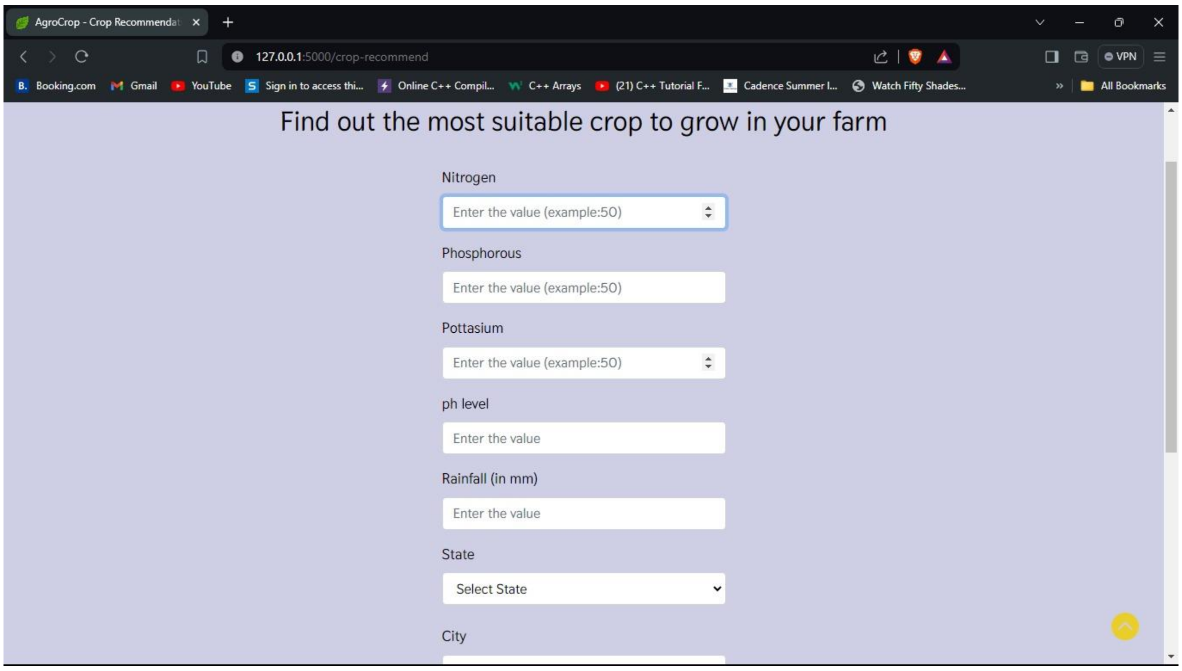
CROP DISEASE

Predicting the name and causes of crop disease and suggestions to cure it

AgroCrop

An Environmental Intelligence Startup

Figure 5.4: Landing page of website -4


Find out the most suitable crop to grow in your farm

Nitrogen
Enter the value (example:50)

Phosphorous
Enter the value (example:50)

Pottasium
Enter the value (example:50)

