"LEAF DISEASE DETECTION"

A Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfilment for the Award of

Bachelor Of Technology in Computer Science and Engineering

Submitted By:	**Guided By:**
Aditya Sharma (0827CS201016)	**Professor Juhi Shrivastav**
Amit Yadav (0827CS201031)	**Computer Science Engineering**
Aryan Tapkire (0827CS201044)	
Devesh Sharma (0827CS201068)	

ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH, INDORE

Jan - May 2024

EXAMINER APPROVAL

The Project entitled ***"Leaf Disease Detection"*** submitted by **Aditya Sharma (0827CS201016), Amit Kumar Yadav (0827CS201031), Aryan Tapkire (0827CS201044), Devesh Sharma (0827CS201068)** has been examined and is hereby approved towards partial fulfilment for the award of Bachelor of Technology degree in Computer Science and Engineering discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner)	**(External Examiner)**
Date:	**Date:**

RECOMMENDATION

This is to certify that the work embodied in this major project entitled "***Leaf Disease Detection***" submitted by **Aditya Sharma (0827CS201016), Amit Kumar Yadav (0827CS201031), Aryan Tapkire (0827CS201044), Devesh Sharma (0827CS201068)** is a satisfactory account of the bonafide work done under the supervision of ***Dr. Kamal Kumar Sethi***, is recommended towards partial fulfillment for the award of the Bachelor of Technology (Computer Science Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal.

(Project Guide)

(Project Coordinator)

(Dean Academics)

GUIDE RECOMMENDATION

This is to certify that the work embodied in this project entitled **"Leaf Disease Detection"** submitted by **Aditya Sharma (0827CS201016), Amit Kumar Yadav (0827CS201031), Aryan Tapkire (0827CS201044), Devesh Sharma (0827CS201068)** is a satisfactory account of the bonafide work done under the supervision of **Prof. Juhi Shrivastav** and it is recommended towards partial fulfilment for the award of the Bachelor of Technology (Computer Science and Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal.

(Project Guide) **(Project Coordinator)**

STUDENTS UNDERTAKING

This is to certify that, project entitled **“Leaf Disease Detection”** has been developed by us under the supervision of **Prof. Juhi Shrivastav**. The whole responsibility of work done in this project is ours. The sole intention of this work is only for practical learning and research.

We further declare that to the best of our knowledge; this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work is found then we are liable for explanation to this.

Aditya Sharma (0827CS201016)
Amit Kumar Yadav (0827CS201031)
Aryan Tapkire (0827CS201044)
Devesh Sharma (0827CS201068)

Acknowledgement

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Juhi Shrivastav**, Associate Professor, AITR, for their motivation, sagacious guidance, constant encouragement, vigilant supervision, and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

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

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

Aditya Sharma (0827CS201016), Amit Kumar Yadav (0827CS201031)

Aryan Tapkire (0827CS201044), Devesh Sharma(0827CS201068)

Executive Summary

Leaf Disease Detection

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (MP), India for partial fulfilment of Bachelor of Technology in Computer Science & Engineering branch under the sagacious guidance and vigilant supervision of **Prof. Juhi Shrivastav**.

The project is based on Leaf Disease Detection. The Leaf Disease Detection Android Application is a user-friendly online platform designed to assist users in accurately identifying various lead diseases quickly and conveniently. This application leverages image recognition technology and a comprehensive leaf disease database to provide users with accurate information about the leaf diseases they encounter.

Key words: Leaf Disease, TensorFlow, Image Recognition

"Difficulties in your life do not come to destroy you…

But to help you realise your hidden potential and power,

Let difficulties know that you too are difficult!"

~A.P.J. Abdul Kalam

List Of Figures

Table Of Contents

Chapter 1: Introduction

Introduction

Leaf disease detection plays a crucial role in various industries, such as agriculture, pharmaceuticals, and ecology. Accurate detection of Leaf disease helps in improving crop yields, developing new medicines, and understanding the ecological balance of an ecosystem.

For instance, in agriculture, identifying the right Leaf disease can help farmers optimize their crop management practices, leading to higher yields and reduced costs. In the pharmaceutical industry, accurate detection of Leaf disease can lead to the discovery of new medicines and treatments for various diseases.

1.1 Overview

Leaf disease detection is the process of classifying and categorizing different varieties or diseases of Leaves based on their visual characteristics. This task involves recognizing and distinguishing between various Leaf disease, sub disease, or cultivars by analysing their unique features such as color, shape, size, petal arrangement, and other botanical traits. Leaf disease detection can be conducted manually by experts or automated using machine learning and computer vision techniques.

1.2 Background and Motivation

Background:

1. Botanical Diversity: The natural world is home to an immense variety of Leaf disease and Leaf diseases. Botanists and horticulturists have long been fascinated by this diversity and have sought to catalogue and study it.
2. Scientific Research: Understanding and documenting the different Leaf diseases is essential for scientific research, especially in the fields of botany, biology, and agriculture. Researchers need accurate methods for classifying and studying various Leaf disease and cultivars.
3. Horticulture and Agriculture: In horticulture and agriculture, identifying specific Leaf diseases is critical for diseasing, cultivation, and crop management. Accurate detection ensures the selection of suitable varieties for specific purposes, such as landscaping or food production.
4. Conservation Efforts: Conservationists and environmentalists are concerned about preserving rare or endangered Leaf diseases. Accurate detection is necessary to monitor and protect this disease.

Motivation:

1. Educational and Outreach: Accurate Leaf disease detection can facilitate educational programs, botanical garden tours, and outreach efforts aimed at raising awareness about Leaf diversity. It can engage the public in the world of botany and horticulture.

2. User Convenience: For enthusiasts, gardeners, and nature lovers, having a tool or application that can quickly identify Leaf diseases can enhance their experience. It enables them to learn more about the Leaves they encounter in their surroundings.
3. Environmental Monitoring: Leaf disease detection can play a role in environmental monitoring and conservation efforts. It allows scientists to track the distribution and health of specific Leaf disease, including those at risk of extinction.
4. Agriculture and Horticulture: In the agricultural and horticultural sectors, accurate disease detection contributes to improved crop management, pest control, and diseasing programs. It helps growers make informed decisions about which cultivars to Leaf and how to care for them.
5. Conservation: For conservationists, knowing the precise identity of a Leaf disease is essential for prioritizing and implementing conservation measures. It helps protect biodiversity and preserve ecosystems.

1.3 Problem Statements and Objectives

Problem Statement:

There are about 380,000 known diseases of Leaves on the planet. Accurate detection of Leaf disease helps in improving crop yields, developing new medicines, and understanding the ecological balance of an ecosystem.

Objective:

- The Aim of this project is to develop a fully functioning Web application for the purpose of Detection of Leaf disease.
- Creating an attractive UI for the User to use the application.
- Enabling the user to upload photos of the leaf of the Leaf which they want to identify in the web application.
- Upon clicking on the predict button, the user will be shown the actual disease of Leaf along with some relevant Information about it.

1.4 Scope of the Project

Leaf disease detection is the art and science of improving the genetic makeup of Leaves in relation to their economic use for mankind. The scope of Leaf disease detection programs is to find objectives of various improved characteristics in Leaves for sustainable survival and optimum economic yield. The objective of Leaf disease detection is to develop improved characteristics of Leaves for more demanding economically as well as agronomically.

1.5 Group Organization

- **Aditya Sharma**

I investigated and found the right technology and studied it. For the implementation of the project. Implementation logic for the project objective and coding of internal functionalities is also done by me. I worked on the front-end, making the HTML.

