A Minor Project Final Report on

Plant Disease Detection Using CNN Algorithm

Submitted in Partial Fulfillment of the Requirements for

the Degree of **Bachelor of IT Engineering**

under Pokhara University

Submitted by:

Prakash Mahara, 211527

Prashanta Rokaya, 211528

Suman Paneru, 211536

Under the supervision of

Er. Himal Acharya

Date:

23 February 2025

Department of IT Enginnering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur,Nepal

Acknowledgment

The success and final outcome of this project required a lot of guidance and assistance from many people, and we are very fortunate to have received this support throughout the completion of our project. We are deeply grateful to express our sincerest gratitude to our highly respected and esteemed supervisor,**M**r. Himal Acharya, for his invaluable supervision, guidance, encouragement, and support in completing this work. His useful suggestions and cooperative behavior are sincerely acknowledged.

We appreciate the guidance given by all other involved parties, especially our subject lecturers, which has significantly improved our research and presentation skills. Their comments and advice have been immensely helpful. Finally, we would like to express our sincere thanks to all our friends, seniors, and others who helped us directly or indirectly during this project

ABSTRACT

Plant Disease Detection System is a mobile application that helps users detect plant diseases by analyzing uploaded images of plant leaves. The system uses machine learning, specifically convolutional neural networks (CNNs), to identify diseases based on visual patterns and symptoms. Once a disease is detected, the system provides recommendations for treatment and prevention. The goal of this project is to simplify plant care by enabling users to easily diagnose diseases, which typically require expert knowledge. The system allows users to upload images of plant leaves and receive instant feedback about possible diseases, along with actionable recommendations for care. The system is built using Flutter with a machine learning model powered by TensorFlow. The model is trained on datasets like PlantVillage, which includes images of common plant diseases. This makes PlantCare an accessible and effective tool for both hobbyists and professional gardeners, helping them manage plant health more efficiently.

Keywords

PlantCare, MachineLearning, CNNs, Flutter, TensorFlow

Contents

List of Figures

1 INTRODUCTION

Plants are an integral part of ecosystems and agriculture, but their health is often compromised by diseases caused by bacteria, fungi, and viruses. Identifying these diseases accurately and promptly is crucial for effective treatment and prevention. However, traditional methods of disease detection rely on expert knowledge and manual observation, which can be time-consuming, error-prone, and inaccessible for many users.

Plant Disease Detection System addresses this challenge by providing a web-based platform that uses machine learning to detect plant diseases through image analysis. By leveraging convolutional neural networks (CNNs), the system identifies visual patterns and symptoms on plant leaves to diagnose diseases with high accuracy. Users can upload images of diseased leaves, and the system provides instant feedback, including the disease name and actionable recommendations for treatment and prevention.

1.1 PROBLEM STATEMENT

Identifying plant disease is a critical challenge for farmers and gardeners. Many plants show visible symptoms like spots, discoloration, and fungal growth, but accurate identification requires expertise. Current methods are time-consuming, prone to human error, and inefficient at scaling for large-scale applications. This project addresses the need for an automated solution to detect diseases quickly and accurately, enabling users to take timely corrective actions.

1.2 PROJECT OBJECTIVES

The main objectives of this project are:

- Develop a system capable of detecting plant disease using machine learning models.
- Enable users to upload plant images for real-time analysis.
- Create a responsive and user-friendly web interface.

1.3 SIGNIFICANCE OF THE STUDY

The significance of our plant disease detection system lies in several key areas:

- Improved Agricultural Productivity: The automated detection system reduces reliance on manual inspection, allowing for faster and more precise diagnosis, which enhances overall agricultural productivity.
- Cost-Effectiveness and Efficiency: The system minimizes the need for frequent expert consultations and laboratory testing, reducing costs while ensuring farmers can quickly access disease identification and treatment recommendations.
- Sustainable Farming Practices: By providing precise disease detection, the system helps in reducing the excessive use of pesticides, promoting eco-friendly and sustainable farming practices.

1.4 SCOPE AND LIMITATION

SCOPE:

- Detect diseases for plants included in the training dataset (e.g., Plant Village).
- Provide treatment recommendations based on detected diseases.
- Accessible through any modern browser on both desktop and mobile devices.

LIMITATIONS:

- Disease detection is restricted to the training dataset.
- Requires high-quality images for optimal results.
- Cannot detect multiple diseases in a single image.