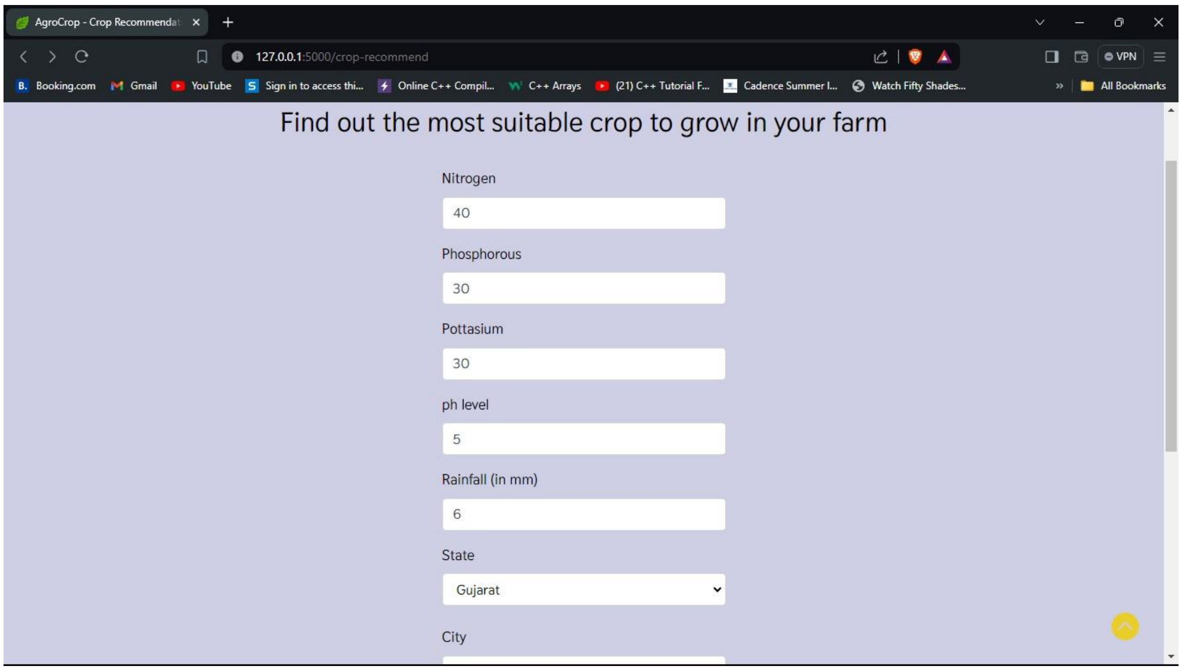
ph level
Enter the value

Rainfall (in mm)
Enter the value

State
Select State

City

Figure 5.5: Crop recommendation system


Find out the most suitable crop to grow in your farm

Nitrogen
40

Phosphorous
30

Pottasium
30

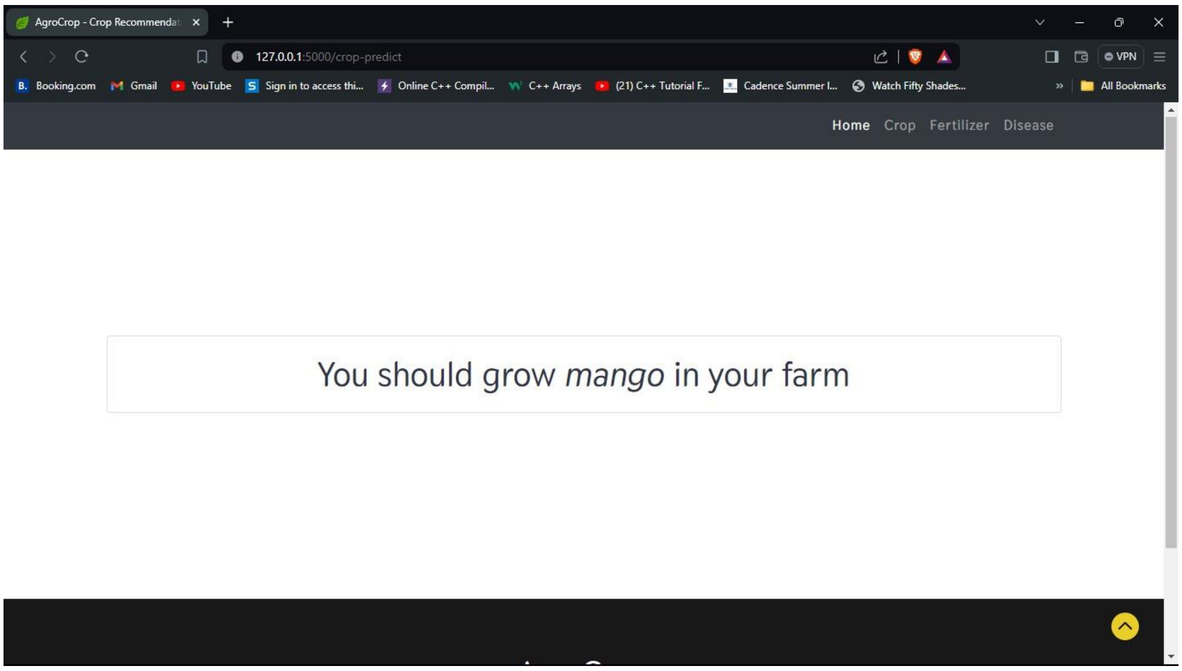
ph level
5

Rainfall (in mm)
6

State
Gujarat

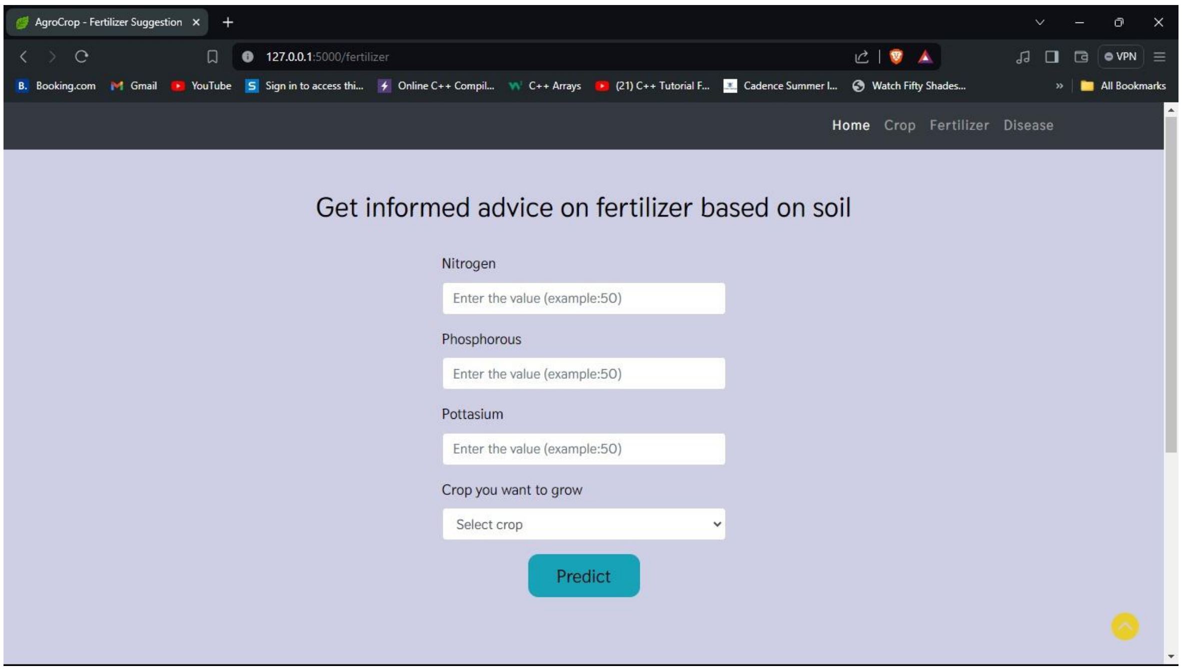