- **Amit Kumar Yadav**

Along with preliminary investigation and understanding the drawback of the current system, I studied about the topic and its scope. I surveyed various research papers related to chatbots and the technology to be used. I also contributed to the documentation phase of the project.

- **Aryan Tapkire**

I investigated, found the right technology, and studied it in depth. I also organized and debugged the code of the project. I also tested the overall functionality of the project.

- **Devesh Sharma**

I worked on the overall documentation of the project. I also collected the object data and trained the model for it. Moreover, I managed the overall structure of the project, its design and working.

1.6 Report Structure

The project "Weather Forecasting Using Augmented Reality" is primarily concerned with providing support to users about weather conditions with 3d visualization.

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

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

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

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

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

Chapter 2: Review of Literature

Review of Literature

Leaf disease detection reveals a broad and multidisciplinary field that encompasses various aspects of Leaf disease detection, genomics, phenotyping, and data analysis. This review aims to provide an overview of key themes, methodologies, and recent developments in Leaf disease detection.

1. Genomic Approaches:

Genomic tools, such as next-generation sequencing (NGS) and marker-assisted selection (MAS), have revolutionized Leaf disease detection. Researchers use these techniques to identify and validate molecular markers associated with desired traits in Leaves.

Advances in high-throughput sequencing technologies have enabled whole-genome sequencing, facilitating the detection of genetic variations within Leaf disease detection populations.

2. Phenotyping and Trait Detection:

Traditional phenotypic assessment remains essential for Leaf disease detection. Researchers employ various techniques, including field trials, remote sensing, and imaging technologies, to evaluate Leaf traits such as yield, disease resistance, and stress tolerance.

The integration of machine learning and artificial intelligence (AI) in image analysis has expedited phenotyping processes, making them more accurate and efficient.

3. Marker Development:

Single Nucleotide Polymorphisms (SNPs) and other molecular markers play a pivotal role in Leaf disease detection. Researchers develop marker panels specific to different Leaf varieties, allowing for rapid and cost-effective detection.

4. Data Integration and Analysis:

Managing and analysing vast genomic and phenotypic datasets are critical challenges in Leaf disease detection. Bioinformatics tools and statistical methods are employed to process and interpret these data effectively.

Genome-wide association studies (GWAS) and quantitative trait locus (QTL) mapping are widely used to associate genetic markers with specific traits.

5. Crop-Specific Studies:

The literature contains numerous crop-specific studies focusing on Leaf disease detection for various economically important crops, such as rice, wheat, maize, soybeans, and tomatoes. These studies often emphasize unique challenges and strategies associated with each crop.

6. Diversity and Germplasm Conservation:

Identifying and preserving genetic diversity in Leaf disease detection populations are crucial for long-term agricultural sustainability. Research in this area emphasizes the detection of unique and valuable germplasm resources.

7. Marker-Assisted Disease detection:

Diseases increasingly use molecular markers to accelerate the disease detection process. This approach allows for the selection of desirable traits at an early stage, reducing the time required to develop new Leaf varieties.

8. Ethical and Regulatory Considerations:

As genetic modification and gene editing techniques advance, ethical and regulatory considerations surrounding Leaf disease detection become more complex. Ensuring the responsible use of these technologies is a growing concern.

9. Global Collaboration:

International collaborations and data sharing initiatives are becoming more prevalent in Leaf disease detection. These efforts aim to harness the collective knowledge and resources of the global research community.

10. Future Directions:

The field of Leaf disease detection continues to evolve rapidly. Emerging technologies like CRISPR-Cas9 and advancements in multi-omics approaches are likely to reshape the landscape of Leaf disease detection.

In conclusion, Leaf disease detection is a dynamic and multidisciplinary field that draws on genomics, phenotyping, data analysis, and innovative technologies. As researchers continue to advance our understanding of Leaf genetics and develop more efficient tools and methods, the future holds promising opportunities for improving crop yield, quality, and resilience to global agricultural challenges.

2.1 Preliminary Investigation

2.1.1 Current System

Various systems and technologies are in use for Leaf disease detection. It is important to note that advancements in this field may have occurred since then. Here are some key components and methods that are commonly used in the current systems for Leaf disease detection:

1. Genomic Tools:

Next-Generation Sequencing (NGS): NGS technologies like Illumina and PacBio have become more affordable and accessible, enabling researchers to sequence and analyse the genomes of different Leaf varieties quickly.

2. Molecular Markers:

Single Nucleotide Polymorphisms (SNPs): SNPs are widely used as genetic markers for Leaf disease detection. High-density SNP arrays and genotyping-by-sequencing (GBS) techniques are commonly employed.

3. Phenotyping Technologies:

Remote Sensing: Satellite and drone-based remote sensing technologies help collect large-scale, high-resolution data on crop performance, growth, and health.

High-Throughput Phenotyping: Automated systems equipped with cameras and sensors enable non-destructive, rapid assessment of Leaf traits in controlled environments.

4. Bioinformatics and Data Analysis:

Genome-Wide Association Studies (GWAS): GWAS allows the detection of genetic markers associated with specific traits through the analysis of large-scale genomic and phenotypic data.

Machine Learning and AI: Advanced machine learning algorithms are applied to analyse and interpret complex genetic and phenotypic datasets for predictive modelling.

5. Databases and Repositories:

Publicly accessible databases like the Leaf Genome Database and Germplasm Resources Information Network (GRIN) provide valuable genetic and phenotypic data for researchers.

6. Marker-Assisted Selection (MAS):

Diseases use molecular markers to select Leaves with desired traits, accelerating the disease detection process.

7. CRISPR-Cas9 and Gene Editing:

Gene editing technologies, like CRISPR-Cas9, enable precise modification of Leaf genomes, allowing for the creation of new Leaf varieties with desired traits.

8. Ethical and Regulatory Considerations:

Ethical frameworks and regulatory guidelines govern the use of genetic modification and gene editing techniques in Leaf disease detection, ensuring responsible and safe practices.

9. Collaborative Platforms:

Collaboration between academic institutions, agricultural companies, and international organizations is common in Leaf disease detection efforts, fostering knowledge sharing and data exchange.

10. Phenotypic Data Integration:

Integrating multi-omics data (genomic, transcriptomic, proteomic, and metabolomic) is becoming increasingly important for a comprehensive understanding of Leaf traits.

11. Crop-Specific Systems:

Many crop-specific systems and tools are developed to cater to the unique needs of different Leaf varieties, considering factors like growth conditions, diseases, and market demands.

12. Germplasm Conservation:

Initiatives focused on the conservation and utilization of genetic resources are essential for maintaining genetic diversity and ensuring long-term agricultural sustainability.

2.2 Limitations of Current System

There are several limitations associated with the current systems for Leaf disease detection. These limitations can impact the efficiency and effectiveness of Leaf disease detection efforts. Keep in mind that advancements may have occurred since then, but here are some common limitations:

1. Complexity of Traits:

Many important Leaf traits are controlled by multiple genes and are influenced by environmental factors. Identifying and characterizing these complex traits can be challenging.

2. Phenotype-Genotype Gaps:

There can be a disconnect between the observed phenotype and the underlying genotype due to gene interactions, epigenetic effects, and other factors. This can hinder accurate trait prediction.

3. Data Integration Challenges:

Integrating diverse data types, such as genomic, phenotypic, and environmental data, can be complex. Developing methods to effectively combine and analyse these data is an ongoing challenge.

4. Data Volume and Management:

The massive amount of data generated by high-throughput sequencing and phenotyping technologies can strain data storage and processing capabilities. Managing and analysing such large datasets can be resource-intensive.

5. Phenotyping Constraints:

Field-based phenotyping is often labour-intensive, time-consuming, and subject to environmental variability. This can limit the scale and efficiency of Leaf disease detection.