2 LITERATURE REVIEW

Identification of diseases is one of the major area in agriculture which needs to be taken care of, though many practices have been done and implemented to cope up with this issue, rapid and quick identification of the diseases still remains in state of inchoate. The use of machine learning in facilitating the identification and detection heps to counter this problems to a much greater extent [1]. This paper talks about the detection and recognition of abnormalities of plants for training and study, potato leaves were taken. Random forest classifier was used for classification and it got trained using images of leaves with an almost seventy percent accuracy [2]. The common tomato leaf diseases like rust, grey spot, brown spot were discussed and found out with the help of deep learning algorithms and improved CNNs. The dataset for diseased leaves were generated, processed and collected. New deep CNN model designed to identify small diseased spots [3]. The most common bacterial, fungal and viral diseases are studied which affects the plant leaves and roots on wide scale and reduces the productivity of the plants can be easily studied and identified through RGB to grey scale conversions [4]. In machine learning various algorithms are available for feature extraction, clustering, segmentation. Selecting the most suitable as per needs of the task can be tough at times. To reduce the complexity and improve the time of response selection of the most suitable algorithm is required. We did a comparative study of the algorithms used in various previous projects and reasoned out the best one suitable for the project [5]. The algorithms used for classification are commonly applied in similar projects, such as Support Vector Machine (SVM) and Convolutional Neural Networks (CNN). Classification is a necessary step as it compares the values received after the feature extraction step with a pre-calculated set of data. For this project, we have opted to use the CNN algorithm, which is a powerful deep learning method particularly suited for image classification tasks. CNNs are highly effective at recognizing patterns and features in images, making them ideal for plant disease detection [6].

3 METHODOLOGY

3.1 SOFTWARE DEVELOPMENT LIFE CYCLE

We used the iterative Model of software development, which involves incremental enhancements until the final system is achieved.

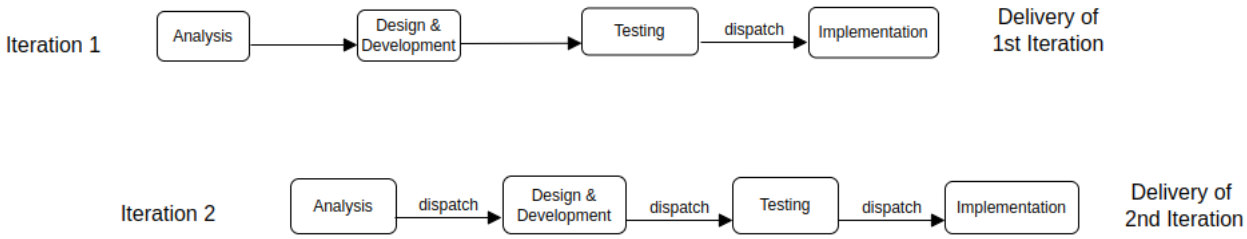


Figure 1: Iterative model

Key phases include:

3.1.1 REQUIREMENT ANALYSIS:

- a. Identified plant diseases, datasets, and machine learning frameworks.
- b. Specified user workflows for image uploads and result visualization.

3.1.2 DESIGN PHASE:

- a. Created diagrams: context diagrams, data flow diagrams, and use case diagrams.
- b. Designed a responsive user interface with Flutter.
-

3.1.3 CODING PHASE:

- a. Implemented the frontend using Flutter for an interactive experience.

- b. Used Python to create backend APIs for communication with the machine learning model.

3.1.4 TESTING AND EVALUATION:

- a. Tested disease detection accuracy using a test set from the PlantVillage dataset.
- b. Validated frontend responsiveness across devices and browsers.

3.2 TOOLS USED

We used the following tools to streamline the development process:

Git and GitHub:

Git is a distributed version control system that we are using to manage different versions of the project and facilitate collaboration among team members. GitHub is a platform that uses Git for version control. We are using GitHub to host the project repository, track changes, and manage different versions of the project.

VS Code:

Visual Studio Code (VS Code) is a code editor that we are using as the primary code editor for development tasks.

Android Studio:

Android Studio is the official integrated development environment (IDE) for Android app development, powered by IntelliJ IDEA. It provides tools like a code editor, visual layout editor, emulator, and debugging tools to streamline app creation. With support for Kotlin, Java, and C++, it enables developers to build high-quality Android apps efficiently.

Jupyter Notebook:

Jupyter Notebook is an open-source, web-based interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports multiple programming languages, including Python, and is widely used in

data analysis, machine learning, and scientific research. Its flexibility makes it a popular tool for both teaching and development.

Xampp:

XAMPP is a free, open-source software package that provides an easy way to set up a local web server. It includes Apache, MySQL (or MariaDB), PHP, and Perl, making it ideal for testing and developing web applications. XAMPP is cross-platform and user-friendly, suitable for beginners and experienced developers alike.

Latex(overleaf):

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents.We are using latex for our documentation tasks.

3.3 TECHNOLOGIES USED

The following technologies were used during the project development task:

FLUTTER(DART):

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and provides a rich set of customizable widgets for creating visually appealing and performant interfaces. With its hot-reload feature, developers can quickly iterate and test changes, making it a popular choice for cross-platform app development.