City

Figure 5.6: Input sample values in crop recommendation system



You should grow *mango* in your farm

Figure 5.7: Result of crop recommendation system



Get informed advice on fertilizer based on soil

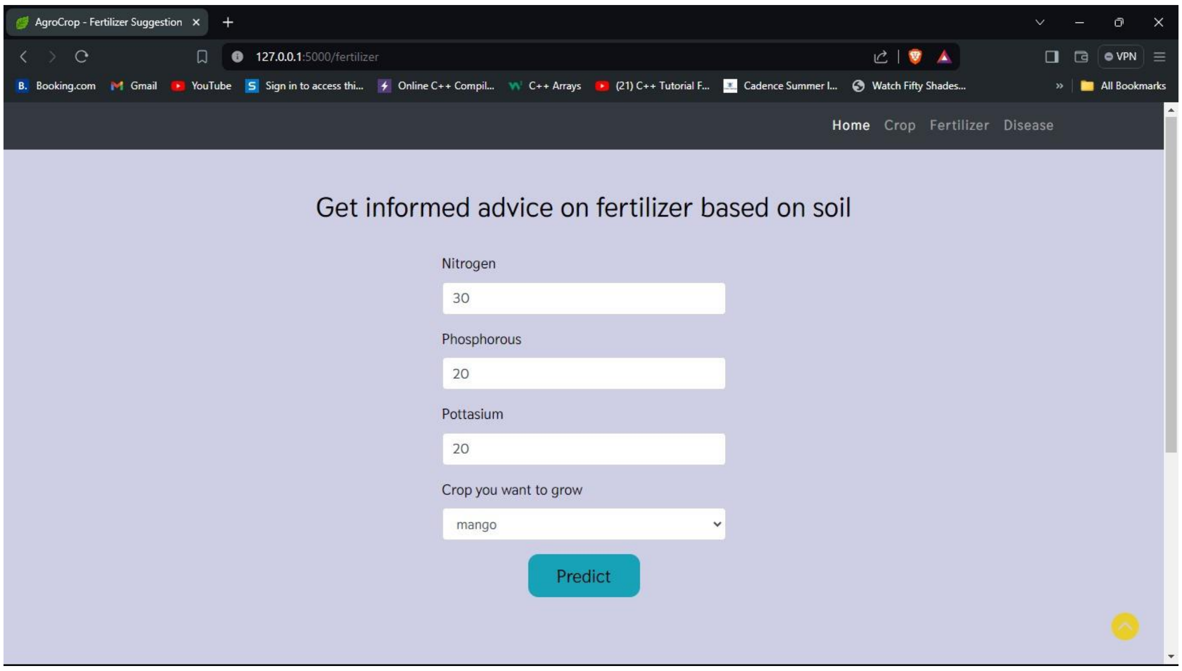
Nitrogen: Enter the value (example:50)

Phosphorous: Enter the value (example:50)

Pottasium: Enter the value (example:50)