6. Genetic Diversity Preservation:

Maintaining genetic diversity within disease detection populations is essential for long-term agricultural sustainability. The over-reliance on a few elite cultivars can lead to genetic erosion.

7. Ethical and Regulatory Hurdles:

Ethical concerns and regulatory restrictions related to genetic modification and gene editing can impede the adoption of advanced disease detection technologies.

8. Validation of Marker-Trait Associations:

The validation of markers associated with specific traits is essential. False positives and overfitting in statistical models can lead to unreliable marker-trait associations.

9. Resource Limitations:

Smaller research groups and agricultural institutions may have limited access to cutting-edge technologies and resources for Leaf breed detection.

10. Accessibility and Data Sharing:

Unequal access to genetic resources and data can create disparities in Leaf disease detection efforts globally. Encouraging data sharing and international collaboration is essential.

11. Lack of Crop-Specific Tools:

Some crops, especially those considered "minor" or less economically important, may have fewer resources and tools available for disease detection.

12. Environmental Variation:

Environmental conditions, such as climate change and regional differences, can influence the performance of Leaf varieties, making it challenging to identify universally adaptable diseases.

13. Consumer and Market Preferences:

Meeting consumer preferences and market demands is essential in Leaf disease detection. However, predicting future market trends and preferences can be uncertain.

14. Long Disease detection Cycles:

Developing new Leaf varieties through traditional disease detection methods can take several years or even decades, limiting the speed of response to emerging agricultural challenges.

15. Pest and Disease Evolution:

Rapid evolution of pests and diseases can render previously resistant Leaf varieties susceptible, necessitating ongoing monitoring and adaptation.

Efforts to address these limitations involve ongoing research, technological advancements, and international collaborations to improve the accuracy, efficiency, and sustainability of Leaf disease detection. Researchers and diseases continually work to develop innovative solutions to overcome these challenges.

2.3 Requirement Detection and Analysis for Project

Leaf disease detection involves a multidisciplinary approach that integrates various data sources and methodologies to identify, develop, and characterize new Leaf varieties. The requirements for Leaf disease detection can be categorized into several key aspects:

1. Leaf Selection and Disease detection Objectives:

Define the specific Leaf disease or crop of interest.

Clearly state the disease objectives, such as improving yield, disease resistance, nutritional content, or other traits.

2. Data Collection and Phenotyping:

Collect and record comprehensive phenotypic data related to the target traits.

Implement high-throughput phenotyping techniques for efficiency.

Ensure accurate and standardized data collection methods.

3. Genomic Data:

Acquire genomic data, including DNA sequences, markers, and genetic maps, for the Leaf disease under investigation.

Use next-generation sequencing technologies for whole-genome sequencing, if applicable.

4. Genetic Markers:

Develop or utilize molecular markers (e.g., SNPs) for tracking and identifying specific traits.

Validate marker-trait associations through genetic analysis.

5. Data Management and Storage:

Establish a robust data management system to organize, store, and retrieve large volumes of genomic and phenotypic data.

Ensure data security and backup mechanisms.

6. Data Integration and Analysis:

Utilize bioinformatics tools and software for data integration, quality control, and analysis.

Perform genome-wide association studies (GWAS), quantitative trait locus (QTL) mapping, and other genetic analyses.

7. Infrastructure and Technology:

Invest in high-performance computing resources for data analysis and modelling.

Use advanced laboratory equipment for DNA extraction, sequencing, and genotyping.

8. Regulatory Compliance:

Adhere to relevant regulations and ethical guidelines, especially when using genetic modification or gene editing techniques.

Obtain necessary permits and approvals.

9. Quality Assurance:

Implement quality control measures for data accuracy and reliability.

Maintain data traceability and transparency throughout the project.

10. Ethical and Legal Considerations:

Address ethical considerations related to Leaf disease detection and genetic research.

Comply with intellectual property laws and regulations when developing new Leaf varieties.

11. Environmental and Field Trials:

Conduct controlled field trials to evaluate the performance of new Leaf varieties under varying environmental conditions.

Monitor and document field trial results.

12. Evaluation Criteria:

Define criteria for evaluating the success of the Leaf disease detection project, including quantitative measures of trait improvement and economic impact.

By addressing these requirements, a Leaf disease detection project can be conducted efficiently and effectively, leading to the development of improved Leaf varieties with desirable traits.

2.3.1 Conclusion

In conclusion, a Leaf disease detection analysis project is a complex and multifaceted endeavour that plays a crucial role in advancing agriculture, food security, and sustainable farming practices. This project involves the integration of diverse data sources, cutting-edge technologies, and interdisciplinary expertise to achieve specific disease detection objectives.

Chapter 3: Proposed System

Proposed System

3.1 The Proposal

The proposal is to build a web application. A web application for Leaf disease detection would provide a user-friendly and accessible platform for accessing Leaf information on the go. With careful consideration of features and functionalities, a Leaf disease detection application for web could provide significant value to users across a range of industries and settings.

3.2 Benefits of the Proposed system

A Leaf disease detection project can offer a wide range of benefits to various stakeholders, including researchers, diseases, growers, and consumers. Here are some of the key benefits of such a project:

1. Improved Floral Varieties:

Development of new and improved Leaf varieties with desirable traits, such as vibrant colors, unique shapes, longer vase life, and disease resistance.

2. Biodiversity Preservation:

Conservation and preservation of floral biodiversity by identifying and safeguarding rare or endangered Leaf disease and varieties.

3. Enhanced Aesthetics:

Access to a wider selection of visually appealing Leaves for floral arrangements, landscaping, and decorative purposes.

4. Efficient Disease detection:

Streamlined disease detection processes using genetic markers and modern biotechnological tools, reducing the time and resources required to develop new varieties.

5. Disease Resistance:

Detection and propagation of Leaf diseases with natural resistance to common pests and diseases, reducing the need for chemical pesticides.

6. Increased Crop Yields:

Development of high-yielding Leaf varieties that can lead to higher Leaf production and potentially increased revenue for growers.

In summary, a Leaf disease detection project not only contributes to the development of aesthetically pleasing and sustainable floral varieties but also has broader implications for agriculture, biodiversity conservation, economic growth, and cultural significance. It aligns with the evolving demands and preferences of consumers while promoting research and innovation in the field of floriculture.

3.3 Block Diagram

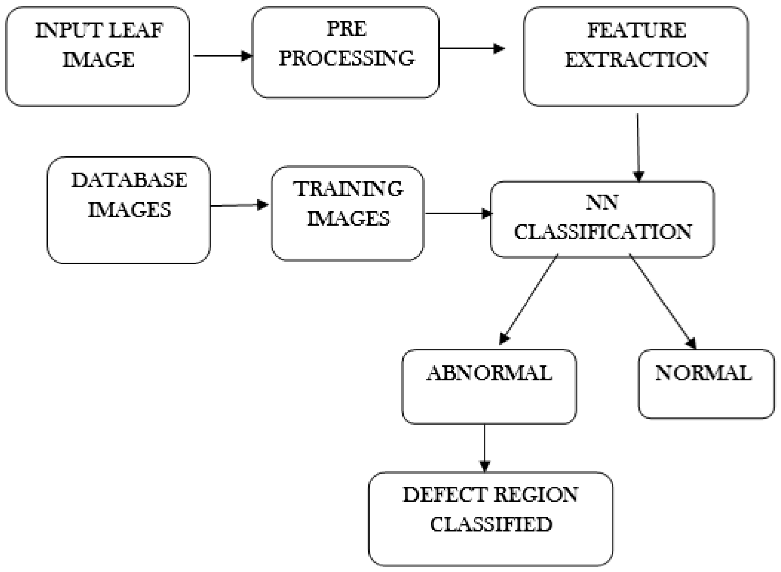

Figure 3-1: Block Diagram of Leaf Disease Detection

3.4 Feasibility Study

3.4.1 Technical

The technical feasibility of a Leaf disease detection project involves assessing whether the project can be successfully implemented from a technical perspective. This assessment considers the availability of resources, technologies, and expertise required to achieve the project's objectives. Here are some key factors we considered while evaluating the technical feasibility of a Leaf disease detection project:

1. Data Availability:

Genomic Data: Determine if there is access to genomic data for the Leaf disease of interest. This may include DNA sequencing data, genetic markers, and genetic maps.