FAST API:

FastAPI is a modern, high-performance web framework for building APIs with Python, based on standard Python type hints. It is designed for speed, ease of use, and automatic generation of interactive API documentation. Its asynchronous capabilities and robust features make it ideal for creating scalable and fast web applications.

MySQL:

MySQL is a widely-used, open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It supports SQL (Structured Query Language) for managing and querying data and is commonly used in web applications and enterprise solutions. MySQL is compatible with various platforms and integrates seamlessly with programming languages like PHP, Python, and Java.

4 SYSTEM DESIGN

4.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between actors (in our case: User, and Admin) and the plant disease detection system. It shows the different use cases or functionalities provided by the platform and the relationships between the actors and these use cases.

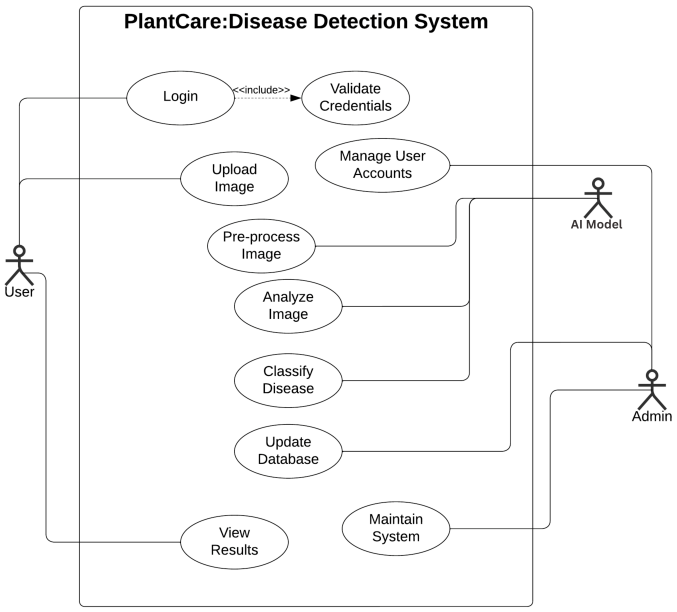


Figure 2: Usecase Diagram

4.2 ER DIAGRAM

An Entity-Relationship (ER) Diagram visually represents the data structure of our plant disease detection system. It shows the entities, such as users, plant, disease, model, prediction and their relationships, helping to organize and define the database schema.

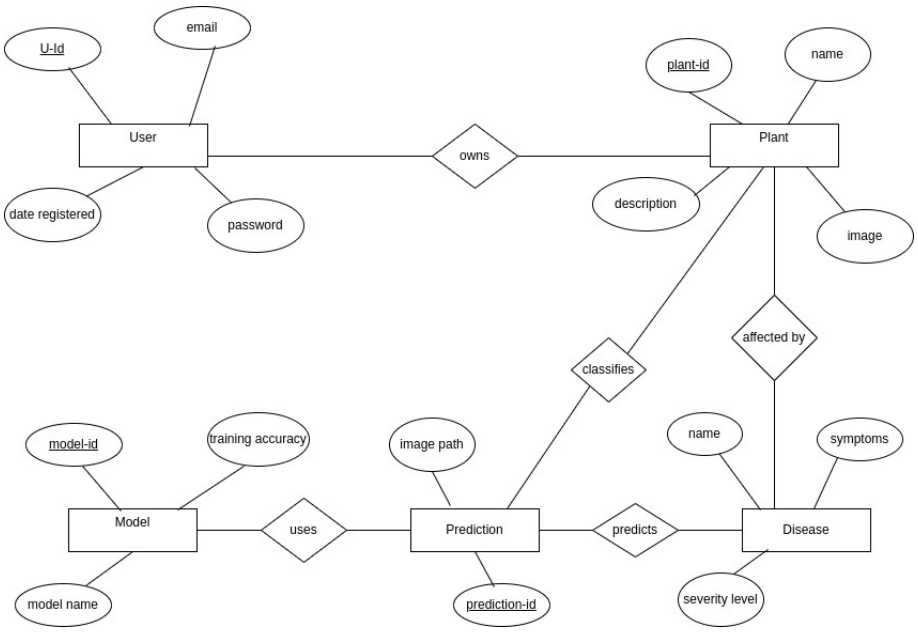


Figure 3: ER diagram

4.3 SEQUENCE DIAGRAM

A sequence diagram for our plant care disease detection system for farmers illustrates the process of detecting and managing plant diseases, showing the interactions between the Farmer, System, Database, and Machine Learning Model. The diagram begins with the farmer logging into the system and uploading an image of a plant leaf, followed by the system analyzing the image using the trained machine learning model. The system then retrieves relevant disease information from the database and provides the farmer with a diagnosis, treatment recommendations, and preventive measures. Additional diagrams could depict processes for user registration, real-time disease alerts, and accessing expert consultations.