Crop you want to grow: Select crop

Figure 5.8: Fertilizer suggestion system



Get informed advice on fertilizer based on soil

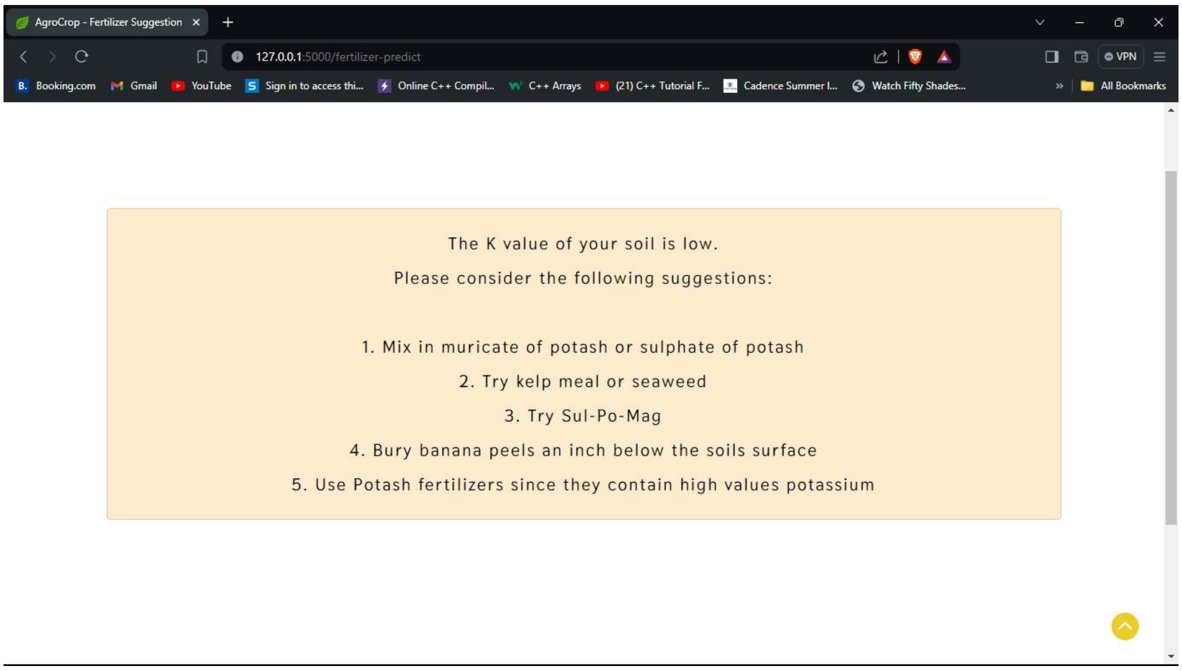
Nitrogen: 30

Phosphorous: 20

Pottasium: 20

Crop you want to grow: mango

Figure 5.9: Input sample values in fertilizer suggestion system


AgroCrop - Fertilizer Suggestion
127.0.0.1:5000/fertilizer-predict

The K value of your soil is low.

Please consider the following suggestions:

1. Mix in muricate of potash or sulphate of potash
2. Try kelp meal or seaweed
3. Try Sul-Po-Mag
4. Bury banana peels an inch below the soils surface
5. Use Potash fertilizers since they contain high values potassium

