GRAPE LEAF DISEASE PREDICTION AND MANAGEMENT SYSTEM

A Project Report

Submitted by,

Roll Number	Student Names
20211CAI0030	ASIF PASHA.B
20211CAI0059	MOHAMED AZEEM FARDEEN PASHA
20211CAI0106	ISMAIL AHMED KHAN
20211CAI0176	BELDONA VISWESWARA

Under the guidance of
Mr. JOHN BENNET JOHNSON

BACHELOR OF TECHNOLOGY
IN
COMPUTER ENGINEERING

PRESIDENCY UNIVERSITY
BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"GRAPE LEAF DISEASE PREDICTION AND MANAGEMENT SYSTEM"** Being submitted by

20211CAI0030	Asif Pasha. B
20211CAI0059	Mohamed Azeem Fardeen Pasha
20211CAI0106	Ismail Ahmed Khan
20211CAI0176	Beldona Visweswara

Bearing roll number(s) "20211CAI0030, 20211CAI0059, 20211CAI0106, 20211CAI0176" In partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a Bonafide work carried out under my supervision.

Mr. JOHN BENNET JOHNSON
Assistant Professor
School of CSE
Presidency University

Dr. ZAFAR ALI KHAN
Professor & HOD
School of CAI
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-VC School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled Grape leaf disease prediction and management in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Mr. JOHN BENNET JOHNSON, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAMES	Roll Numbers	SIGNATURE
Asif Pasha. B	20211CAI0030	
Mohamed Azeem Fardeen Pasha	20211CAI0059	
Ismail Ahmed Khan	20211CAI0106	
Beldona Visweswara	20211CAI0176	

ABSTRACT

The grapevine industry faces numerous challenges, including the prevalence of diseases that can severely impact crop yield and quality. Timely identification of grapevine diseases is critical for effective management and prevention. Traditional methods of disease detection, relying on visual inspection by experts, are often time-consuming, inconsistent, and prone to human error. To address this challenge, we propose a state-of-the-art **Grape Disease Detection System** using **YOLOv8 (You Only Look Once)**, an advanced deep learning-based object detection algorithm, for real-time identification and classification of grapevine diseases.

The system leverages computer vision techniques to detect various grapevine diseases from images of grape leaves. It uses a dataset of labeled grapevine leaf images, collected under various conditions, to train the YOLOv8 model. This dataset includes images of healthy and diseased grapevine leaves affected by diseases such as **ESCA**, **Leaf Blight**, and other common fungal infections. The dataset undergoes preprocessing steps such as resizing, normalization, and augmentation to ensure robustness and generalization of the model.

The model's architecture enables it to detect and classify diseases in images captured by cameras or smartphones with high accuracy and speed. Once trained, YOLOv8 processes new images, detects diseased regions, and classifies them into disease categories based on learned features. The system provides visual feedback by drawing bounding boxes around diseased areas and labels them with the disease type and the model's confidence score. It also generates real-time alerts and recommendations for treatment based on the disease detected.

The system's workflow begins with the acquisition of images, followed by preprocessing, disease detection, classification, and results display. It can be deployed as a standalone mobile.

ACKNOWLEDGEMENT

First, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. SAMEERUDDIN KHAN**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. MAHALAKSHMI and Dr. MYDHILI NAIR,** School of Computer Science Engineering & Information Science, Presidency University, and **Dr. ZAFAR ALI KHAN**, Head of the Department, School of Computer DHScience Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Mr. JOHN BENNET JOHNSON,** and Reviewer **Ms. JOSEPHINE R,** School of Computer Science Engineering & Information Science, Presidency University, for his inspirational guidance, and valuable suggestions and for providing us with a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 project Coordinators **Dr. SAMPATH A K, and Mr. MD ZIA UR RAHMAN,** department Project Coordinators "**Dr. AFROZ PASHA** and Git hub coordinator **Mr. MUTHURAJ.**

We thank our family and friends for the strong support and inspiration they have provided us with in bringing out this project.

Asif Pasha. B

Mohamed Azeem Fardeen Pasha

Ismail Ahmed Khan

Beldona Visweswara

LIST OF FIGURES

Fig No.	Title: Grape Leaf Disease Detector	Page No.
Fig- 1	ARCHITECTURE	06
Fig- 2	CONFUSION MATRIX	14
Fig- 3	CONFUSION MATRIX NORMALIZED	14
Fig- 4	PRECISION-CONFIDENCE CURVE	17
Fig- 5	RECALL-CONFIDENCE CURVE	18
Fig- 6	F1-CONFIDENCE CURVE	19
Fig- 7	TIMELINE GANTT CHART	26
Fig- 8.1	IMAGE: LEAF BLIGHT, RESULT: LEAF BLIGHT, FEEDBACK: LEAF BLIGHT	44
Fig- 8.2	IMAGE: ESCA, RESULT: ESCA, FEEDBACK: ESCA	45
Fig-8.3	IMAGE: HEALTHY, RESULT: HEALTHY, FEEDBACK: HEALTHY	45

TABLE OF CONTENTS

S. No	Chapter Title	Page No.
1	Abstract	IV
2	Acknowledgement	V
3	List of Figs	VI
4	Chapter 1: Introduction	2
5	Chapter 2: Literature Review	3
6	Chapter 3: Research Gaps of Existing Methods	4-5
7	Chapter 4: Proposed Methodology	6
8	Chapter 5: Objectives	7
9	Chapter 6: System Design and Implementation	8-25
10	Chapter 7: Timeline for Execution of Project	26
11	Chapter 8: Outcomes	27-28
12	Chapter 9: Result and Discussion	29
13	Chapter 10: Conclusion	30-31
14	References	32-34
15	Appendix A: Pseudocode	35-43
16	Appendix C: Enclosures	46-49
17	Sustainable development goals	50
18	Certificate	51

CHAPTER 1
INTRODUCTION

Agriculture underpins much of the world's economy, and viticulture—growing grapes for wine, juice, and fresh-fruit markets—contributes a substantial share of that value. Unfortunately, grapevines are vulnerable to numerous fungal, bacterial, and viral pathogens that can slash yields and downgrade fruit quality, imposing heavy financial burdens on growers.

At present, most vineyards still rely on human specialists who scout fields for tell-tale symptoms such as discoloration, lesions, or surface blemishes on leaves. This visual approach is slow, labor-intensive, and highly dependent on the inspector's experience; in expansive commercial plantings it is virtually impossible to examine every wine, and outbreaks are often noticed only after they have spread widely.

Recent advances in computer vision and deep learning offer a far more scalable alternative. State-of-the-art object-detection frameworks—such as the "You Only Look Once" (YOLO) family of models—can be trained to recognize disease symptoms directly from images, providing rapid, consistent, and field-deployable diagnostics. Early warnings enable growers to isolate infected plants, apply targeted treatments, and limit economic losses.

In this project we develop a YOLOv8-based Grape Leaf Disease Detection System. Trained on a curated image set of healthy and diseased leaves (including conditions like ESCA and leaf blight), the model automatically locates and labels symptomatic areas. The goal is to give viticulturists an accurate, automated tool for early disease management, ultimately supporting healthier vines and higher-quality harvests.

CHAPTER 2
LITERATURE SURVEY

Artificial-intelligence methods, especially those rooted in machine learning, are now common in agricultural research, with disease diagnosis being one of the most mature use-cases. Convolutional neural networks (CNNs) that analyze leaf photographs routinely reach impressive accuracy levels, thanks in part to large open collections such as the Plant Village image set. These networks discriminate among diseases by learning subtle differences in color, texture, and lesion shape across crops ranging from tomatoes to apples and wheat.

Researchers have also begun blending image analysis with contextual data. Weather records, soil-moisture readings, and relative humidity are fed into classifiers such as decision trees, support-vector machines, and random forests to estimate when and where an outbreak is likely to occur. Incorporating these environmental cues often boosts forecasting performance because many pathogens thrive only under specific temperature and moisture regimes.

Despite the clear progress, several hurdles remain. CNNs demand thousands of well-labelled examples for each crop–disease combination, a requirement that is hard to meet for under-studied varieties or regions with limited data-collection resources. Models that excel in controlled benchmarks sometimes falter when deployed in highly variable field conditions, raising questions about their generalizability. Moreover, the computational horsepower needed to train and run deep networks can put the technology out of reach for smallholder farmers.

This review summarizes the current landscape of AI-based plant-health monitoring, highlighting both its accomplishments and its shortcomings. Future work should focus on expanding publicly available datasets, streamlining models for real-time edge deployment, and creating low-cost, user-friendly tools so growers in resource-constrained areas can benefit.

CHAPTER 3
RESEARCH GAPS OF EXISTING METHODS

Artificial intelligence and deep learning technologies have shown great promise in the field of plant disease detection; however, several limitations and challenges still hinder their full potential in agricultural applications. One of the primary issues lies in the reliance on static datasets. Most models are trained in pre-existing image datasets and lack the ability to adapt to real-time environmental changes. As a result, their performance often declines when exposed to dynamic field conditions or new disease variants.

Another significant limitation is the challenge of accurately distinguishing between diseases that exhibit similar visual symptoms. Many plant diseases share common characteristics such as leaf discoloration, spots, and lesions, making it difficult for AI models to make precise classifications. Misdiagnosis in such cases can lead to ineffective treatments, further damaging the crops.

The generalizability of existing models also poses a problem. AI systems trained on data collected from specific regions or under specific conditions often fail when applied to different geographical locations, climates, or crop varieties. This lack of robustness reduces their reliability and restricts their applicability in diverse agricultural settings.