4.3.1 SIGNUP

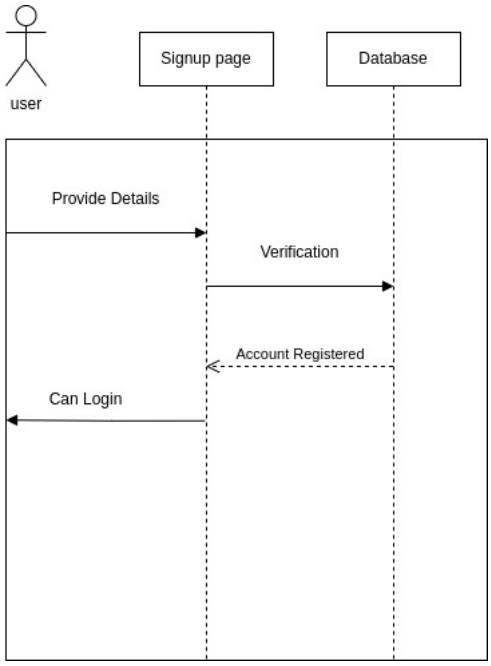


Figure 4: Sequence Diagram for Signup Page

4.3.2 LOGIN

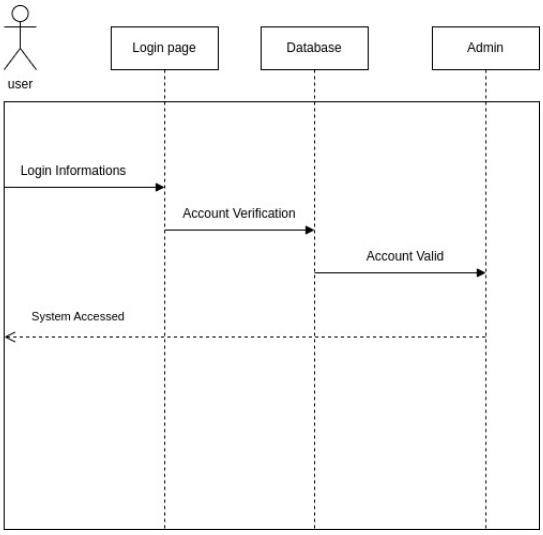


Figure 5: Sequence Diagram for User Login Page

4.3.3 IMAGE PROCESSING

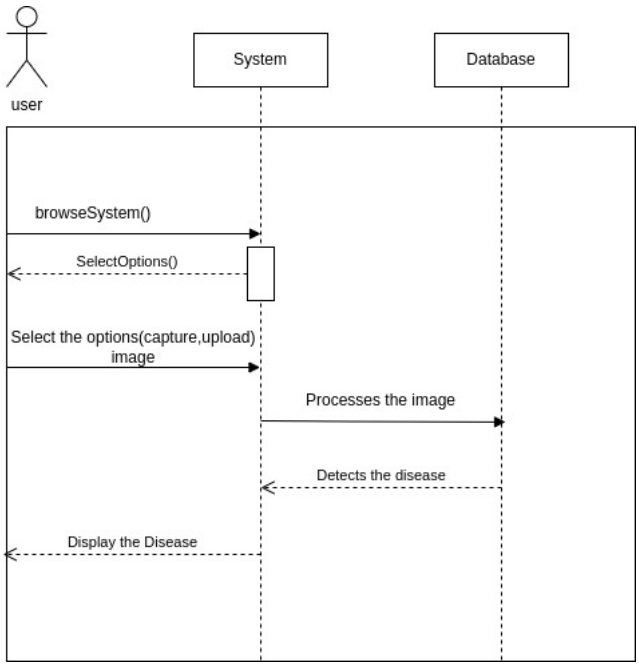


Figure 6: Sequence Diagram for Home Page

5 PROJECT TASK AND TIME SCHEDULE

5.1 TIME SCHEDULE

The project schedule has been planned with careful attention to requirements and constraints. We prioritized the Requirement Analysis phase to ensure a clear understanding of project needs and to stay aligned with the project goals. This focused approach will help ensure that the design, development,and testing, phases proceed smoothly and effectively.