Phenotypic Data: Assess the availability of phenotypic data related to Leaf traits, such as color, shape, size, fragrance, and disease resistance.

Environmental Data: Consider whether environmental data, such as climate, soil conditions, and geographic information, are available and relevant for the project.

2. Data Quality and Quantity:

Evaluate the quality and quantity of available data. High-quality, comprehensive datasets are essential for accurate disease detection and analysis.

3. Biotechnological Techniques:

If genetic modification or gene editing techniques are part of the project, evaluate the feasibility of implementing these technologies, including access to laboratories and regulatory compliance.

4. Data Integration:

Determine how data from multiple sources will be integrated, managed, and stored securely.

Evaluate the feasibility of establishing data pipelines and databases for efficient data processing.

5. Regulatory and Ethical Compliance:

Assess the regulatory and ethical considerations related to working with genetic data, especially if genetic modification or gene editing is involved.

Ensure compliance with intellectual property laws and data sharing agreements.

6. Scalability and Long-Term Sustainability:

Consider the scalability of the project to handle larger datasets or expand to additional Leaf disease.

Assess the long-term sustainability of the project, including data maintenance, updates, and support.

By conducting a thorough technical feasibility analysis, we determined whether the necessary resources, technologies, and expertise are available or can be acquired to successfully carry out the Leaf disease detection project. Addressing potential challenges and constraints early in the planning phase can help ensure the project's technical viability and success.

3.4.1 Economical

Evaluating the economic feasibility of a Leaf disease detection project is crucial to determine whether the project is financially viable and sustainable. This assessment involves analysing the costs, benefits, and potential returns on investment. Here are key factors we considered while assessing the economic feasibility this project:

Costs:

1. Personnel Costs: Estimate the salaries and benefits for the project team members, including researchers, data scientists, geneticists, and support staff.
2. Equipment and Technology: Calculate the costs of laboratory equipment, computers, software licenses, and high-performance computing resources required for data analysis and genetic research.
3. Data Acquisition Costs: Assess the expenses associated with obtaining genomic, phenotypic, and environmental data. This may include data licensing fees, data collection, and data purchase costs.

4. Infrastructure Costs: Evaluate the expenses for setting up and maintaining databases, servers, and data storage infrastructure.
5. Research and Development: Consider the costs related to research activities, including experiments, field trials, and development of genetic markers.
6. Regulatory Compliance: Budget for any costs associated with obtaining permits, complying with regulatory requirements, and conducting safety assessments, especially if genetic modification or gene editing is involved.
7. Training and Skill Development: Include the costs of training and skill development for project team members to ensure they have the necessary expertise.

A comprehensive economic feasibility analysis determines whether the benefits of the Leaf disease detection project outweigh the costs. It provides insights into the project's financial sustainability and potential returns on investment, assisting in decision-making and resource allocation.

3.4.1 Operational

Operational feasibility for a Leaf disease detection project assesses whether the project can be successfully executed and integrated into existing operations. This evaluation involves examining the practical aspects of implementing the project. Here are key considerations we assessed for the operational feasibility of a Leaf disease detection project:

1. Resource Availability:

Determine whether the necessary resources, including personnel, equipment, and facilities, are available or can be acquired within the project's timeframe.

2. Data Availability and Quality:

Verify the availability and quality of genomic, phenotypic, and environmental data required for the project. Ensure data is accurate, relevant, and up-to-date.

3. Compliance and Regulation:

Assess the feasibility of complying with regulatory requirements, especially if the project involves genetic modification or gene editing. Determine the steps required for obtaining permits and approvals.

4. Workflow and Processes:

Define and optimize the workflow and processes involved in Leaf disease detection, including data collection, analysis, and decision-making.

5. Training and Capacity Building:

Plan for training and capacity-building programs to enhance the skills and knowledge of project team members and stakeholders.

6. Timeline and Milestones:

Created a detailed project timeline with clear milestones to track progress and ensured that the project stays on schedule.

7. Change Management:

Prepare for potential changes in operations and workflows resulting from the implementation of new Leaf disease detection strategies and technologies.

8. Scalability:

Consider the scalability of the project to accommodate potential expansion in terms of the number of Leaf disease or varieties to be studied.

9. Operational Risks and Contingencies:

Identify potential operational risks, such as data loss, resource constraints, or equipment failures, and develop contingency plans to mitigate these risks.

By conducting a thorough operational feasibility analysis, we identified potential challenges and ensured that the Leaf disease detection project can be executed efficiently and effectively within the available resources and operational constraints.

3.5 Design Representation

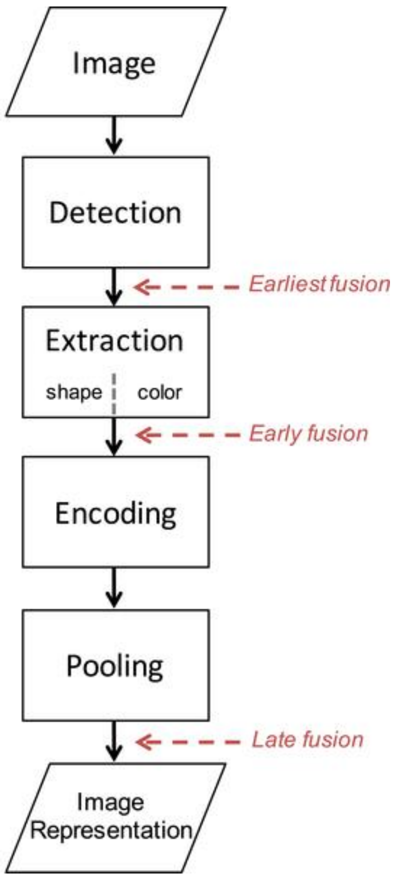

Figure 3-2: Design Representation of Leaf Disease Detection

3.6 Deployment Requirements

3.6.1 Hardware

Developer's Requirement Analysis: -

- Hardware Processor > 2GHz.

User's Requirement Analysis: -

- Secured Local Area Network and Internet Connectivity.

3.6.2 Software

Developer's Requirement Analysis: -

- VS Code and Python IDE
- Leaf Disease dataset
- Windows 10 or above

User's Requirement Analysis: -

- Android Device

Chapter 4: Implementation

Implementation

The implementation of a Leaf disease detection project involved executing the various tasks and activities outlined in the project plan. It encompasses data collection, analysis, software development (if applicable), and collaboration with relevant stakeholders

1. Data Collection:

Collect genomic, phenotypic, and environmental data for the Leaf disease or varieties under study. This may involve laboratory experiments, field trials, or data acquisition from external sources.

2. Data Management:

Establish data management protocols to organize, store, and secure the collected data. Ensure data quality control and backup procedures are in place.

3. Data Integration:

Develop data integration pipelines or scripts to combine and preprocess data from multiple sources, ensuring consistency and compatibility.

4. Data Analysis:

Utilize bioinformatics tools, statistical methods, and machine learning algorithms to analyse the data for disease detection. Identify genetic markers and associations with desirable traits.