Figure 5.10: Result of fertilizer suggestion system

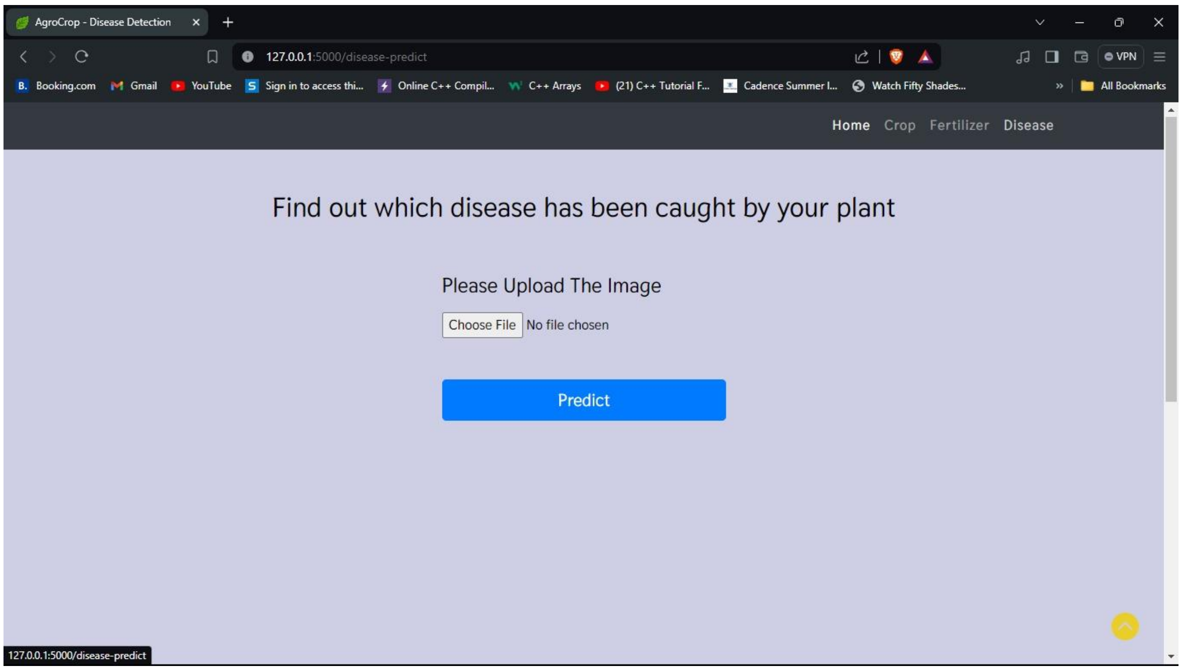

AgroCrop - Disease Detection
127.0.0.1:5000/disease-predict
Home
Crop
Fertilizer
Disease
Find out which disease has been caught by your plant
Please Upload The Image
Choose File
No file chosen
Predict

Figure 5.11: Disease Detection system

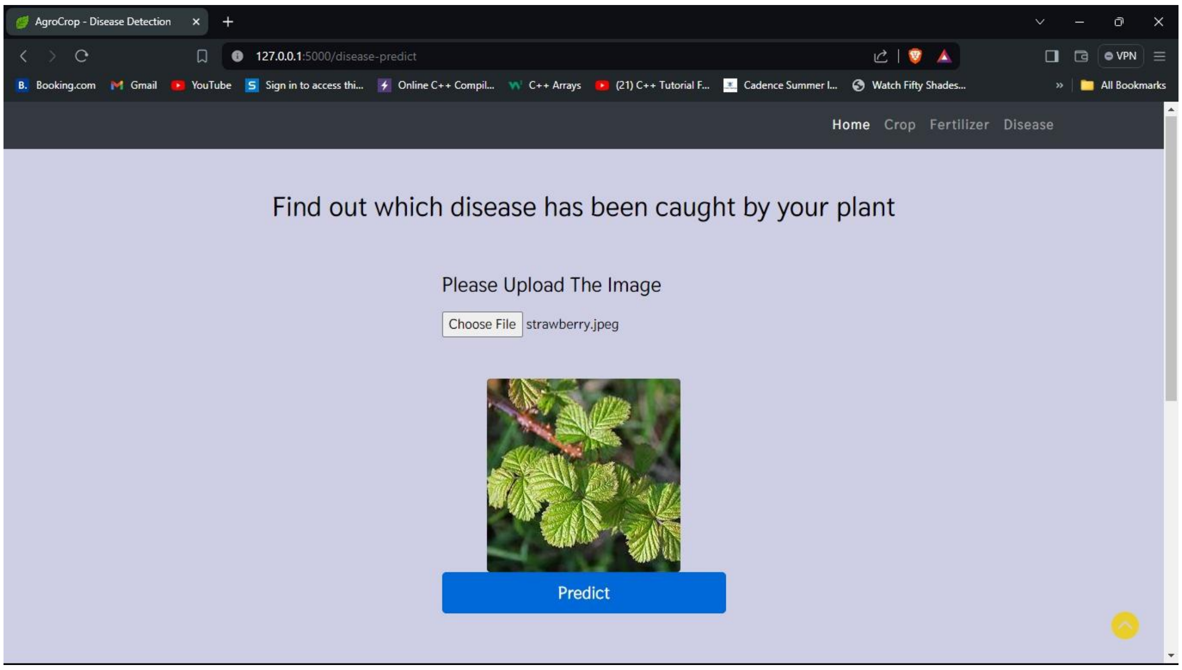
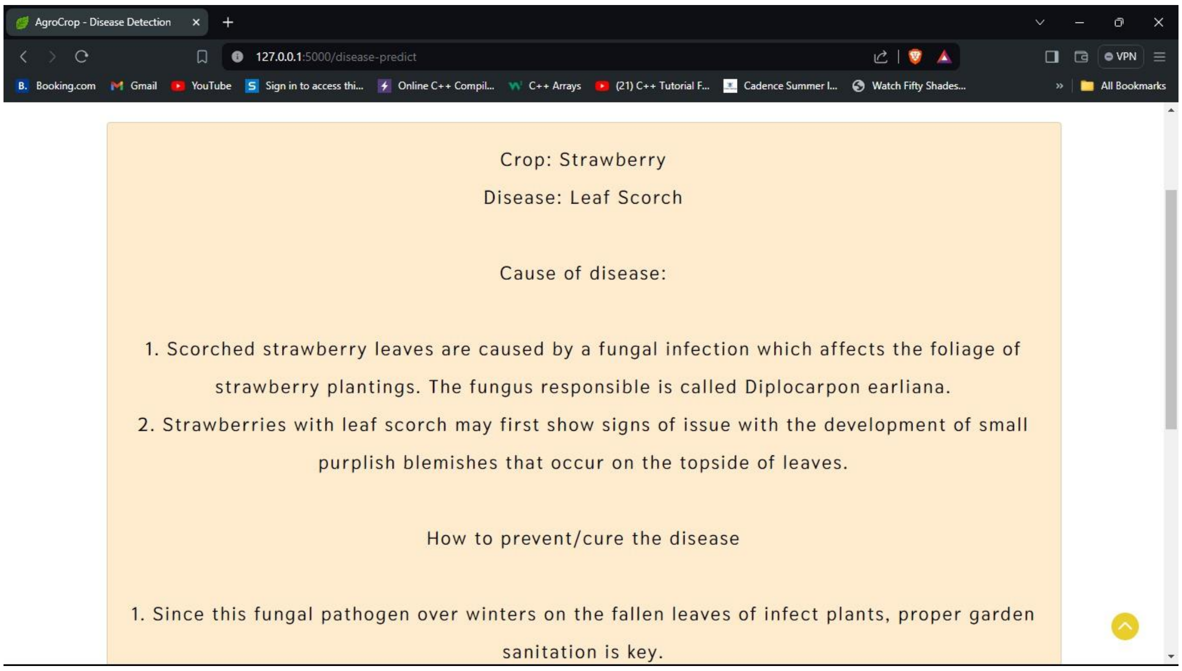