Task	Duration (Weeks)	Start Date	End Date	Description
Requirement Analysis	1	Week 1	Week 1	Define project requirements, scope and design architecture.
Dataset Collection & Image processing	1	Week 2	Week 2	Gather plant disease images, process and prepare dataset.
Model Training and Evaluation	2	Week 3	Week 4	Train a CNN model, evaluate it's accuracy, and finalize for integration.
Frontend Development (Flutter)	2	Week 4	Week 5	Develop frontend with flutter for uploading and processing image.
Backend Development (FAST API)	2	Week 4	Week 5	Build APIs for processing uploaded images and connecting the model.
Frontend and Backend Integration	1	Week 6	Week 6	Integrate frontend and backend for seamless operation.
System Testing & Debugging	1	Week 7	Week 7	Perform end-to-end testing, debug issues, and optimize performance.
Deployment & Final Documentation	1	Week 8	Week 8	Deploy the system to the cloud and finalize the documentation.

Figure 7: Time Schedule

5.2 GANTT CHART

We followed the Incremental Software Process Model. At each increment, we set the timeline for the particular work. Each work wass divided into tasks, and tasks are set at each iterations to be completed.

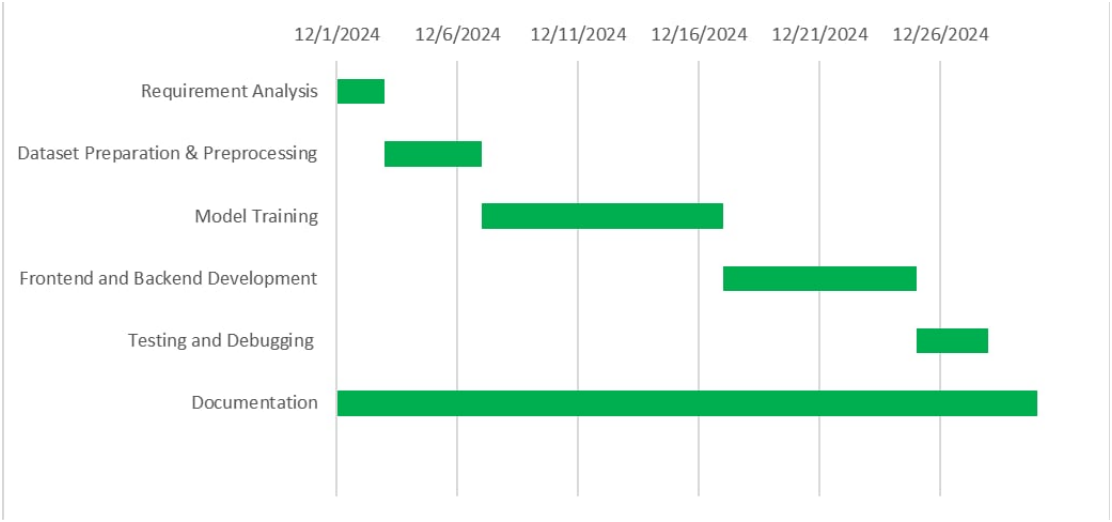


Figure 8: Iteration 1

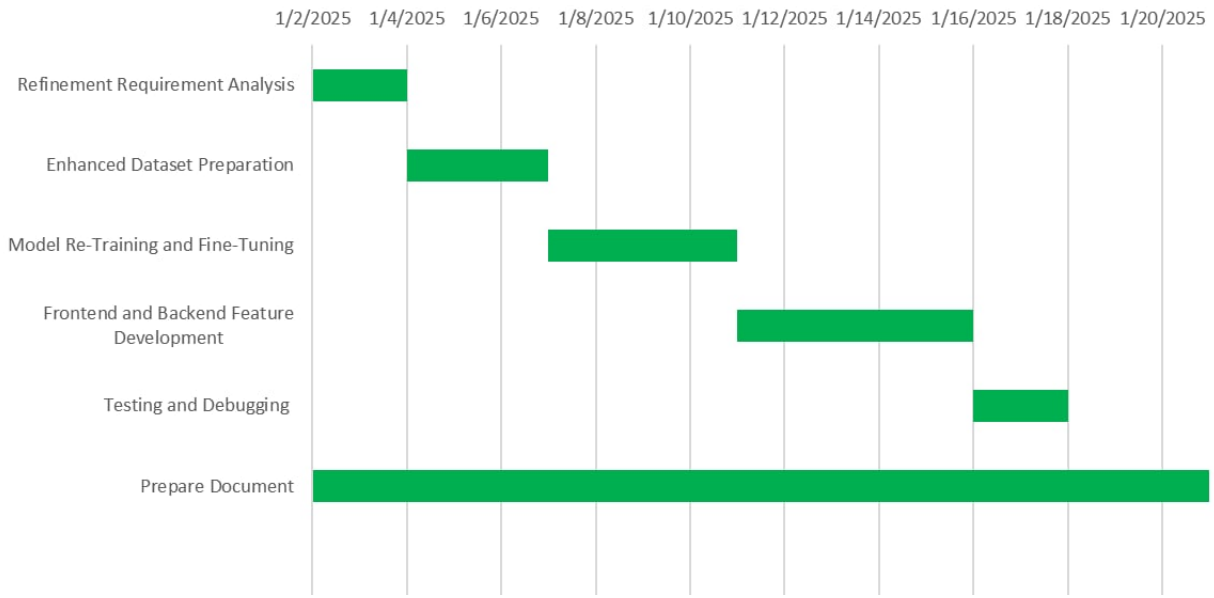


Figure 9: Iteration 2

6 CONCLUSION

The core objective of our project was to create a platform that enhances the productive farming. By transitioning from traditional methods to a digital and web-based system, we have successfully achieved this goal.