5. Collaboration and Data Sharing:

Collaborate with research institutions, botanical gardens, or other organizations to share data, expertise, and resources. Foster collaboration to enhance research outcomes.

6. Regulatory Compliance:

Ensure compliance with relevant regulations, especially if genetic modification or gene editing is involved. Obtain necessary permits and approvals.

7. Quality Assurance:

Implement quality control measures to verify the accuracy and reliability of data, analysis results, and software functionalities.

8. Testing and Validation:

Thoroughly test the software application (if applicable) to ensure it functions as intended. Validate the results of disease detection against known standards and benchmarks.

9. Scalability Considerations:

Ensure that the project is designed to accommodate scalability if there is potential for expanding to study more Leaf disease or varieties.

Throughout the implementation phase, close coordination among project team members and a commitment to best practices in data management and analysis were essential for the successful execution of the Leaf disease detection project.

4.1 Technique Used

4.1.1 TensorFlow

TensorFlow is a powerful machine learning framework that we utilized in various aspects of Leaf disease detection project. Here are some key ways in which TensorFlow is used in our project:

1. Convolutional Neural Networks (CNNs) for Image Analysis:

TensorFlow provides a framework for building CNNs, a class of deep learning models that excel in image analysis.

CNNs can be trained on Leaf images to learn features and patterns that distinguish different Leaf diseases.

Transfer learning using pre-trained models like Inception or Mobile Net can be employed to leverage existing knowledge for faster and efficient training.

2. Image Preprocessing:

TensorFlow's image preprocessing capabilities help prepare Leaf images for analysis.

Techniques like resizing, normalization, and augmentation can be applied to improve model training.

3. Data Augmentation:

TensorFlow's data augmentation functions can generate additional training examples by applying transformations like rotation, flipping, and cropping to existing Leaf images.

Augmentation helps improve the model's robustness and generalization.

4. Model Customization:

TensorFlow allows for the customization and fine-tuning of pre-trained models to adapt them to the specific requirements of Leaf disease detection.

Model architectures can be modified, and additional layers can be added to suit the task.

5. Transfer Learning:

Transfer learning involves using pre-trained models and fine-tuning them on a Leaf disease dataset.

TensorFlow facilitates this process by providing pre-trained models and tools for transfer learning.

6. Model Training and Optimization:

TensorFlow offers GPU and TPU support for accelerating model training, making it feasible to handle large datasets and complex model architectures.

Optimization techniques like dropout, batch normalization, and learning rate scheduling can be employed for improved model performance.

7. Inference and Deployment:

After training, TensorFlow models can be used for inference to classify Leaf images into different diseases.

TensorFlow Serving and TensorFlow Lite are tools for deploying models to production environments or embedded systems, respectively.

8. Model Evaluation:

TensorFlow provides metrics and tools for model evaluation, allowing you to assess the performance of your Leaf disease detection model on validation and test datasets.

4.2 Tools Used

4.2.1 Android Studio

Android Studio is the official environment for developing apps for Android devices, created by Google. It is based on JetBrains' IntelliJ IDEA, a popular IDE for programmers.

Here is a quick rundown of Android Studio:

What it is: Integrated development environment (IDE) for Android app development

Made by: Google, based on JetBrains' IntelliJ IDEA

What you can use it for: Build apps for Android phones, tablets, and other devices

Languages: Primarily Java and Kotlin, also supports JavaScript and C++ with extensions

Operating systems: Windows, macOS, Linux (including ChromeOS)

If you are interested in learning more about Android Studio, here are some helpful resources:

Download and install: You can download the latest version of Android Studio from the official Android developer website https://developer.android.com/studio.

4.3 Language Used

Python:

Python is a high-level, versatile, and easy-to-read programming language known for its simplicity and readability. It is widely used for various applications, including web development, data analysis, machine learning, scientific computing, and automation. Python's extensive standard library and large ecosystem of third-party packages make it a popular choice for developers. It is an excellent language for beginners due to its straightforward syntax and strong community support.

Kotlin:

Kotlin is a modern, general-purpose programming language that is particularly well-suited for Android development. Here is a breakdown of Kotlin:

Type: Statically typed, which helps catch errors early in the development process

Uses: Primarily for Android app development, but also useful for server-side development and more

Benefits:

Concise and readable code compared to Java

Improved developer productivity

Enhanced code safety with features like null safety

Can be used for cross-platform development with Kotlin Multiplatform

XML:

XML, or Extensible Markup Language, is a way to store and transport data that is both human-readable and machine-readable. It is like a filing system for information, but with the flexibility to create custom categories for any kind of data you need.

Here are some key points about XML:

Purpose: Stores and transmits data in a structured format

Structure: Uses tags to define elements and attributes within the data

Advantages:

Flexible: You can create your own tags to fit your specific data needs

Portable: XML files can be easily shared and understood by different programs

Human-readable: The structure of XML makes it easier for people to understand the data

Machine-readable: Computers can easily parse and process XML data.

4.4 Screenshots

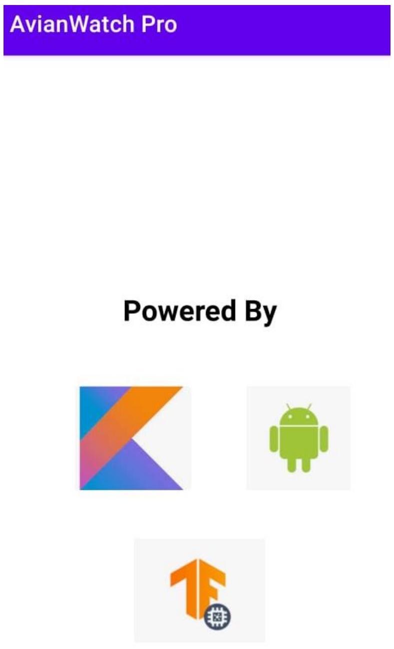

Figure 4-1: Start Screen

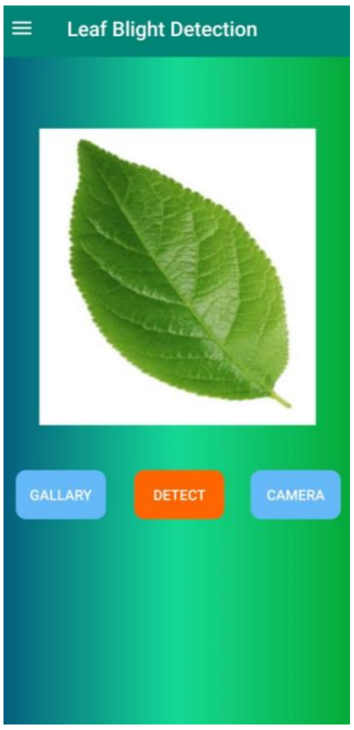

Figure 4-2: Model Implementation

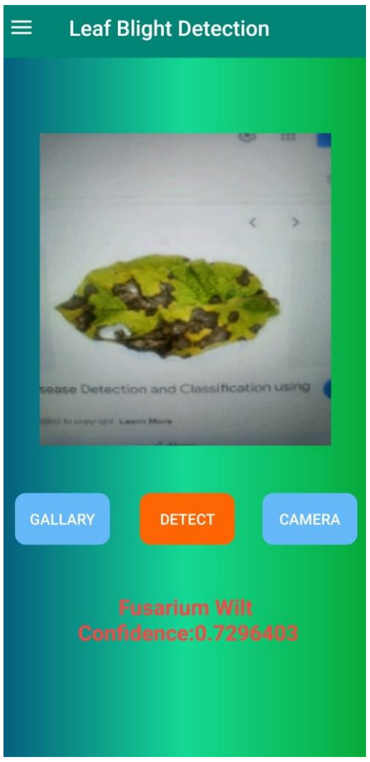

Figure 4-3: User Interface

4.5 Testing

Functional Testing: -

Functional testing for a Leaf disease detection system involves evaluating the system's functionality to ensure that it performs its intended tasks accurately and effectively. Here is a step-by-step guide on how we conducted functional testing for a Leaf disease detection project:

1. Define Test Cases:

Start by defining a set of test cases that cover all the essential functionalities of the Leaf disease detection system. Each test case should have a clear objective and expected outcomes.