AgroCrop - Disease Detection
127.0.0.1:5000/disease-predict
Home
Crop
Fertilizer
Disease
Find out which disease has been caught by your plant
Please Upload The Image
Choose File
strawberry.jpeg
Predict

Figure 5.12: Input sample image in disease detection system



Crop: Strawberry

Disease: Leaf Scorch

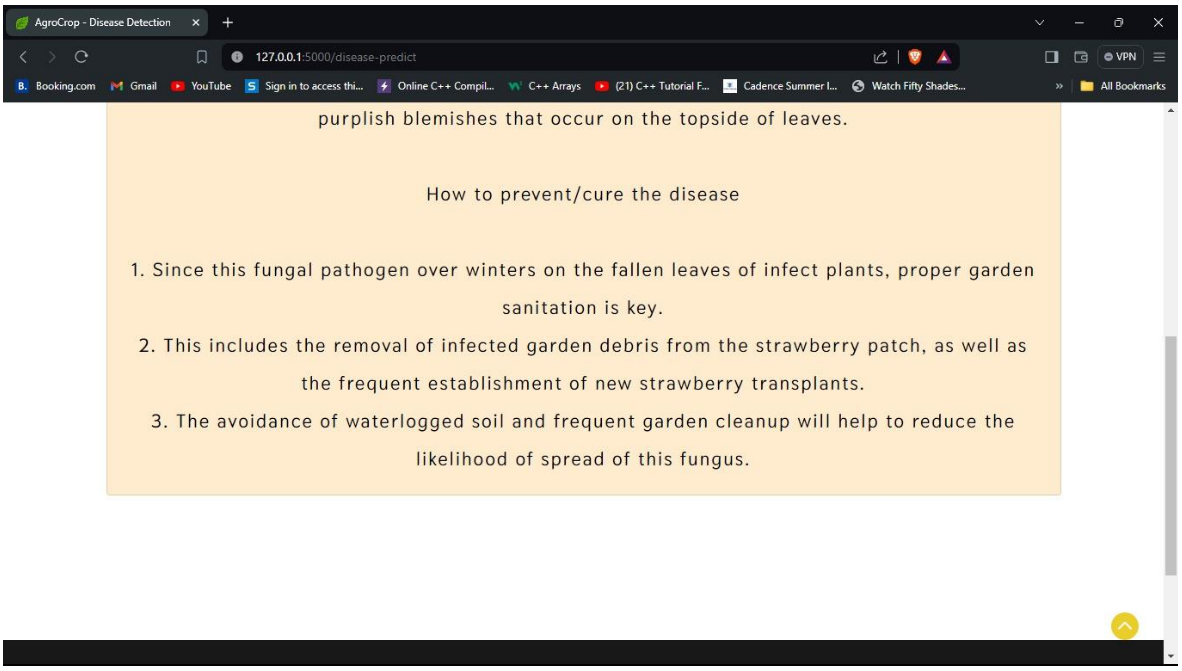
Cause of disease:

1. Scorched strawberry leaves are caused by a fungal infection which affects the foliage of strawberry plantings. The fungus responsible is called Diplocarpon earliana.
2. Strawberries with leaf scorch may first show signs of issue with the development of small purplish blemishes that occur on the topside of leaves.

How to prevent/cure the disease

1. Since this fungal pathogen over winters on the fallen leaves of infect plants, proper garden sanitation is key.

Figure 5.13: Result of disease detection system - 1



purplish blemishes that occur on the topside of leaves.