Based on this analysis, we designed our platform to address these issues effectively. We created and tested prototypes throughout the development process, using the incremental methodology to interactively build and refine the system. This approach allowed us to develop and test individual features, incorporating feedback and making improvements along the way. The entire development process took approximately 60 days to complete. Our disease detection platform is designed to modernize and enhance productive farming. It supports various functionalities including plant disease detection, classifying the plants, cure recommendations. The system enables farmers to be aware about possible plant disease and the cure of the disease. Through our project, we have successfully developed a comprehensive and user-friendly plant disease detection platform that improves the health of a plant by predicting disease and recommending the cure according to disease.

6.1 Tablessss

Image example	Obtained result	Expected results
	- leaf mold passalora fulva 0.99999 - septoria leaf spot septoria lycopersici 5 - mosaic virus 2.34571e-06 - late blight phytophthora tans 1.02782e-06 - early blight solani 2.06558e-07	- leaf mold passalora fulva

Table 1: Comparison of obtained and expected results

Class (disease type)	**Number of images**
Bacterial Spot *Xanthomonas campestris*	793
Early Blight *Alternaria solani*	406
Late Blight *Phytophthora infestans*	727
Leaf Mold *Passalora fulva*	361
Septoria Leaf Spot *Septoria lycopersici*	735
Two Spotted Spider Mite *Tetranychus urticae*	721
Target Spot *Corynespora cassiicola*	548
Mosaic Virus	140
Yellow Leaf Curl Virus	2101
Tomato leaves healthy	644
Total	**7176**

Table 2: Distribution of images by disease type

Class (disease type)	**Number of images**
Bacterial Spot *Xanthomonas campestris*	793
Early Blight *Alternaria solani*	406
Late Blight *Phytophthora infestans*	727
Leaf Mold *Passalora fulva*	361
Septoria Leaf Spot *Septoria lycopersici*	735
Two Spotted Spider Mite *Tetranychus urticae*	721
Target Spot *Corynespora cassiicola*	548
Mosaic Virus	140
Yellow Leaf Curl Virus	2101
Tomato leaves healthy	644
Total	**7176**

Table 3: Distribution of images by disease type

7 FURTHER WORK

1. Enhanced User Interface and Experience Improving the UI/UX of the plant disease detection system by making it more user-friendly, responsive, and accessible across various devices. Features such as real-time feedback, intuitive navigation, and interactive visualizations can enhance user engagement.

2. Additional Feature Integration Expanding system capabilities by incorporating features like multi-plant species detection, early disease prediction, and a recommendation system for disease treatment. Integration of real-time weather data and soil health analysis could further improve prediction accuracy.

3. Cross-Platform and Cloud Integration Enabling cloud-based storage and cross-platform compatibility will allow users to access the system from different devices seamlessly. Integration with IoT-based agricultural sensors can further enhance disease detection efficiency by utilizing real-time environmental data.