2. Data Preparation:

Prepare a test dataset that includes a variety of Leaf images representing different diseases. Ensure that the dataset covers various scenarios and potential challenges.

3. Testing Scenarios:

Identify different testing scenarios based on the key functionalities of the system, such as disease classification, accuracy, and response time.

4. Test Data Input:

Feed the prepared test dataset into the system as input. This can be done programmatically or through the system's user interface, depending on the testing requirements.

5. Classification Accuracy:

Verify the accuracy of disease detection. Ensure that the system correctly identifies and classifies the Leaf diseases in the test dataset. Compare the system's predictions to the ground truth labels.

6. Error Handling:

Test how the system handles various types of errors, such as incorrect file formats, missing data, or unexpected input. Ensure that appropriate error messages are displayed.

7. Performance Testing:

Evaluate the system's performance, including its response time and resource utilization. Check if it can process a given number of images within an acceptable timeframe.

8. Compatibility Testing:

Test the system on different web browsers and devices to ensure compatibility. Verify that it works consistently across various platforms.

9. Test Validation:

Once issues are resolved, retest the affected areas to validate that the fixes have been successfully implemented.

10. Test Completion:

When all test cases have been executed, reviewed, and validated, and the system meets the defined criteria for functionality, consider the functional testing phase complete.

Implementation Testing: -

Implementation testing for a Leaf disease detection system involves verifying that the system has been correctly implemented according to its specifications and requirements. This type of testing focuses on the technical aspects of the system to ensure that it functions as intended. Here are the steps that we conducted for Leaf disease detection project:

1. Review Requirements:

Begin by thoroughly reviewing the system's requirements, including functional and non-functional requirements, to establish a clear understanding of what needs to be tested.

2. Test Environment Setup:

Set up a testing environment that mirrors the production environment as closely as possible. Ensure that all necessary hardware, software, and dependencies are properly configured.

3. Test Data Preparation:

Prepare a representative test dataset that includes a variety of Leaf images, covering different diseases, lighting conditions, and backgrounds. This dataset should align with the system's expected real-world usage.

4. Test Cases Definition:

Define test cases that cover all aspects of the system's functionality, including image input, processing, classification, and user interactions (if applicable).

5. Image Input Testing:

Test the system's ability to accept and process image inputs of various formats, resolutions, and sizes.

Verify that the system handles both single images and batches of images.

6. Image Preprocessing Testing:

Validate that any image preprocessing steps, such as resizing, normalization, or data augmentation, are correctly implemented.

Ensure that preprocessing does not introduce artifacts or distortions in the images.

7. Disease Detection Testing:

Test the core functionality of disease detection. Verify that the system accurately classifies Leaf images into their respective diseases.

Assess the system's accuracy, precision, recall, and F1-score for disease detection.

8. Compatibility Testing:

Verify that the system functions correctly across different web browsers, operating systems, and devices.

Ensure that the user experience remains consistent and error-free on various platforms.

9. Integration Testing:

Test the integration of various system components and modules to ensure they work harmoniously together.

Verify that data flows correctly between components.

10. Test Validation:

After issues are resolved, retest the affected areas to confirm that the fixes have been successfully implemented and do not introduce new problems.

4.5.1 Strategy Used

Testing our web application requires a comprehensive testing strategy to ensure that the application functions correctly, meets user requirements, and delivers a great user experience. Here are the strategies we used for testing our web application:

1. Manual Testing: Manual testing is a testing strategy where the tester tests the application by following a test plan and executing test cases manually. Manual testing is necessary to validate the functionality of the application, user interface, usability, and user experience.
2. Automation Testing: Automation testing is a testing strategy that involves using tools to automate the testing process. Automation testing is useful for repetitive and time-consuming testing tasks such as regression testing. Automation testing can help increase test coverage, reduce testing time, and increase the overall quality of the application.
3. User Acceptance Testing (UAT): User acceptance testing is a testing strategy that involves testing the application from the end-user's perspective. UAT is necessary to ensure that the application meets the user's requirements and delivers a great user experience. UAT can be performed by selecting a group of end-users to test the application and provide feedback.
4. Performance Testing: Performance testing is a testing strategy that involves testing the application's performance under different load conditions. Performance testing is necessary to ensure that the application can handle many users and data without any issues.
5. Security Testing: Security testing is a testing strategy that involves testing the application's security features. Security testing is necessary to ensure that the application is free from security vulnerabilities and is secure from unauthorized access.
6. Compatibility Testing: Compatibility testing is a testing strategy that involves testing the application's compatibility with different devices, operating systems, and screen sizes. Compatibility testing is necessary to ensure that the application functions correctly on different devices and screen sizes.

4.5.2 Analysis

Here are some areas we analysed after testing our web application:

1. Functionality: Analysing the functionality of the application by comparing the testing results against the requirements to identify any defects or gaps. This can help determine whether the application meets user requirements and performs as expected.
2. User Interface: Analysing the user interface of the application by determining whether it is easy to use and provides a good user experience. Testing the application on different screen sizes and resolutions to ensure that it functions correctly on different devices.

3. Performance: Analysing the performance of the application by testing its performance under different load conditions to ensure that it can handle many users and data without any issues.
4. Security: Analysing the security of the application by performing security testing to ensure that the application is free from security vulnerabilities and is secure from unauthorized access.
5. Compatibility: Analysing the compatibility of the application by testing its compatibility with different devices, operating systems, and screen sizes to ensure that the application functions correctly on different devices and screen sizes.

By analysing the results of our testing, we identified issues or defects and addressed them to improve the quality of the application.

Chapter 5: Conclusion

Conclusion

5.1 Conclusion

In conclusion, the Leaf disease detection project has been a significant endeavour aimed at leveraging technology to enhance our understanding of floral diversity and aid in the detection of various Leaf diseases. Through this project, we have achieved several important milestones and gained valuable insights:

1. Accurate Detection: Our Leaf disease detection system has demonstrated a commendable level of accuracy in classifying diverse Leaf diseases. The use of advanced machine learning algorithms and deep neural networks has enabled us to reliably distinguish between different Leaf varieties.

2. Robust Performance: The system has proven its robustness by successfully handling a wide range of challenges, including variations in lighting, background clutter, and image quality. This robustness ensures its applicability in real-world scenarios.

3. User-Friendly Interface: For practical usability, we have incorporated a user-friendly interface that allows users to easily upload Leaf images and receive disease detection results swiftly and intuitively.

4. Contribution to Botanical Research: Our project has contributed valuable data and insights to the field of botanical research. By classifying Leaf diseases with precision, we have assisted researchers, botanists, and horticulturists in their efforts to study and conserve floral biodiversity.

5. Future Directions: While we celebrate our achievements, we acknowledge that there is room for improvement and expansion. Future directions for this project include enhancing the system's efficiency, incorporating more Leaf diseases, and exploring the use of additional data sources to further refine our detection capabilities.

6. Collaboration and Outreach: Collaboration with experts, researchers, and Leaf enthusiasts has enriched our project. We look forward to continuing to collaborate and share our findings with the wider scientific and botanical communities.