How to prevent/cure the disease

1. Since this fungal pathogen over winters on the fallen leaves of infect plants, proper garden sanitation is key.
2. This includes the removal of infected garden debris from the strawberry patch, as well as the frequent establishment of new strawberry transplants.
3. The avoidance of waterlogged soil and frequent garden cleanup will help to reduce the likelihood of spread of this fungus.

Figure 5.14: Result of disease detection system - 2

Chapter 6: Conclusions and Future Scope

6.1 Conclusion

Using this Smart Agriculture Crop Disease Detection project marks an important step in transforming centuries-old farming techniques and incorporating cutting edge technology for more effective crop surveillance. Based on this project, farmers should be made aware about some of the crop diseases which are able to spread using different methods like machine learning, image processing, among other technologies. Using machine learning algorithms system aimed to supply farmers with timely, valid and practical information for early disease detection in time for prompt actions and prevention.

This entire project is holistic which comprises gathering and cleansing of data, modeling, testing, and implementing interface. On the other hand, it is essential to consider the various difficulties involved in this process such as data quality, the strength from models, live detections, and infrastructure scaling across different agrarian areas. To overcome these hurdles calls for multi-disciplinary approach integrating Agricultural skills, Artificial intelligence, Software development, and technology application.

The success of this project lies not only in technical aspects but also in a capability to enhance farmers' lives, rational resource use, and boosting world food production. Consideration of these features in this project has an exciting pathway towards developing sustainable and productive agricultural system in the coming days.

Summary In essence, the Smart Agriculture Crop Disease Detection is a blend of technological advancement and agriculture expertise aimed at overhauling farming practice and food safety in an ever-evolving globe. This shows how much importance it placed on developing new methods using advancements in technology to improve agriculture by reducing crop losses in order to strengthen farmers' economic development whilst protecting the environment.

6.2 Future Scope

The Smart Agriculture Crop Disease Detection project opens doors to numerous future opportunities and advancements in agricultural technology:

Enhanced Model Accuracy:
Disease diagnosis using machine learning is an evolving technique that calls for continuous revising and improvement of the machine learning models so as to ensure enhanced precision and reliability. Such algorithms can be upgraded in future to capture a set of other different diseases and also consider many other climatic scenarios that have not been covered.

Integration of IoT and Sensor Data:
The dataset can be enhanced by adding information from IoT devices and sensors in agricultural fields, which can provide environmental parameters in real time. More thorough insights and early detection capabilities may be provided by integrating this data with disease detection models.

AI-Driven Decision Support Systems:
Including personalized recommendations through the use of AI-based decision support systems in this project would broaden its scope. Disease detection results can also be used by these systems to provide customized options, like corrective treatment measures, irrigation scheduling, and crop management plans.

Mobile Applications for Farmer Accessibility:
The technology can be available to farmers by developing mobile applications that are democratic in nature. Such apps should enable farmers to take and transmit photos of crops and receive an immediate diagnosis and treatment measures against a certain disease.

Expanding Crop and Disease Coverage:
Increasing the project’s scope beyond the selected crops and diseases will enhance its applicability. For instance, tackling certain diseases affecting different crops in distinct regions and weather condition may help greatly towards improving productivity of agriculture on a global scale.

Blockchain for Data Security:
By guaranteeing the legitimacy of gathered data and disease detection findings, blockchain technology implementation could improve data security and integrity. This would make the system more transparent and trustworthy.

Collaboration with Agricultural Research:
Agricultural research institute partnerships can promote ongoing innovation. Working together can improve our understanding of diseases and make it possible to incorporate the most recent research findings into detection models.

AI-Enabled Precision Agriculture:
Using of precision agriculture together with crop diseases detection would allow saving resources. The use of chemicals decreases as artificial intelligence-based treatments are applied locally thus lowering the environmental effects of crop production.

In summary, the next step for Smart Agriculture Crop Disease Detection is that it will always change with latest innovations in technology, dataset, or farming practice. This project has great potential for sustainability in global agricultural ecosystems by promoting innovation and collaboration.