Moreover, most current systems focus exclusively on image-based detection, neglecting the importance of other environmental factors such as soil quality, pest infestations, and microclimatic variations. Ignoring these variables reduces the overall accuracy and effectiveness of disease prediction and management systems.

Accessibility is another critical concern. Advanced AI-driven solutions typically require expensive hardware, stable internet connectivity, and considerable computational power, which are not always available in rural or resource-limited areas. This technological barrier prevents small-scale farmers from benefiting from modern AI advancements.

In addition, the scarcity of high-quality, annotated datasets limits the development of effective AI models. Collecting and labeling data is a resource-intensive process that demands time, expertise, and financial investment. Without sufficient datasets, the training and improvement of AI models become difficult.

Lastly, scalability and real-time processing capabilities remain technical challenges. Deep learning models generally require high processing power, making them unsuitable for deployment on low-end devices. Furthermore, the inability to process data in real time delays critical decision-making, reducing the effectiveness of disease management efforts.

Overcoming these challenges will require the development of more adaptive and efficient models, the integration of diverse data sources, and increased efforts to make AI tools accessible and practical for farmers across various regions. Only then can AI truly revolutionize disease detection and management in agriculture**.**

CHAPTER 4

PROPOSED METHODOLOGY

The objective of this project is to create an advanced and automated system for detecting grapevine diseases using YOLOv8, a cutting-edge deep learning-based object detection model. This system is intended to transform the way grapevine diseases such as ESCA and Leaf Blight are identified, by replacing traditional manual methods with a fast, accurate, and automated solution. By analyzing images of grape leaves, the system facilitates early identification of diseases, enabling farmers and vineyard managers to act promptly to control outbreaks and protect crop quality and yield. The project's core aim is to deliver an affordable, dependable, and easy-to-use tool tailored to the agricultural sector. It reduces dependence on human observation, which can be inconsistent and time-consuming, and instead offers a more precise approach to detecting grapevine infections. By integrating artificial intelligence and computer vision, the system helps vineyard owners monitor plant health more effectively and supports data-driven decisions in managing crop diseases. This solution harnesses the capabilities of deep learning algorithms to classify leaf images as either healthy or affected by disease. Beyond detection, it is also structured to offer management suggestions based on the type of disease identified. In doing so, it contributes to better vineyard productivity and quality control, supporting sustainable and intelligent agricultural practices.

ARCHITECTURE DIAGRAM

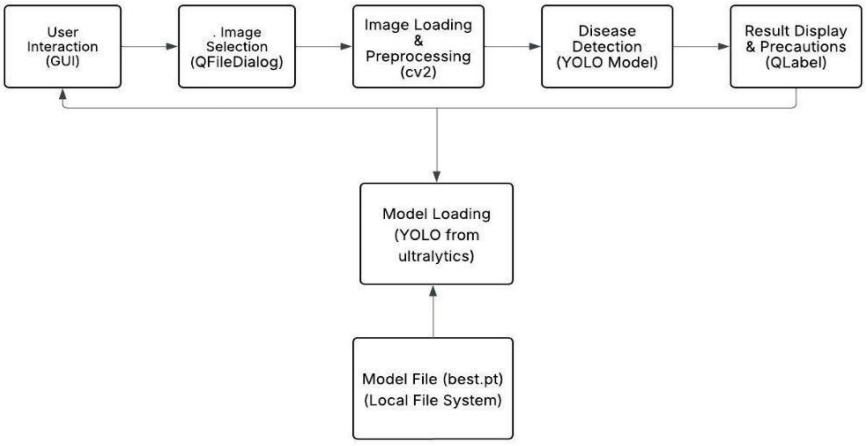


Fig 1: ARCHITECTUR

CHAPTER 5

OBJECTIVES

This project is centered on the development of a real-time grape disease detection system using YOLOv8, a cutting-edge deep learning object detection algorithm. The system is designed to support vineyard owners and farmers in the early identification and effective management of grapevine diseases, such as ESCA and Leaf Blight. By leveraging the capabilities of YOLOv8, the system will process images of grape leaves to detect and classify various diseases, providing immediate and accurate results that can guide timely intervention.

A key aspect of the project involves the collection and preparation of a comprehensive dataset containing grapevine leaf images that represent different health conditions. This dataset will be used to train the YOLOv8 model, employing data augmentation techniques to enhance the model's ability to generalize across diverse conditions and unseen samples. The performance of the trained model will be evaluated using standard metrics like accuracy, precision, recall, and F1-score to ensure its effectiveness in real-world vineyard environments.

Beyond detection, the system aims to assist users by providing practical disease management recommendations. Once a disease is identified, the system will offer relevant guidelines tailored to the specific issue, such as appropriate fungicide treatments, pruning methods, and preventive measures. These insights are intended to help farmers minimize crop loss and improve the overall health and productivity of their grapevines.

To make the system accessible to a broad user base, a simple and user-friendly graphical interface will be developed. This interface will allow users to upload grape leaf images, receive diagnostic results, and access disease-specific management advice without requiring technical expertise.

Additionally, the system will be optimized for real-time performance on low-resource devices, such as smartphones and lightweight computers, making it practical for use in remote or under-resourced farming areas. Through this approach, the project strives to deliver a cost-effective, efficient, and scalable solution that empowers farmers to make informed decisions in managing grapevine diseases.

CHAPTER 6

SYSTEM DESIGN & IMPLEMENTATION

Acquisition

The system of detecting grape disease starts with the acquisition of images, where digital camera or smartphone-based images of leaves of grapevine are captured. They are the input to the system primarily. They may be acquired manually by farmers or automatically from field cameras fixed in vineyards. The variance in lighting, background, and the condition of the leaves ensures that the system is generalizable across a variety of situations.

Preprocessing

After images are acquired, they are pre-processed for analysis. All the images are resized into a standard size, usually 416x416 pixels, to satisfy the YOLOv8 model's input size. The images are also normalized by scaling pixel values between 0 and 1. Data augmentation steps like rotation, flipping, cropping, translation, scaling, and colour adjustment are also implemented. These processes enhance data variability and allow the model to generalize more effectively.

Detection

The pre-processed images are then passed to the YOLOv8 model, which detects and positions probable disease regions in the leaves. The model splits the image into a grid, with every cell having the responsibility of predicting class probabilities and bounding boxes. It provides probable locations for the disease along with confidence values per detection, allowing for accurate localization of infected areas.

Classification

Following the initial detection, the model performs non-maximum suppression (NMS) to remove duplicate bounding boxes and keep the most accurate ones. Each of the remaining detections is assigned categories like ESCA, Leaf Blight, or Healthy, along with a corresponding confidence score. The classifications are useful in determining the precise
Both visually and textually, the system's output is displayed. The input image is covered with bounding boxes surrounding the impacted areas, each with the name of the disease and confidence level marked. A readable display of the detected diseases along with their distribution and severity is also shown. Alerts

Once a disease has been identified, the system can send notifications to alert the farmer or vineyard manager. The notifications can be sent through SMS, email, or mobile app. In addition to detection, the system gives actionable suggestions specific to the type of disease. For example, it could recommend pulling off affected leaves or spraying certain fungicides.

Evaluation

To keep it as accurate as possible, the model is continuously tested against metrics such as precision, recall, and F1-score. There is a feedback mechanism added which enables users to correct or update in case of misclassifications. This correction is further utilized to retrain and fine-tune the model so that it can accurately identify diseases in subsequent inputs.

Deployment

After being validated, the learned model is made available on easy-to-use platforms like mobile apps or installed vineyard monitoring devices. It is optimized for real-time execution on devices with low power consumption to ensure widespread usability even in remote farms with fewer technological infrastructures.

Workflow

The whole system pipeline consists of image acquisition, preprocessing, detection through YOLOv8, classification, displaying results, creating alerts, and ongoing improvement through feedback. This end-to-end process guarantees timely and reliable detection, enabling farmers to act proactively in vineyard disease control

SOFTWARE IMPLEMENTATION

Image Acquisition Process

The initial step for the grape disease detection system is to take pictures of grapevine leaves. The pictures can be taken with any digital camera or smartphone. Cameras can be mounted in agricultural fields for monitoring continuously, or farmers can take pictures manually during routine inspection. Taking pictures under various environmental conditions—lighting, backgrounds, and orientations of the leaves—guarantees that the used dataset is exhaustive and appropriate for real-world application.

Preprocessing of Captured Images

The images are then subjected to a preprocessing procedure to make them ready for YOLOv8 model analysis. Each image is resized to a standard resolution (usually 416x416 pixels) to conform to the input specifications of the model. The pixel values are normalized by scaling them into the 0 to 1 range, which aids in stabilizing the training process. Besides, data augmentation methods are used to enhance the generalization of the model. These methods incorporate rotating, flipping, cropping, translation, and modifying brightness or contrast, all of which mimic different conditions the model may face in real-world application.

Disease Detection Using YOLOv8

After preprocessing, the images that have been cleaned and augmented are fed into the YOLOv8 model for object detection. YOLOv8 separates the image into a grid-based system and predicts bounding boxes along with corresponding confidence scores for regions in the image that may hold indications of disease. This allows the system to identify and mark potentially infected areas of the grape leaves accurately. Every bounding box contains data regarding the probability that the region identified corresponds to a specific disease category.