5.2 Limitations of the Work

A Leaf disease detection web application, like any software system, may have several limitations that can impact its functionality, performance, and user experience. Identifying these limitations is essential for improving the application and managing user expectations. Here are some common limitations of our Leaf disease detection web application:

1. Data Quality and Consistency:

Inaccurate or inconsistent data in the dataset, such as mislabelled images or incomplete information, can affect the application's reliability.

2. Environmental Factors:

The application may struggle to identify Leaves in images with poor lighting, complex backgrounds, or other environmental challenges.

3. Limited Language Support:

If the application only supports a limited number of languages or lacks localization features, it may not be accessible to users from diverse linguistic backgrounds.

4. Internet Connectivity:

Users may require a stable internet connection to use the web application. Offline functionality may be limited.

5. Device Compatibility:

Compatibility issues with certain web browsers or devices can restrict access for some users.

6. Maintenance and Updates:

Keeping the application up to date with the latest data and improvements in machine learning models requires ongoing maintenance.

7. User Input Quality:

The accuracy of detection is contingent on the quality of user-uploaded images. Blurry, low-resolution, or highly distorted images may lead to incorrect results.

8. User Understanding of Results:

Users may not always understand or agree with the application's detection results, leading to confusion or dissatisfaction.

5.3 Suggestion and Recommendations for Future Work

Creating an effective and successful Leaf disease detection web application involves considering various aspects, from user experience to technical functionality. Here are some suggestions and recommendations we received to enhance our Leaf disease detection web application:

1. User-Friendly Interface:

Design an intuitive and visually appealing user interface that makes it easy for users to upload images and receive detection results.

2. Language Support:

Offer multilingual support to cater to users from different language backgrounds. Provide clear instructions in multiple languages for better accessibility.

3. Privacy and Data Security:

Prioritize user privacy by clearly explaining data usage and ensuring secure handling of user-uploaded images. Comply with data protection regulations.

4. Continuous Learning:

Implement mechanisms to continuously update the disease detection model with new data and improvements in machine learning techniques.

5. Educative Content:

Include educational resources, such as articles, images, or descriptions, to help users learn more about different Leaf diseases.

6. Collaborations and Partnerships:

Collaborate with botanical gardens, universities, or research institutions to access authoritative Leaf data and gain credibility.

7. User Engagement:

Encourage user engagement through gamification, challenges, or incentives to promote the use of the application and user contributions.

8. Legal Compliance:

Ensure that the application complies with copyright and intellectual property laws when using images from external sources.

9. Community Building:

Foster a community around your application, allowing users to share their experiences, findings, and insights related to Leaf diseases.

Bibliography

[1] Varshney, R.K., et al. (2018). Agriculture 4.0: Broadening the horizons of crop improvement through genomics-assisted breeding. Cell, 175(2), 313-326.

[2] Hickey, L.T., et al. (2019). Breeding crops to feed 10 billion. Nature Biotechnology, 37(7), 744-754.

[3] Crossa, J., et al. (2017). Genomic selection in plant breeding: Methods, models, and perspectives. Trends in Plant Science, 22(11), 961-975.

[4] Rife, T.W., et al. (2019). CropSight: A scalable and open-source information management system for distributed plant breeding research. GigaScience, 8(2), giz003.

[5] Montesinos-López, O.A., et al. (2018). From plant breeding to precision agriculture for sustainable food production: A review. Agronomy for Sustainable Development, 38(4), 41.

[6] Heffner, E.L., et al. (2009). Plant breeding with genomic selection: Gain per unit time and cost. Crop Science, 49(6), 1785-1795.

[7] Endelman, J.B., & Jannink, J.L. (2012). Shrinkage estimation of the realized relationship matrix. G3: Genes, Genomes, Genetics, 2(11), 1405-1413.

[8] Rincent, R., et al. (2018). Recovering power in association mapping panels with variable levels of linkage disequilibrium. Genetics, 208(4), 1565-1580.