8 REFERENCES

1. PlantVillage Dataset, https://www.kaggle.com/datasets/emmarex/plantdisease. [Accessed: Nov. 28, 2024].

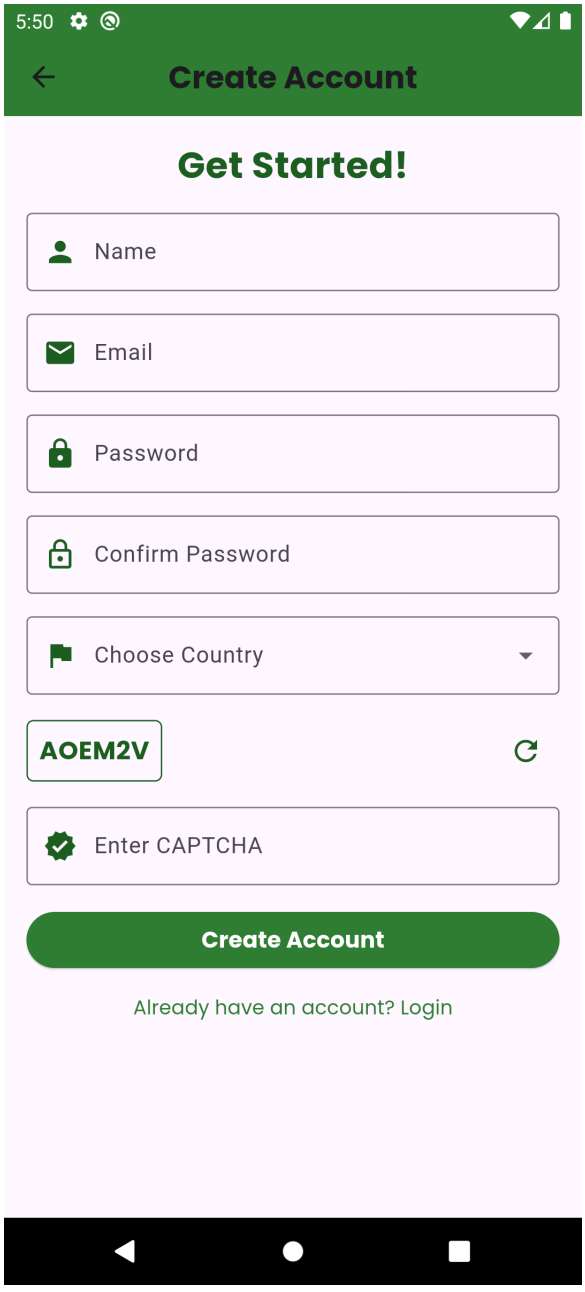
2. Z. N. Reza, F. Nuzhat, N. A. Mahsa, and M. H. Ali, ”Detecting jute plant disease using image processing and machine learning,” in Proc. 3rd Int. Conf. on Electrical Engineering and Information Communication Technology (ICEEICT), Dhaka, 2016, pp. 1-6.

3. S. Ramesh et al., ”Plant Disease Detection Using Machine Learning,” in Proc. 2018 Int. Conf. on Design Innovations for 3Cs Compute Communicate Control (ICDI3C), Bangalore, 2018, pp. 41-45.

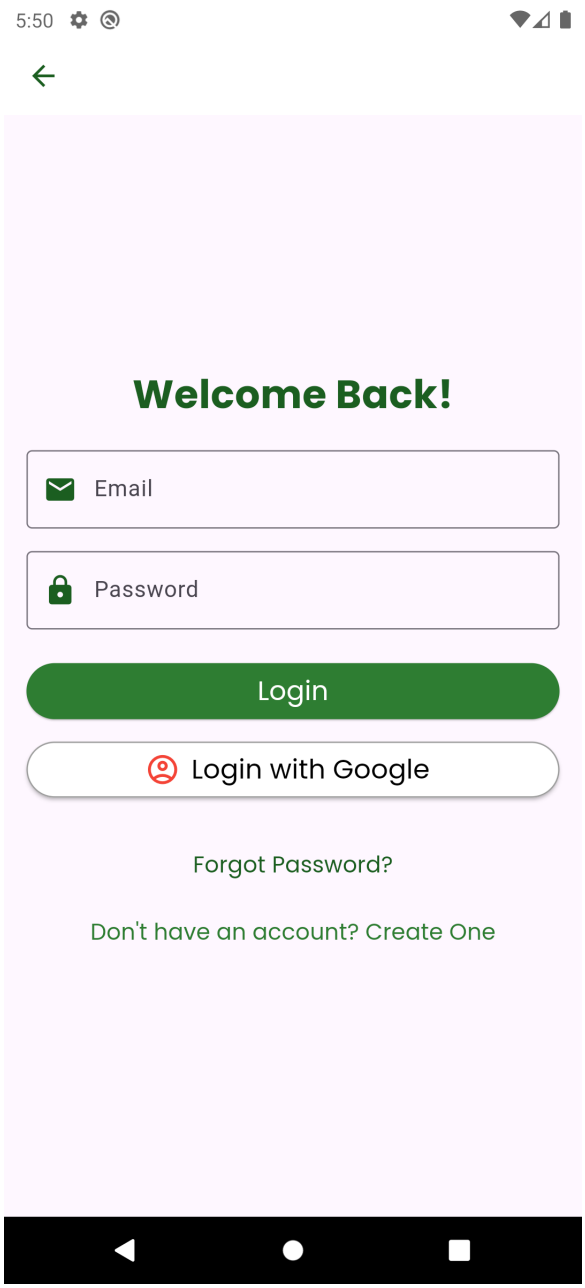
4. J. D. Pujari, R. Yakkundimath, and A. S. Byadgi, “Identification and classification of fungal disease affected on agriculture/horticulture crops using image processing techniques,” in Proc. IEEE Int. Conf. on Computational Intelligence and Computing Research, Coimbatore, 2014, pp. 1-4.