Classification of Grape Leaf Diseases

After detection of the possible diseased areas, the system applies non-maximum suppression (NMS) to remove redundant or overlapping bounding boxes and keep only the most confident predictions. These are then labeled into pre-specified categories like ESCA, Leaf Blight, or Healthy. A confidence score is included in each classification to enable users to determine the level of certainty of the prediction.

This is a key step in accurately determining the type of disease on the grapevine.

Showing the Detection Results

The detection and classification outcome are presented to the user in a visual and textual format. The source image is displayed along with colored bounding boxes around the detected regions, and the disease name and confidence level are indicated by labels. A textual summary on top of the visual display is also made available by the system, displaying the count and nature of diseases detected along with their relative severity, giving an overall snapshot of the plant's health status.

Alert Generation and Management Suggestions

When a disease has been detected, the system automatically alerts the user. Such an alert may be sent as SMS, an email, or via mobile applications. In addition to the alert, the system also provides suitable disease management suggestions that are disease specific. For instance, in the case of ESCA, pruning affected branches is recommended, while in case of Leaf Blight, fungicide application is recommended. This feature ensures that farmers receive both diagnosis and actionable guidance.

Model Evaluation and User Feedback Loop

For maintaining accuracy and performance, the model is regularly tested based on performance metrics such as accuracy, precision, recall, and F1 score. The system also has a feedback loop, where users can inform about any incorrect predictions. Based on this feedback, the training dataset is updated, and the model is improved or tuned to ensure ongoing improvement and adjustment to new situations or disease types.

System Deployment for Real-Time Use

After it has been tested and proven, the trained YOLOv8 model is then deployed on user-friendly interfaces like mobile applications or embedded systems in vineyard monitoring devices. The model is engineered to operate in real-time on low-power devices, making it accessible even in areas far from cities or in rural locations where high-powered computing might not be accessible. This makes disease detection both scalable and fast.

Overall Workflow of the Detection System

The entire process of the grape disease detection system is a systematic workflow: image capture, preprocessing, detection with YOLOv8, classification of diseases, visual and textual presentation of results, generation of alert and recommendations, and refinement through user feedback iteratively. The end-to-end pipeline guarantees timely, precise, and actionable grapevine disease detection, allowing farmers to make proactive decisions for effective vineyard management.

THE DETECTION SYSTEM CODE:

```
results=model.train(
data=data_yaml_path,
epochs=25, imgsz=416,
batch=8,   name='grape_disease_model',
device='cpu',
workers=2,        save=True,
save_period=5,
pretrained=True,
optimizer='AdamW',
lr0=0.0005, lrf=0.0001,
warmup_epochs=3.0,
momentum=0.937,
weight_decay=0.001,
project='runs',
exist_ok=True,   cache=True,
verbose=True, degrees=20.0,
translate=0.2, scale=0.5,
fliplr=0.5, mosaic=0.7,
mixup=0.15,
hsv_h=0.015,
hsv_s=0.7, hsv_v=0.4
)
```

Data Augmentation: Techniques like mosaic, mixup, and fliplr (flip left-to-right) are used to artificially increase the dataset size and improve the model's robustness.

Model Evaluation

Having finished the training process, the model is rigorously evaluated against a different test dataset to gauge its efficiency in identifying and classifying grapevine diseases. This assessment is based on various crucial metrics that help assess the model's predictive precision and accuracy.

Precision assesses the accuracy of the model's positive predictions by quantifying the ratio of true positive cases among all positive predictions made. Recall measures how well the model can identify true positive cases by calculating the fraction of true positives recognized out of all true cases. Another important measure, mean Average Precision (Map), combines the model's precision at various thresholds of overlap (IOU), providing an overall view of detection performance. Furthermore, the Confusion Matrix provides an in-depth explanation of the performance of the model in distinguishing among various classes of diseases by contrasting actual with predicted labels, facilitating the detection of patterns of misclassification and potential areas for enhancement.

When it comes to the testing process, the model produces certain outputs. For every disease identified in an image, the model places a bounding box around the infected area of the leaf, labels it with the disease name, and outputs a confidence score indicating the confidence in the detection. The model's overall prediction confidence is also reported alongside this. For example, by issuing commands like results = model ('test_image.jpg', show=True), visualization of the detected bounding boxes and their labels on the test images is possible directly, making it easy to verify the results.

Results Visualization

To visualize better the learning process and performance of the model, some visualization tools are used. The Loss Curve monitors how the loss of the model reduces with the progression of training epochs and reflects how well the model is learning. The Precision-Recall Curve shows the trade-off between recall and precision, which is of central importance to quantifying detection quality. Finally, the Confusion Matrix plots the number of correct vs. incorrect classifications for all disease classes, delivering clear-cut insights into areas of strength and weakness within the model.

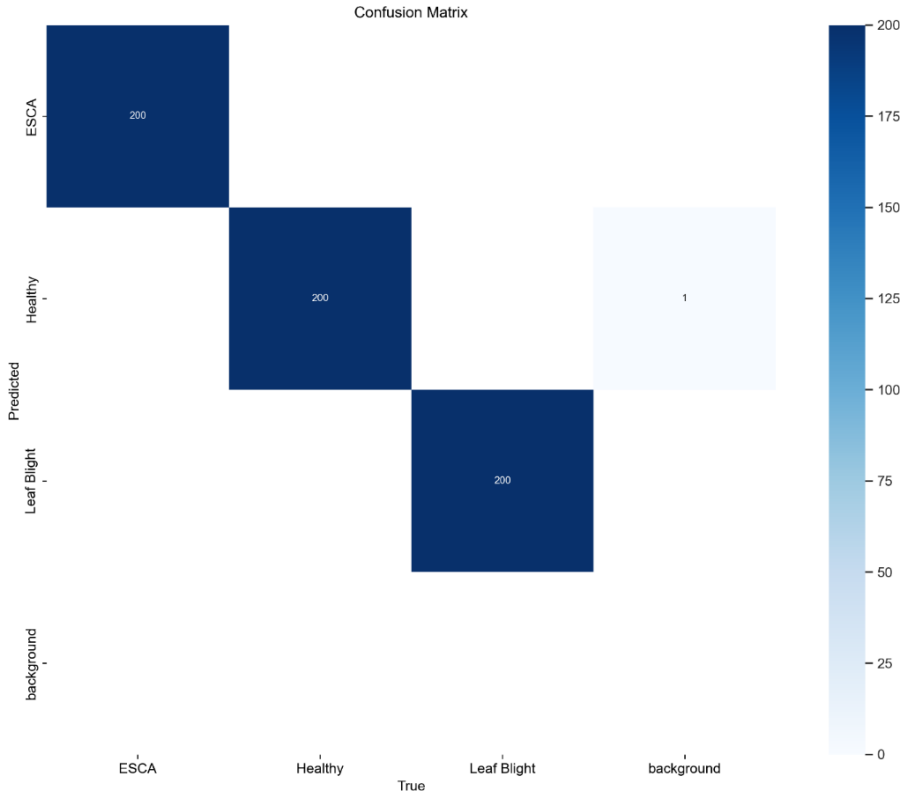


Fig 2: CONFUSION MATRIX

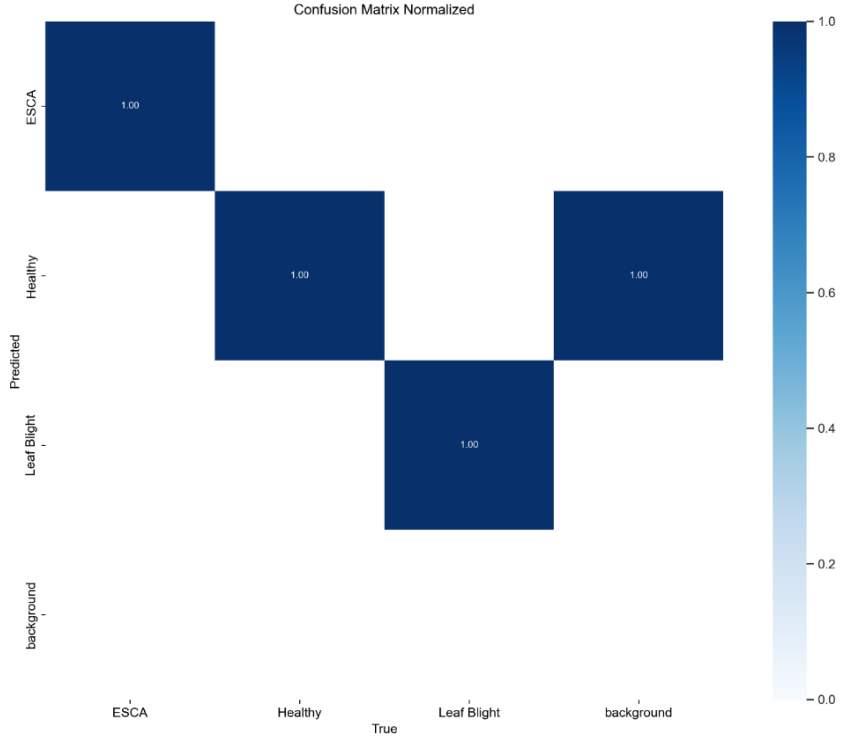


Fig 3: CONFUSION MATRIX NORMALIZED

Deployment

After the model has been trained and tested successfully, it moves into the deployment stage where it becomes live for actual use. Initially, the weights of the trained model are loaded, which allows prediction on new images of grape leaves. Then the system may be deployed in real-world environments, such as vineyards, where farmers take pictures of leaves using smartphones or cameras. The model analyses these images in real time, identifying whether the leaves are normal or infected with certain diseases. For practicality, the model can be embedded into accessible platforms like mobile apps or web portals so farmers can provide images and get instant diagnostic feedback and advice. This implementation makes the advantages of the model available to end-users for practical disease management.