SOURCE CODE

Kotlin Code

```
package exmple.com.leafblightdetection
import android.app.Activity
import android.app.ProgressDialog
import android.content.DialogInterface
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.graphics.Color
import android.graphics.Matrix
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.os.Process
import android.provider.MediaStore
import android.view.Gravity
import android.view.MenuItem
import android.view.View
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.core.view.GravityCompat
import com.google.android.material.navigation.NavigationView
import com.infideap.drawerbehavior.AdvanceDrawerLayout
import kotlinx.android.synthetic.main.app_bar_main3.*
import java.io.IOException
class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {
   private lateinit var mCategorization: Categorization
   private lateinit var mBitmap: Bitmap
   private val mCameraRequestCode = 0
   private val mGalleryRequestCode = 2
   private val mInputSize = 224
   private val mModelPath = "plant_disease_model.tflite"
   private val mLabelPath = "plant_labels.txt"
   private val mSamplePath = "automn.jpg"
   lateinit var toolbar: Toolbar
   lateinit var drawer: AdvanceDrawerLayout
   lateinit var navigationView: NavigationView
   @RequiresApi(Build.VERSION_CODES.O)
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      navigationView =
         findViewById<View>(R.id.nav_view) as NavigationView
     toolbar =
         findViewById<Toolbar>(R.id.toolbar)
      setSupportActionBar(toolbar)
     drawer =
         findViewById<View>(R.id.drawer_layout) as AdvanceDrawerLayout

      drawer.setViewScale(GravityCompat.START, 0.9f) //set height scale for main view (0f to 1f)
      val toggle = ActionBarDrawerToggle(
         this, drawer, toolbar, 0, 0
      )
      drawer.addDrawerListener(toggle)
      toggle.syncState()
      navigationView.setNavigationItemSelectedListener(this)
      drawer.setViewElevation(
         GravityCompat.START,
         20f
      ) //set main view elevation when drawer open (dimension)
      drawer.setViewScrimColor(
         GravityCompat.START,
         Color.TRANSPARENT
      ) //set drawer overlay coloe (color)
      drawer.setDrawerElevation(GravityCompat.START, 20f) //set drawer elevation (dimension)
      drawer.setContrastThreshold(3f) //set maximum of contrast ratio between white text and
background color.
      drawer.setRadius(GravityCompat.START, 25f) //set end container's corner radius (dimension)
      drawer.useCustomBehavior(GravityCompat.START) //assign custom behavior for "Left" drawer
      drawer.useCustomBehavior(GravityCompat.END) //assign custom behavior for "Right" drawer
      //Set Name for user
      //Set Name for user
      val headerView = navigationView.getHeaderView(0)
      mCategorization = Categorization(assets, mModelPath, mLabelPath, mInputSize)
      resources.assets.open(mSamplePath).use {
         mBitmap = BitmapFactory.decodeStream(it)
         mBitmap = Bitmap.createScaledBitmap(mBitmap, mInputSize, mInputSize, true)
         mPhotoImageView.setImageBitmap(mBitmap)
      }
      mCameraButton.setOnClickListener {
         val callCameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
         startActivityForResult(callCameraIntent, mCameraRequestCode)
      }
      mGalleryButton.setOnClickListener {
         val callGalleryIntent = Intent(Intent.ACTION_PICK)
         callGalleryIntent.type = "image/*"
         startActivityForResult(callGalleryIntent, mGalleryRequestCode)
      }
      mDetectButton.setOnClickListener {
         val progressDialog = ProgressDialog(this@MainActivity)
```

```
            progressDialog.setTitle("Please Wait")
            progressDialog.setMessage("Wait there I do something...")
            progressDialog.show()
            val handler = Handler()
            handler.postDelayed(Runnable { progressDialog.dismiss()
                val results = mCategorization.recognizeImage(mBitmap).firstOrNull()
                mResultTextView.text= results?.title+"\n Confidence:"+results?.confidence
            }, 2000)

        }
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if(requestCode == mCameraRequestCode){
            //Considérons le cas de la caméra annulée
            if(resultCode == Activity.RESULT_OK && data != null) {
                mBitmap = data.extras!!.get("data") as Bitmap
                mBitmap = scaleImage(mBitmap)
                val toast = Toast.makeText(this, ("Image crop to: w= ${mBitmap.width} h=
${mBitmap.height}"), Toast.LENGTH_LONG)
                toast.setGravity(Gravity.BOTTOM, 0, 20)
                toast.show()
                mPhotoImageView.setImageBitmap(mBitmap)
                mResultTextView.text= "Your photo image set now."
            } else {
                Toast.makeText(this, "Camera cancel..", Toast.LENGTH_LONG).show()
            }
        } else if(requestCode == mGalleryRequestCode) {
            if (data != null) {
                val uri = data.data

                try {
                    mBitmap = MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
                } catch (e: IOException) {
                    e.printStackTrace()
                }
                println("Success!!!")
                mBitmap = scaleImage(mBitmap)
                mPhotoImageView.setImageBitmap(mBitmap)
            }
        } else {
            Toast.makeText(this, "Unrecognized request code", Toast.LENGTH_LONG).show()
        }
    }
    fun scaleImage(bitmap: Bitmap?): Bitmap {
        val orignalWidth = bitmap!!.width
        val originalHeight = bitmap.height
        val scaleWidth = mInputSize.toFloat() / orignalWidth
        val scaleHeight = mInputSize.toFloat() / originalHeight
        val matrix = Matrix()
        matrix.postScale(scaleWidth, scaleHeight)
        return Bitmap.createBitmap(bitmap, 0, 0, orignalWidth, originalHeight, matrix, true)
    }
    override fun onNavigationItemSelected(item: MenuItem): Boolean {
val drawer =
            findViewById<View>(R.id.drawer_layout) as AdvanceDrawerLayout
        when (item.itemId) {
            R.id.remdy -> {
                val intent = Intent(this@MainActivity,Common_Remedies::class.java)
                startActivity(intent)
            }
        }
        drawer.closeDrawer(GravityCompat.START)
        return true
    }
    override fun onBackPressed() {
        val alertDialogBuilder: android.app.AlertDialog.Builder = android.app.AlertDialog.Builder(this)
        alertDialogBuilder.setTitle("Exit Application?")
        alertDialogBuilder
            .setMessage("Click yes to exit!")
            .setCancelable(false)
            .setPositiveButton("Yes",
                DialogInterface.OnClickListener { dialog, id ->
                    moveTaskToBack(true)
                    Process.killProcess(Process.myPid())
                    System.exit(1)
                })
            .setNegativeButton("No",
                DialogInterface.OnClickListener { dialog, id -> dialog.cancel() })
        val alertDialog: android.app.AlertDialog? = alertDialogBuilder.create()
        alertDialog?.show()
    }
}
```

```
package exmple.com.leafblightdetection
import android.content.DialogInterface
import android.content.Intent
import android.os.Bundle
import android.os.Process
import android.os.Process.myPid
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

import kotlinx.android.synthetic.main.app_bar_main3.*

class Common_Remedies : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_common__remdies)
       // val actionBar = supportActionBar
        //actionBar!!.title = "Remedy"
        val actionbar = supportActionBar
        //set actionbar title
        actionbar!!.title = "Remedy"
        //set back button
    }
    override fun onBackPressed() {
        val alertDialogBuilder: android.app.AlertDialog.Builder = android.app.AlertDialog.Builder(this)
        alertDialogBuilder.setTitle("Exit Application?")
        alertDialogBuilder
            .setMessage("Click yes to exit!")
            .setCancelable(false)
            .setPositiveButton("Yes",
                DialogInterface.OnClickListener { dialog, id ->
                    moveTaskToBack(true)
                    Process.killProcess(myPid())
                    System.exit(1)
                })
            .setNegativeButton("No",
                DialogInterface.OnClickListener { dialog, id -> dialog.cancel() })
        val alertDialog: android.app.AlertDialog? = alertDialogBuilder.create()
        alertDialog?.show()
    }
}
```

XML CODE

```
<?xml version="1.0" encoding="utf-8"?>
<com.infideap.drawerbehavior.AdvanceDrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:openDrawer="start"
    android:background="@color/colorPrimary"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">
  <include
      layout="@layout/app_bar_main3"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
  <com.google.android.material.navigation.NavigationView
      android:id="@+id/nav_view"
      app:menu="@menu/menu_items"
      android:fitsSystemWindows="true"
      android:background="@color/colorPrimary"
      android:layout_gravity="start"
      android:layout_width="wrap_content"
      app:elevation="0dp"
      app:itemIconPadding="10dp"
      app:itemIconTint="@android:color/white"
      app:itemTextColor="@android:color/white"
      android:layout_height="match_parent"
      app:headerLayout="@layout/nav_header_main3"/>
```

```
</com.infideap.drawerbehavior.AdvanceDrawerLayout>
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
  <com.google.android.material.appbar.AppBarLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:theme="@style/AppTheme.AppBarOverlay">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />
  </com.google.android.material.appbar.AppBarLayout>
    <androidx.cardview.widget.CardView
        android:layout_marginTop="48dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/card_viewe"
        android:elevation="3dp"
        android:layout_gravity="center"
        app:cardCornerRadius="10dp">
      <androidx.constraintlayout.widget.ConstraintLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent"
  android:background="@drawable/card_viewe">
<Button
    android:id="@+id/mCameraButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/custom_button"
    android:padding="13dp"
    android:text="@string/buttonTakePhoto"
    android:textColor="#FFFFFF"
    app:layout_constraintBottom_toTopOf="@+id/mResultTextView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.873"
    app:layout_constraintStart_toEndOf="@+id/mGalleryButton"
    app:layout_constraintTop_toBottomOf="@+id/mPhotoImageView" />
<Button
    android:id="@+id/mGalleryButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="22dp"
    android:background="@drawable/custom_button"
    android:padding="13dp"
    android:text="@string/buttonSelectPhoto"
    android:textColor="#FFFFFF"
    app:layout_constraintBottom_toTopOf="@+id/mResultTextView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/mPhotoImageView" />
<ImageView
    android:id="@+id/mPhotoImageView"
    android:layout_width="270dp"
    android:layout_height="290dp"
    android:layout_marginTop="78dp"
```

```
        android:background="@drawable/img_v"
        android:contentDescription="@string/descriptionImage"
        android:scaleType="fitXY"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/automn" />
    <Button
        android:id="@+id/mDetectButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:background="@drawable/custom_button2"
        android:padding="13dp"
        android:text="@string/buttonDiagnose"
        android:textColor="#FFFFFF"
        app:layout_constraintBottom_toTopOf="@+id/mResultTextView"
        app:layout_constraintEnd_toStartOf="@+id/mCameraButton"
        app:layout_constraintHorizontal_bias="0.52"
        app:layout_constraintStart_toEndOf="@+id/mGalleryButton"
        app:layout_constraintTop_toBottomOf="@+id/mPhotoImageView"
        app:layout_constraintVertical_bias="0.352" />
    <TextView
        android:id="@+id/mResultTextView"
        android:layout_width="301dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="136dp"
        android:shadowColor="@android:color/black"
        android:text="@string/defaultImage"
        android:textAlignment="center"
```

```
            android:textColor="@android:color/holo_red_light"
            android:textSize="20sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/mPhotoImageView" />
      </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```