References

1. C. Jackulin and S. Murugavalli "A comprehensive review on detection of plant disease using machine learning and deep learning approaches", Measurements: Sensors Elsevier, 2022
2. L. Li, S. Zhang and B. Wang, "Plant Disease Detection and Classification by Deep Learning—A Review," in IEEE Access, vol. 9, pp. 56683-56698, 2021.
3. H. Hong, J. Lin and F. Huang, "Tomato Disease Detection and Classification by Deep Learning," 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Fuzhou, China, 2020, pp. 25-29.
4. P. Panchal, V. C. Raman and S. Mantri, "Plant Diseases Detection and Classification using Machine Learning Models," 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bengaluru, India, 2019, pp.
5. B. Wang and D. Wang, "Plant Leaves Classification: A Few-Shot Learning Method Based on Siamese Network," in IEEE Access, vol. 7, pp. 151754-151763, 2019.
6. P. Jiang, Y. Chen, B. Liu, D. He and C. Liang, "Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks," in IEEE Access, vol. 7, pp. 59069-59080, 2019.
7. A. A. Sarangdhar and V. R. Pawar, "Machine learning regression technique for cotton leaf disease detection and controlling using IoT," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2017, pp. 449-454.
8. Mohanty SP, Hughes DP, Salathé M. Using Deep Learning for Image-Based Plant Disease Detection. Front Plant Sci. 2016 Sep 22;7:1419.
9. S. L. Pimm, C. N. Jenkins, R. Abell, T. M. Brooks, J. L. Gittleman, L. N. Joppa, P. H. Raven, C. M. Roberts, and J. O. Sexton, ''The biodiver- sity of species and their rates of extinction, distribution, and protection,'' *Science*, vol. 344, no. 6187, May 2014, Art. no. 7246752. doi: 10.1126/science.1246752.
10. O. Kulkarni, "Crop Disease Detection Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697390.
11. Weizheng, S., Zhanliang, C., Hongda, Yachun, W., W. (2008),"Grading Method of Leaf Spot Disease Based on Image Processing.", 2008 international Conference on

Computer Science and Software Engineering , Vol. 06, pp. 491-494, December 12 - 14, 2008.

12. Hillnhuetter, C. and A.-K. Mahlein, Early detection and localisation of sugar beet diseases: new approaches, Gesunde Pfianzen vol. 60 no. 4 , pp. 143–149, 2008
13. R. M. Haralick, K. Shanmugam,I. Dinstein, "Textural Features for Image Classification," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, no. 6, pp. 610–621, 1973
14. Haque, M.A., Marwaha, S., Deb, C.K. *et al.* Deep learning-based approach for identification of diseases of maize crop. *Sci Rep* 12, 6334 (2022).
15. Surbhi Jain, Joydip Dhar, "Image based search engine using deep learn- ing", 2017 Tenth International Conference on Contemporary Computing (IC3), August 2017
16. A. Caglayan *Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419*
17. C. DeChant et al., ''Automated identification of northern leaf blight- infected maize plants from field imagery using deep learning,'' Phy- topathology, vol. 107, no. 11, pp. 1426–1432, 2017.
18. C. Wu, C. Luo, N. Xiong, W. Zhang, and T. Kim, ''A greedy deep learning method for medical disease analysis,'' *IEEE Access*, vol. 6, pp. 20021–20030, 2018.
19. M. Dutot, L. M. Nelson, and R. C. Tyson, ''Predicting the spread of postharvest disease in stored fruit, with application to apples,'' Postharvest Biol. Technol., vol. 85, pp. 45–56, Nov. 2013.
20. Bi, C., Wang, J., Duan, Y. et al. MobileNet Based Apple Leaf Diseases Identification. Mobile Netw Appl 27, 172–180 (2022).
21. A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, ''Deep learning for image-based cassava disease detection,'' *Frontiers Plant Sci.*, vol. 8, p. 1852, Oct. 2017.
22. A.Johannes*etal.*,''Automaticplantdiseasediagnosisusingmobilecap- ture devices, applied on a wheat use case,'' *Comput. Electron. Agricult.*, vol. 138, pp. 200–209, Jun. 2017.
23. Sanjeev S. Sannakki, Vijay S Rajpurohit, V. B. Nargund and PallaviKulkarni, "Diagnosis and Classification of Grape Leaf Diseases using Neural Networks", *International Conference on Computing Communications and Networking Technologies IEEE, 2013.*

24. P. R. Rothe and R. V. Kshirsagar, "Cotton Leaf Disease Identification using Pattern Recognition Techniques", *International Conference on Pervasive Computing (ICPC) IEEE,* pp. 1-6, 2015.
25. P. Revathi and M. Hemalatha "Classification Of Cotton Leaf Spot Disease Using Image Processing Edge Detection Technique", *International Conference on Emerging Trends in Science, Engineering and Technology IEEE,* PP. 169-173, 2012.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): **PhD Thesis** **M.Tech Dissertation/ Report** **B.Tech Project Report** **Paper**

Name: ______________________________**Department:** _________________**Enrolment No** _________

Contact No. ______________________________**E-mail.** _______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at.................. (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Major_Project_2024_G-124

by Ravindara Bhatt

Submission date: 14-May-2024 03:01PM (UTC+0530)
Submission ID: 2244160333
File name: Major_Project_2024_G-124.pdf (4.25M)
Word count: 10904
Character count: 64022

Major_Project_2024_G-124

ORIGINALITY REPORT

15%	12%	6%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

4%
★ ir.juit.ac.in:8080
Internet Source

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		