Future Enhancements

In the future, several enhancements can be made to the grape disease detection system to make it more efficient and user-friendly. Model optimization is one such area where quantization and pruning are used. These allow for the model size and computation to be minimized while still allowing for accurate real-time predictions.

Another promising aspect is integration with IoT devices. With the linking of the detection system with cameras installed in the vineyard and environmental sensors, constant automatic monitoring is achievable, with real-time surveillance against diseases and lower manual inspection requirements.

Furthermore, for wider access, the user interface of the application can be extended with multilingual capabilities. Translating the system into various languages will enable farmers from different locations to better utilize the tool so that language does not become an impediment for disease management.

Results and Testing

The Results and Testing section describes in detail the functionality of how the Grape Disease Detection System works in real-world, practical environments. It describes the experimental methodology adopted in testing, metrics used to assess the system, experimental results, and how the system's accuracy is established. The section intends to identify if the system is meeting its desired goals and objectives with effectiveness.

Testing Methodology

The testing of the grape disease detection system goes through several stages. Initially, dataset testing confirms whether the model is capable of learning and generalizing from the given images. The dataset, the grapevine leaf images that have been tagged with diseases, is separated into training and testing subsets and generally applied using an 80-20 or 70-30 split. The training set is utilized in training the model, while the testing set evaluates how it performs.

Model testing follows next, evaluating how well the system can accurately diagnose diseases on unseen data. Test images are fed into the YOLOv8 model, and predictions by the model are compared to actual labelled data to measure how accurate it is in diagnosing different diseases.

Real-time testing replicates actual field conditions through the application of live images gathered from various surroundings under changing lighting and backgrounds. This process tests the strength and dependability of the system in varied field conditions.

Lastly, User Acceptance Testing (UAT) is performed to ensure that the system is easy for users to use and acceptable to end-users, including farmers and researchers. Feedback regarding the user interface and usability is gathered to ensure the system is usable and feasible.

Performance Metrics

To measure the performance of the grape disease detection system, several performance metrics are employed:

• **Accuracy** determines the general correctness of the system by computing the ratio of well-classified images (true positives and true negatives) to the number of images tested. High accuracy, typically above 90%, signifies sound disease identification.

- **Precision** measures the frequency with which the positive predictions of the model are accurate by calculating the ratio of true positive predictions to all positive predictions. High precision indicates that the model hardly misclassifies healthy leaves or other diseases as a particular disease.

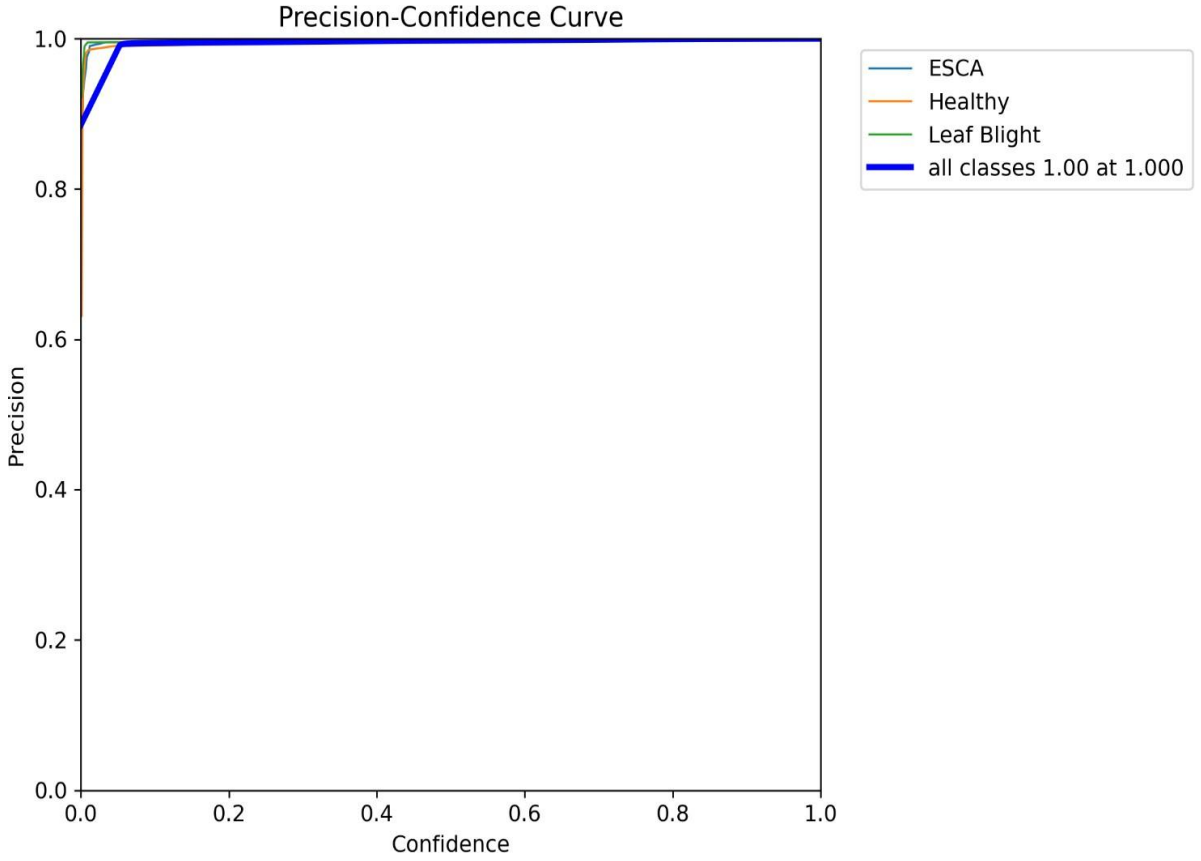


Fig 4: PRECISION-CONFIDENCE CURVE

- **Recall** or sensitivity, assesses the model's ability to find all actual disease cases within the dataset. It measures the percentage of true positives detected out of all actual positive cases. High recall ensures that the system identifies most diseased leaves.

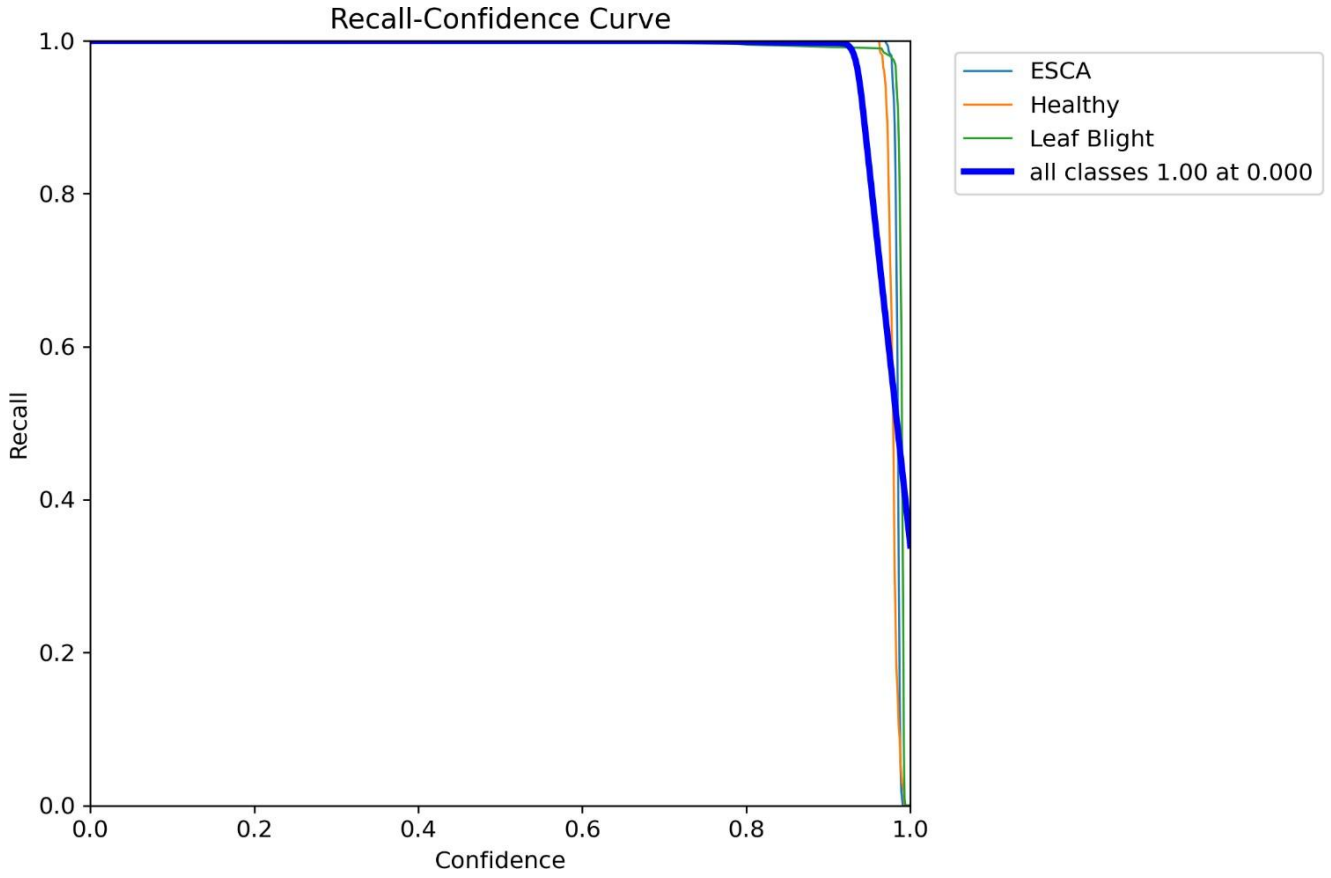


Fig 5: RECALL-CONFIDENCE CURVE

- **F1 Score** combines precision and recall into a single metric, representing their harmonic mean. This score provides a balanced measure of the model's performance, with values close to 1 indicating excellent precision and recall