9 APPENDIX

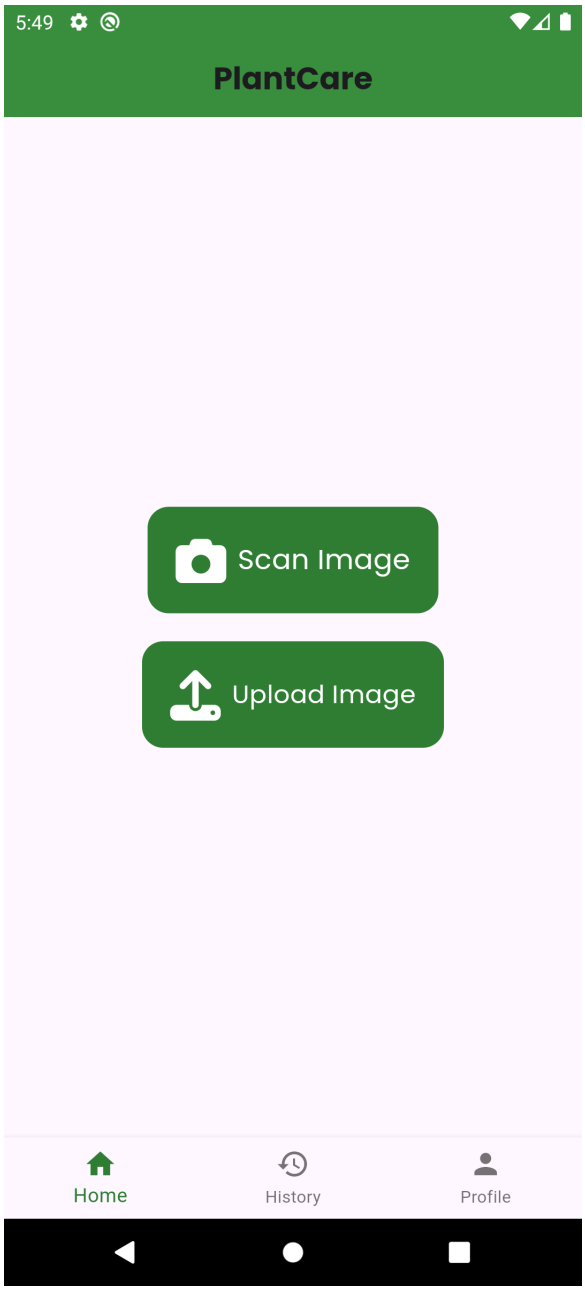
9.1 SIGNUP



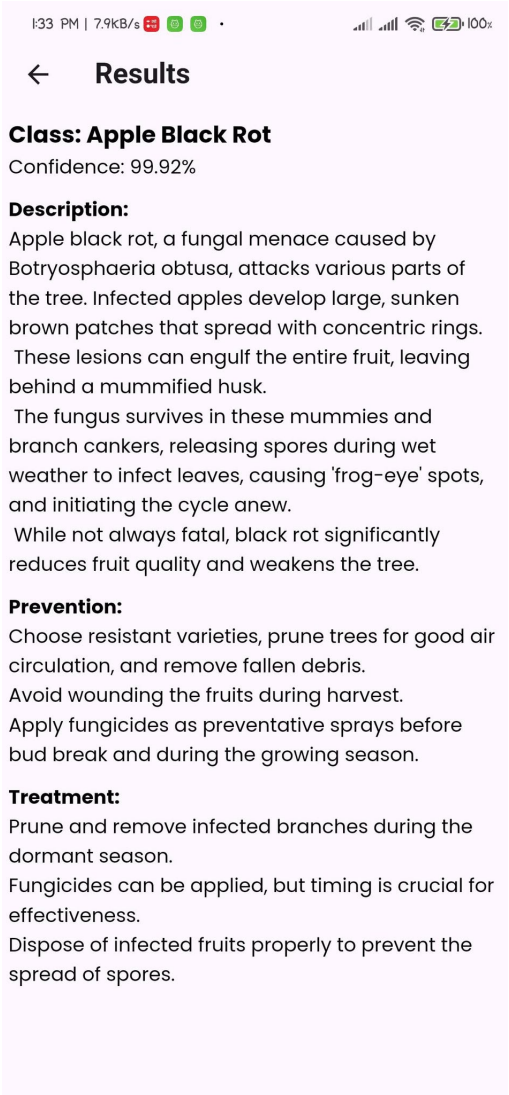
9.2 LOGIN PAGE



9.3 HOME PAGE


5:49
PlantCare
Scan Image
Upload Image
Home
History
Profile

9.4 DISEASE DETECTION PAGE



9.5 WARNING PAGE