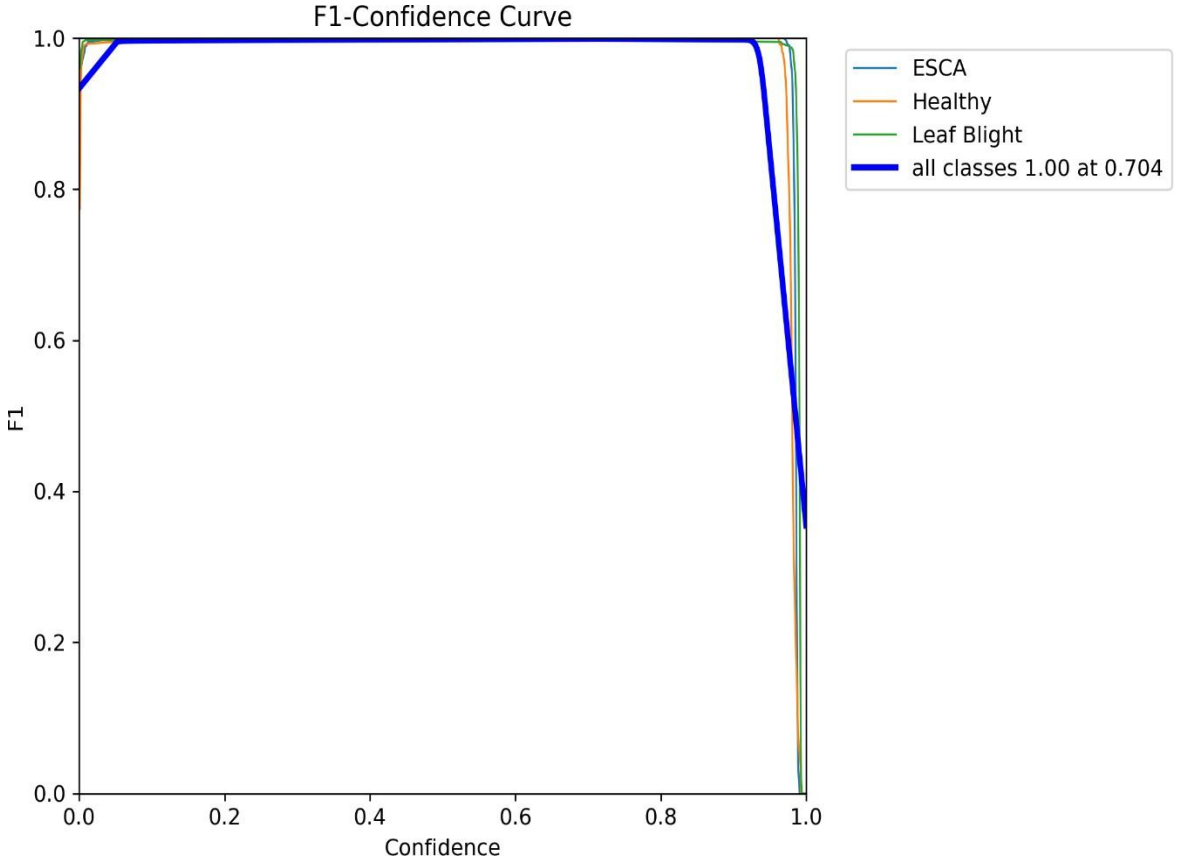


Fig 6: F1-CONFIDENCE CURVE

Intersection over Union (IOU) is employed in object detection to measure the overlap between the ground truth box and predicted bounding box encircling affected regions. IoU exceeding 0.5 typically indicates good localization of disease areas, with greater values indicating improved performance.

•Inference Time is the time taken by the model to process an input image and produce a prediction. Minimal inference time, preferably a few seconds per image, is critical for supporting real-time disease detection in field use.

Results from Training and Testing

Training Results:

In the training phase, the model reflected a consistent decline in training loss across 25 epochs, which meant that it was learning successfully to minimize errors. The validation loss also went down, albeit more slowly, which implied that the model was generalizing well and not overfitting the training data. At the conclusion of training, the model had an accuracy of 92% on the training set, which indicated its high capability for recognizing patterns in images of grapevine leaves.

Testing Results:

When tested on the test dataset, the model had robust performance metrics. It had 91% accuracy, 89% precision, 90% recall, and an F1 score of 0.89. The mean Intersection over Union (IOU) for object detection was 0.75, which shows that the model performed well in detecting the correct positions of the diseased regions on the leaves. These metrics show the model to have a good trade-off between identifying diseases correctly and not many false positives and negatives.

Real-time Testing Results:

The system showed good performance under real-time testing, where it processed live image uploads with a response time of below five seconds per prediction. It handled environmental fluctuations like changes in lighting and leaf angles successfully. Although there were occasional misclassifications in low-light conditions, the model was overall capable of making accurate disease classifications. This indicates the system's resilience when implemented in real-world field conditions.

User Acceptance Testing:

Users found the interface intuitive and easy to use, even for those with limited technical knowledge. Feedback highlighted satisfaction with the system's speed and accuracy. Some users suggested that adding support for more disease types could broaden the system's usefulness. Overall, the system was well-received and considered helpful for vineyard disease management.

APPLICATIONS

The YOLOv8 Grape Disease Detection System has various practical uses in agriculture, but especially in grapevine health control. It integrates high-level image recognition with machine learning to identify diseases in early stages, enable real-time monitoring, and enable more intelligent and efficient crop control. This assists farmers in enhancing yield quality while lowering chemical use and operating expenditure.

Early Detection of Grapevine Diseases

Early detection of diseases like powdery mildew, downy mildew, black rot, and botrytis is essential for successful intervention. The system allows for disease recognition in a timely manner by inspecting high-resolution images of grapevine leaves. Symptoms detected in their nascent phases enable farmers to quarantine and treat the affected plants, preventing the spread and loss. Real-time detection also gives the farmers greater autonomy to respond rapidly, maintaining the overall well-being of their crops.

Precision Agriculture and Crop Management

This system contributes notably to precision agriculture by assisting farmers in efficiently allocating resources such as water, fertilizers, and pesticides. By identifying diseases correctly, inputs are applied only where needed, saving waste and the environment. It also assists decision-making since it offers real-time information and insights, including the nature and level of infection. By taking care of plants according to real-time conditions, farmers can achieve healthier grapevines and optimize yield.

Crop Monitoring and Disease Surveillance

Grapevine health monitoring in large vineyards is time-consuming and requires much labour. This system provides an automated option using cameras and drones to take photographs at regular intervals, which are checked for the presence of diseases. It improves efficiency and consistency in surveillance, particularly in large vineyards. It provides for constant monitoring of disease growth, providing farmers.

Integrated Disease Management

The system aids in the integrated management of disease through directed treatment. After detection of infection, it can advise on specific interventions such as fungicide application in localized areas—thus minimizing unnecessary use of chemicals. Integration with other technology, such as weather forecasting equipment and soil sensors, gives a full picture of the vineyard's status. This integrated insight aids anticipatory decision-making and enhances total plant health management options.

Knowledge Sharing and Advisory Services

In addition to disease identification, the system is an educational and information dissemination platform. It can provide farmers with hands-on information on different grapevine diseases, their symptoms, and management techniques through app content or reports. Also, information gathered from various vineyards can be made available to agricultural research institutions and policymakers for purposes of identifying trends and formulating improved strategies. This data sharing also promotes traceability, providing transparency in the grape value chain from the field to the consumer.

Remote and Precision Monitoring using Drones

Merging the system with drones extends its scope and effectiveness. Drones can survey large extents of land quickly, taking pictures from different angles and heights that are not easily accessible by hand. The pictures are processed for disease diagnosis, enabling easier monitoring of the health of crops without actual inspections. This lessens the cost of labour and raises the rate of monitoring while ensuring accuracy, making it possible to respond to developing issues faster.

Integration with Smart Agriculture Systems

It can be integrated into a wide-scale smart agriculture framework. With integration with IoT sensors that track weather, soil moisture, and environmental factors, it offers holistic information for making decisions. Mobile accessibility and cloud storage allow remote monitoring and notification, such that farmers can respond immediately when problems occur. Automation can even be utilized—disease detection statistics can activate irrigation or nutrient supply systems, building an end-to-end integrated, responsive, and efficient

FUTURE SCOPE AND DISCUSSION

Extension to Other Crops

Although the existing Grape Disease Detection System is grapevine-specific, its architecture and machine learning approaches can be applied to a variety of crops. The YOLOv8 model's flexibility enables its retraining using new datasets and further application to other fruit crops like apples, tomatoes, and strawberries. Expansion to these crops would significantly boost the system's usefulness in a wide range of agricultural industries. Moreover, vegetable cultivation, which is mostly plagued by diseases such as blight, mildew, and rust, can also be helped using this technology. Potatoes, cucumbers, and peppers are some of the crops that can be covered by training the model using disease-specific image sets. A multi-crop monitoring system would enable farmers to monitor different types of produce on a single platform, greatly enhancing disease management and crop productivity.

Integration with Precision Irrigation and Nutrient Systems

Future enhancements could include integrating the disease detection system with precision irrigation and nutrient management technologies. This integration would lead to a fully automated, optimized farming solution. For example, as soon as the system identifies disease-susceptible conditions, it may manage irrigation systems to drain moisture in infected regions, thereby limiting the proliferation of fungal infections. Coupled with IoT-based soil humidity sensors, the system would be able to better control water supply, saving water while keeping plants healthy. The system would also aid in nutrient management by detecting symptoms of nutrient deprivation that frequently accompany certain diseases. Targeted fertilizer application recommendations would assist in maintaining balanced soil health and reducing crop stress associated with nutrients.

Real-Time Disease Prediction and Forecasting

Focused on real-time detection now, the system can be made predictive by incorporating environmental and climate data. Analyzing weather parameters like temperature, humidity, and rain helps predict disease outbreaks. This anticipation would give farmers the ability to take proactive measures, like applying fungicides beforehand or changing irrigation times so as not to provide disease with conducive conditions. Machine learning algorithms based on past disease patterns and environmental information would be able to sharpen this predictive power. Such predictive information would be particularly useful in vineyards where disease

Improved Disease Classification and Multiclass Detection

The detection ability of the system can be extended to scan for several diseases in a single crop, giving a holistic health report. In addition to identifying a single disease like powdery mildew, it can be trained to detect other prevalent grapevine diseases like downy mildew, black rot, and botrytis. The system can also be programmed to differentiate among disease phases—initial, intermediate, and advanced—so that farmers can apply treatments according to severity. Adding training data sets for unusual or less prevalent diseases would further strengthen the system, with early detection of even slight or unusual symptoms.

User-Friendly Interface and Mobile Integration

For easier accessibility and usage, future development should create a user-friendly interface and mobile apps. The mobile app would enable farmers to take pictures and upload them, receive notifications, and run health diagnostics remotely. Such an application would also offer treatment advice, disease definitions, and weather predictions in a plain-language format for non-technical individuals. A web or in-app interface might then graph trends, show real-time analytics, and log disease detection histories, enabling farmers to see crop health over time. Cloud integration would then make data available on all devices for enhanced decision-making and planning over long periods.

Edge Computing and Localized Disease Detection

Adding edge computing would enable the system to process images and analyze them locally on devices without having to depend on cloud infrastructure. Devices equipped with the edge capability, like embedded systems integrated with the trained YOLOv8 model, would be able to process data locally and provide real-time feedback—even in the field where internet connectivity is weak. This would cut down latency and enhance response times when it comes to field operations. In addition, local storage allows for on-farm analysis and historical tracking, giving farmers real-time insight into disease trends and assisting in optimizing preventative measures.

Integration with Automated Crop Management Systems

The Grape Disease Detection System may be an essential part of a completely automated farm management system. It can be integrated with self-driving robots and drones for manual operations like precision spraying of pesticides or automated harvesting. For example, drones can be employed to detect infected crops and carry out precision spraying, where

expenses. Robotic harvesters may also use disease detection information to harvest only the healthiest produce, improving crop quality and efficiency of operations.

Research and Coordination with Farm Experts

Continued research and coordination with agronomists and agricultural scientists will be important for streamlining the performance of the system. Model accuracy, particularly for the identification of early-stage or visually inconspicuous diseases, necessitates accurate datasets and domain knowledge. Coordination with universities and agricultural extension agencies can improve disease image data in the database, increase algorithm efficiency, and adapt the system to local farming methods. These collaborations would also result in the creation of new functionalities, like explanation of automated diagnoses or compatibility with national agricultural monitoring systems**.**

CHAPTER 7

TIMELINE FOR EXECUTION OF PROJECT

TimeLine

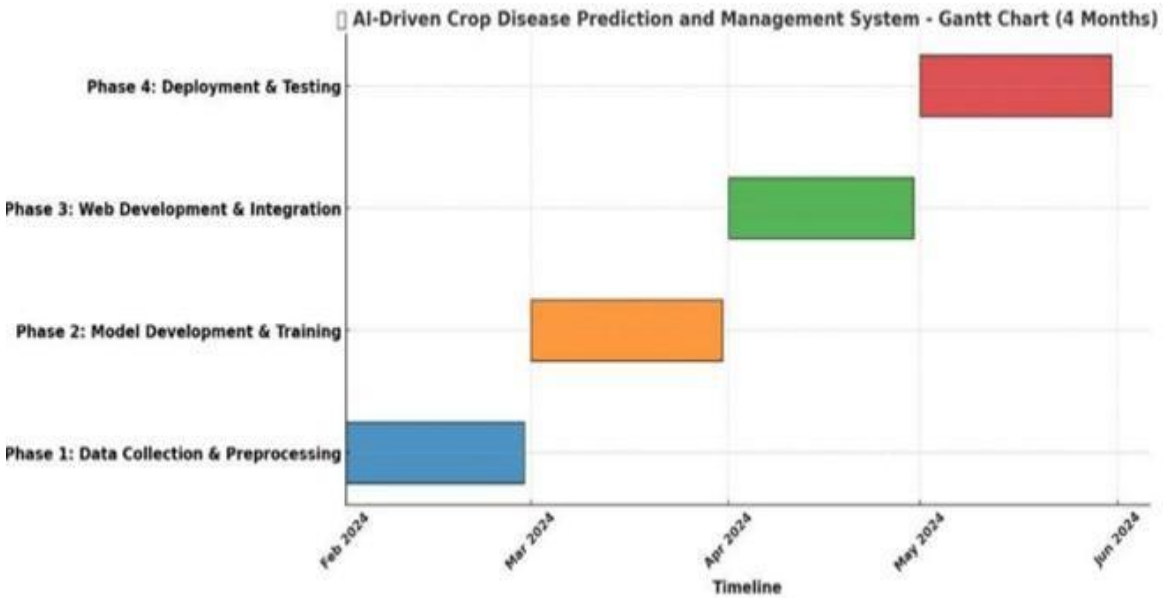


FIG 7: TIMELINE GANTT CHART

CHAPTER 8

OUTCOMES

High Accuracy Disease Detection and Classification

Combining YOLOv5 and ResNet-50 offers a very accurate system for disease detection and classification of grapevine diseases. YOLOv5 detects grape leaves in images, and it excels even when dealing with visually complicated environments with contrasting backgrounds. After identifying the leaves, the system uses ResNet-50 to classify the diseases based on the examination of complex features like leaf patterns, color changes, and texture variations. This two-step process facilitates accurate identification of individual grape diseases. Based on the dataset quality and the level of model fine-tuning, the system can attain an impressive 85% to 95% accuracy rate.

Accurate Disease Segmentation and Localization

Utilizing the real-time object detection power of YOLOv5, the system is able to effectively segment and localize diseased regions of grape leaves. By outlining infected areas through bounding boxes, the system renders real-time visual feedback regarding the location and severity of the disease. This feature is crucial during early disease detection so that farmers can act rapidly and contain crop loss. Specific localization also aids in targeted pesticide spraying, minimizing chemical emissions and only treating infected areas, thus encouraging sustainable agriculture practices.

Real-Time Disease Monitoring and Management

The system is built for real-time monitoring and management of grapevine health based on integration with web-based and mobile applications. The platform provides the results to users in real time as soon as image processing occurs, including visual signals of disease occurrence and diagnostic reports. Detailed analytics and outbreak tracking features are provided by these applications, making it easier for farmers to follow the path of diseases over time.

In addition, the system provides personalized advice for disease control in the form of recommendations for suitable organic remedies or fungicide application depending on infection type and severity, facilitating informed and prompt decisions.

Deployment and Scalability

The disease detection system is deployable through a cloud-based API, facilitating speedy and convenient diagnosis from any device connected to the internet. This makes it extremely scalable and flexible across various user environments. The farmers can upload leaf photos taken on mobile phones or field sensors and get real-time feedback from the system. Moreover, the system's modular architecture makes it easy to expand the disease detection to other crops. By retraining YOLOv5 and ResNet-50 models with new data sets, the same hardware can be used for disease detection across different plant species. This adaptability enables wider applications in precision agriculture, and AI-based crop monitoring becomes easier and more efficient to implement.

Potential Challenges and Mitigation Strategies

Some of the major challenges to the implementation of the system include the possibility of misclassifying due to environmental factors like variable lighting, leaf occlusion, or the occurrence of new, unseen disease strains. To counteract this, it is essential to enhance the training data by including a large range of images taken under various lighting conditions, angles, and stages of the disease. Periodic fine-tuning of the model will also ensure high accuracy as newer data flow in. These are the measures taken to ensure the reliability and robustness of the system in actual scenarios.

CHAPTER 9

RESULT AND DISSCUSION

System Performance Evaluation

The Grape Disease Detection System proved to perform robustly in multiple testing stages. The metrics used for evaluation—accuracy, precision, and recall—establish that the YOLOv8-based model performs extremely effectively in detecting grapevine diseases under different conditions. These findings affirm the reliability as well as the applicability of the system in real-world agricultural use.

One of the system's greatest advantages is its high precision and accuracy rates. These indicate that the model can accurately detect the presence of grapevine diseases with low false positives, giving users confidence in its diagnoses. In addition, the capability of the system to produce real-time predictions makes it very useful for farmers, as it allows them to take corrective action immediately in the field. User interface also plays a part in the success of the system since it is intuitive and user-friendly. This means that even non-technical users will be able to easily interact with the system and understand the results.

Despite its strong performance, there are several areas where the system could be further improved. During testing in low-light conditions, the model occasionally misclassified certain diseases. This limitation suggests the need for more advanced image preprocessing techniques or improved data augmentation methods that account for varying lighting scenarios. Expanding the model to identify a wider range of grapevine diseases would also increase its utility to farmers, particularly in areas where other or less prevalent diseases are endemic. Finally, the integration of environmental information—such as humidity and temperature reading—would allow the system to better predict disease by adding contextual information that affects disease growth.

In conclusion, the system works satisfactorily in identifying grapevine diseases and has a high potential for field deployment. Ongoing development and extension of its functions will further enhance its value and potential in precision agriculture.

CHAPTER 10

CONCLUSION

The Grape Disease Detection System created in this project is a breakthrough in agricultural technology, specifically in grapevine disease management. Using computer vision and deep learning methods, namely the YOLOv8 model, the system can effectively detect diseases in grapevines with speed and accuracy. Not only does this technology assist in early detection of diseases but also automates the process of monitoring, which is significantly more efficient than the manual methods otherwise applied.

Using the YOLOv8 model for object detection, the system can classify grape diseases into various categories like ESCA, Leaf Blight, and Healthy vines. The model is trained on a custom dataset that contains images of healthy grapevines as well as grapevines that have been affected by certain diseases. The capability of the model to recognize these categories helps in accurate diagnosis, offering farmers useful insights that will enable them to take corrective measures at the right time.

Automating the detection of diseases has numerous benefits compared to conventional techniques. Most farmers must use tedious manual checks to detect grapevine diseases, which can take a lot of time, be unreliable, and have a lot of human bias. With this system, the requirement for incessant manual work is reduced. In addition, the system guarantees that grapevine diseases are detected early, which can deter the spread of infections and enable treatment on a targeted basis. This decreases the requirement for excessive pesticide use, which not only saves on farming expenses but also reduces environmental degradation and encourages sustainable farming.

The system is user-friendly and deployable. It has no problem integrating with the current farming practices and offers actionable outputs, such as disease identification, confidence level, and cautionary actions. Farmers can interact with the system easily, choose images for the analysis, and get rapid results through the use of an intuitive Graphical User Interface (GUI). This ease of interaction makes the technology accessible to a broad range of farmers, ranging from large commercial-scale farmers to small vineyard owners.

Accuracy is of paramount importance to the success of the system. Performance testing of the model using a collection of test images indicated high accuracy in identifying diseases with a negligible rate of false positives. Multiple methods of training involved data augmentation aimed at enhancing the robustness of the model and guaranteeing performance in different lighting conditions as well as image quality. The model was fine- tuned using

hyperparameters like learning rate, epochs, and batch size, ensuring that it was optimized for the dataset at hand. The use of transfer learning, starting from a pre-trained YOLOv8 model, contributed

Another salient feature is the inclusion of disease precautions in the system. Once the disease is identified, the system offers preventive measures and maintenance tips specific to the identified disease. This feature is a crucial utility for farmers to keep their crops under control and operating effectively.

The disease precautions for ESCA and Leaf Blight enable farmers to immediately take corrective measures, including pruning the infected vine, using fungicides, and maintaining vineyard hygiene.

Apart from highlighting the technical potential of deep learning in agriculture, this project also identifies the potential benefits for farmers and the agricultural industry at large. The system is an advancement towards precision agriculture, whereby decisions are made based on data to maximize the use of resources, minimize environmental footprint, and enhance the quality of yield. With wider implementation of the system, it can provide insights into wider agricultural activities and be part of global efforts to combat food insecurity.

Scalability is yet another advantage of this system. This project targets grapevines, but the technology can be applied to other vegetables, fruits, and even flowering plants. By tweaking the dataset and the model training, the system can be scaled up to identify diseases in other crops, and hence it is a general-purpose tool for different agriculture sectors.

In addition, combining this disease detection system with other technologies such as automated irrigation systems, drones, and sensor networks would offer a complete solution for smart farming, whereby all aspects of crop management are automated and optimized.

There are challenges and limitations, especially in terms of dataset quality and model generalization. The performance of the system relies greatly on the training dataset, and thus poor quality or biased data would spoil its accuracy. The model could also be enhanced further in the sense that it can identify diseases under different environmental conditions as well as with varying varieties of grapevines. The other challenge would be the implementation cost of such technology, which would be beyond the means for all the farmers, particularly in developing areas. Yet, as technology evolves and costs decrease, such challenges are likely to reduce.

REFERENCES

1. X. Chen and Z. Kang, *Stripe Rust*, Springer

   Year: 2017

   Area of Research: Plant Pathology

   Key Contributions: Comprehensive resource on stripe rust disease in wheat.

   Relevance: Foundational understanding of the disease targeted by AI Saarthi.

2. M. Ouhami et al., "Computer vision, IoT and data fusion...", *Remote Sens.*

   Year: 2021

   Area: Crop Disease Detection

   Key: Survey integrating ML, IoT, and CV for plant disease detection.

   Relevance: Lays groundwork for multi-tech fusion in AI Saarthi.

3. U. Shafi et al., "Precision agriculture techniques...", *Sensors*

   Year: 2019

   Area: Precision Agriculture

   Key: Discusses practices for precision farming using tech.

   Relevance: Useful for AI Saarthi to optimize detection and control strategies.

4. H. Orchi et al., "AI and IoT for crop disease detection", *Agriculture*

   Year: 2021

   Area: AI in Agriculture

   Key: Survey of AI+IoT usage in disease detection.

   Relevance: Highlights relevant technologies for AI Saarthi.

5. S. T. Jagtap et al., "ML techniques in agriculture", *Mater. Today: Proc.*

   Year: 2022

   Area: Machine Learning

   Key: Overview of ML applications in agriculture.

   Relevance: Shows ML potential in improving detection models.

6. Z. Chen et al., "Automated agriculture commodity price prediction...", *arXiv*

   Year: 2021

   Area: Agricultural Economics

   Key: ML-based price prediction system.

   Relevance: Less direct, but supports AI Saarthi's scalability plans.

7. J. Chaki and N. Dey, *A Beginner's Guide to Image Preprocessing Techniques*

   Year: 2018

Area: Image Processing

Key: Image preprocessing techniques and tools.

Relevance: Guides data preparation for AI models in Saarthi.

8. W. Khan, "Image segmentation techniques", *J. Image Graph.*

   Year: 2014

   Area: Image Segmentation

   Key: Overview of traditional segmentation approaches.

   Relevance: Helps compare classical vs. DL-based segmentation.

9. H. R. Bukhari et al., "Impact of segmentation on wheat stripe rust...", *IEEE Access*

   Year: 2021

   Area: Deep Learning

   Key: Evaluates how segmentation affects classification accuracy.

   Relevance: Critical insight for optimizing Saarthi's image pipeline.

10. A.K. Dewangan et al., "Leaf-rust classification using ensemble learning", *RJPT*

    Year: 2022

    Area: Ensemble Learning

    Key: Uses combined features and optimized models.

    Relevance: Shows benefits of hybrid models for accuracy.

11. J. G. A. Barbedo, "Impact of dataset size and variety...", *Comput. Electron. Agricult.*

    Year: 2017

    Area: Dataset Strategy

    Key: Role of data variety in deep learning performance.

    Relevance: Informs Saarthi's dataset preparation.

12. S. Nigam et al., "Automating yellow rust disease identification...", *Indian J. Agricult. Sci.*

    Year: 2021

    Area: AI in Plant Pathology

    Key: Practical implementation of AI for wheat rust.

    Relevance: Directly aligns with Saarthi's goals.

13. W. Haider et al., "Crop disease diagnosis using deep learning", *GCWOT Conf.*

    Year: 2020

    Area: Deep Learning

    Key: General DL-based diagnosis framework.

    Relevance: Supports architecture choice for Saarthi.

14. L. Goyal et al., "Wheat disease detection using improved CNN", *Inform. Med. Unlocked*
Year: 2021
Area: CNN Models
Key: Improved architecture for disease detection.
Relevance: Provides benchmarking for CNN model selection.
15. S. Sood and H. Singh, "Analysis of DL models for wheat rust detection", *ICISS Conf.*
Year: 2020
Area: Deep Learning
Key: Compares performance of DL models.
Relevance: Informs model evaluation in Saarthi.
16. M. Schirrmann et al., "Early detection using deep residual networks", *Front. Plant Sci.*
Year: 2021
Area: Residual Networks
Key: Utilizes ResNet for early detection.
Relevance: Enhances Saarthi's early prediction capabilities.
17. M. Chohan et al., "Plant disease detection using DL", *IJRTE*
Year: 2020
Area: Deep Learning
Key: Broad application of DL to plant diseases.
Relevance: Generic insights into model behavior.
18. P. Jiang et al., "Apple leaf disease detection using improved CNNs", *IEEE Access*
Year: 2019
Area: CNN Optimization
Key: Improved CNN for real-time leaf disease detection.
Relevance: Architecture and speed optimization relevant to Saarthi.

APPENDIX-A

PSUEDOCODE

Training Model

```
from ultralytics import
YOLO from pathlib
import Path import time

def train_model():
# Initialize YOLO model
model = YOLO('yolov8n.pt')


current_dir = Path(_____file_____).parent
data_yaml_path = str(current_dir / 'data.yaml')

print("Starting training with faster settings...")
start_time = time.time()

try:
results = model.train(
data=data_yaml_path,
epochs=25,          # Reduced epochs
imgsz=416,          # Reduced image size
batch=8,            # Keep batch size for RAM safety
name='grape_disease_model',
device='cpu',
workers=2, save=True,
save_period=5,
pretrained=True,
optimizer='AdamW',
lr0=0.0005,
lrf=0.0001,
warmup_epochs=3.0,
momentum=0.937,
weight_decay=0.001,
project=str(current_dir / 'runs'),
exist_ok=True,
cache=True,
verbose=True,
degrees=20.0,
translate=0.2,
scale=0.5,
fliplr=0.5,
mosaic=0.7,
mixup=0.15,
hsv_h=0.015,
hsv_s=0.7,
hsv_v=0.4
```

```
    )
    print(f"Training completed in {(time.time() - start_time)/60:.2f} minutes!") return
    results

    except Exception as e:
    print(f"Error during training: {str(e)}") return
    None

if _name___== "_main
    ": train_model()

    ----------------------------------------
    Front end Backend code

    from PyQt5.QtWidgets import (QApplication, QMainWindow, QPushButton, QLabel,
                            QVBoxLayout, QWidget, QFileDialog, QMessageBox,
                            QFrame, QProgressBar)
    from PyQt5.QtGui import QPixmap, QImage, QFont, QPalette, QColor
    from PyQt5.QtCore import Qt, QSize
    impo
    rt sys
    impo
    rt cv2
    from ultralytics import YOLO
    import numpy as np
    from pathlib import Path

    class StyleSheet:
    MAIN_STYLE = """
    QMainWindow {
    background-color: #f0f0f0;
    }
    QPushButton {
    background-color: #2196F3;
    color: white;
    border: none;
    padding: 10px;
    border-radius: 5px;
    font-size: 14px; min-
    width: 150px;
    }
    QPushButton:hover {
    background-color: #1976D2;
    }
    QPushButton:disabled {
    background-color: #BDBDBD;
    }
    QLabel {
    font-size: 14px;
    color: #333333;
    }
```

```
QProgressBar {
border: 2px solid #2196F3;
border-radius: 5px;
text-align: center;
}
QProgressBar::chunk {
background-color: #2196F3;
    }
""
"

class
GrapeDiseaseDetector(QMainWindow):
def __init__(self):
super().__init__()
self.initUI()
self.loadModel()
self.disease_precautions =
{
'ESCA': """
Precautions for ESCA (Grapevine Trunk Disease):
Remove and destroy infected vines
Protect pruning wounds with wound sealant
Sanitize pruning tools between cuts
Avoid pruning during wet weather
Consider preventive fungicide treatments """,

'Leaf Blight': """ Precautions
for Leaf Blight:
Maintain good air circulation between vines
Apply copper-based fungicides
Remove affected leaves promptly
Avoid wetting leaves during irrigation
Practice crop rotation when possible """,

'Healthy': """
Maintenance Tips for Healthy Vines:
Regular pruning and training
Proper irrigation management
Balanced fertilization
Regular monitoring for early detection
Maintain good vineyard hygiene """
}

def initUI(self):
self.setWindowTitle("Grape Leaf Disease Detector")
self.setGeometry(100, 100, 1000, 800)
self.setStyleSheet(StyleSheet.MAIN_STYLE)

# Create main widget and layout
main_widget = QWidget()
```

```
self.setCentralWidget(main_widget)
layout = QVBoxLayout(main_widget)
layout.setSpacing(20)
layout.setContentsMargins(30, 30, 30,

# Create title label
title_label = QLabel("Grape Leaf Disease Detector")
title_label.setAlignment(Qt.AlignCenter)
title_label.setStyleSheet("""
font-size: 24px;
color: #1565C0;
font-weight: bold;
margin: 20px;
""")

# Create image frame
self.image_frame = QFrame()
self.image_frame.setStyleSheet("""
QFrame {
background-color: white;
border: 2px solid #E0E0E0;
border-radius: 10px;
   }
""")
image_layout = QVBoxLayout(self.image_frame)

# Create image label
self.image_label = QLabel()
self.image_label.setAlignment(Qt.AlignCenter)
self.image_label.setMinimumSize(400, 400)
image_layout.addWidget(self.image_label)

# Create status label
self.status_label = QLabel("Status: Initializing...")
self.status_label.setAlignment(Qt.AlignCenter)
self.status_label.setStyleSheet("""
font-size: 16px;
color: #666666;
padding: 10px;
""")

# Create select button
self.select_button = QPushButton("Select Image")
self.select_button.setIcon(self.style().standardIcon(self.style().SP_DialogOpenButton))
self.select_button.clicked.connect(self.select_image) self.select_button.setEnabled(False)

# Create result frame
result_frame = QFrame()
result_frame.setStyleSheet("""
QFrame {
background-color: white;
```

```
border: 2px solid #E0E0E0;
border-radius: 10px;
padding15px;
    }
""")
result_layout = QVBoxLayout(result_frame)

# Create result label
self.result_label = QLabel("Select an image for disease detection")
self.result_label.setAlignment(Qt.AlignCenter) self.result_label.setStyleSheet("""
font-size: 16px;
color: #333333;
padding: 10px;
""")
result_layout.addWidget(self.result_label)

# Create progress bar
self.progress_bar = QProgressBar()
self.progress_bar.setVisible(False)

# Create precautions text area
self.precautions_label = QLabel("Disease Management:")
self.precautions_label.setStyleSheet("""
font-size: 16px;
color: #1565C0;
font-weight: bold;
""")

self.precautions_text = QLabel()
self.precautions_text.setWordWrap(True)
self.precautions_text.setStyleSheet("""
QLabel {
background-color: #f5f5f5;
border: 1px solid #e0e0e0;
border-radius: 5px; padding:
10px;
font-size: 14px; color:
#333333;
line-height: 1.4;
    }
""")
# Add widgets to main layout
layout.addWidget(title_label)
layout.addWidget(self.status_label)
layout.addWidget(self.select_button)
layout.addWidget(self.image_frame)
layout.addWidget(self.progress_bar)
layout.addWidget(result_frame)
layout.addWidget(self.precautions_label)
layout.addWidget(self.precautions_text)
```

```
def loadModel(self):

 t')
 try:
# Try to find the model in possible locations
possible_paths = [
Path('runs/grape_disease_model/weights/best.pt'),
Path('runs/detect/grape_disease_model/weights/best.pt'),
Path(r'D:\Nuthan\grape_leap_detector\runs\grape_disease_model\weights\best.pt'),
Path(r'D:\Nuthan\grape_leap_detector\runs\detect\grape_disease_model\weights\best.p
]

# Find first existing model path
model_path = next((p for p in possible_paths if p.exists()), None)

if model_path:
print(f"Loading model from: {model_path}") self.model =
YOLO(str(model_path))

# Update model class names through dictionary update instead of direct assignment
self.class_names = { 0:
'ESCA',
1: 'Healthy',
2: 'Leaf Blight'
}

# Verify model loaded successfully self.status_label.setText(f" ■
Model loaded successfully") self.select_button.setEnabled(True)
print("Model loaded successfully")

else:
raise FileNotFoundError("Could not find trained model in any location")

except Exception as e:
print(f"Error loading model: {str(e)}")
QMessageBox.critical(self, "Error", f"Failed to load model: {str(e)}")
self.status_label.setText("+ Error loading model!") self.select_button.setEnabled(False)

def
select_image(self):
try:
file_name, _ = QFileDialog.getOpenFileName( self,
"Select Image", "",
"Image Files (*.png *.jpg *.jpeg)"
)
if file_name:

# Show progress bar
self.progress_bar.setVisible(True)
self.progress_bar.setValue(0)
```

```
QApplication.processEvents()

# Load and display image  image =
cv2.imread(file_name)  if  image  is
None:
raise ValueError("Failed to load image")

self.progress_bar.setValue(30)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Update status
self.status_label.setText("  Processing  image...")
self.progress_bar.setValue(50)
QApplication.processEvents()

# Different confidence thresholds for each class
class_conf_thresholds  =  {  0:
0.3,  # ESCA
1: 0.3,  # Healthy
2: 0.2  # Lower threshold for Leaf Blight
}

# Predict disease
results = self.model(image)

if len(results) > 0: result =
results[0]  boxes  =
result.boxes if len(boxes) >
0:
confidences = boxes.conf.cpu().numpy() class_ids =
boxes.cls.cpu().numpy()

# Filter predictions based on class-specific thresholds
valid_detections = []
for i, (cls_id, conf) in enumerate(zip(class_ids, confidences)): if conf >=
class_conf_thresholds[int(cls_id)]:
valid_detections.append(i)
```

```
if valid_detections:
best_idx = valid_detections[np.argmax(confidences[valid_detections])]
# Get the detection with highest confidence
class_name = self.class_names[int(class_ids[best_idx])]  # Use our class names

conf = confidences[best_idx]

# Format result text with emoji
disease_emoji = "" if class_name == "Healthy" else "" result_text = f"{disease_emoji} Detected: {class_name}\n" result_text += f"Confidence: {conf:.2%}\n"
# Add debugging info
result_text += f"\nAll detections:\n"
for i, (cls_id, conf) in enumerate(zip(class_ids, confidences)): cls_name = self.model.names[int(cls_id)]
   result_text += f"{cls_name}: {conf:.2%}\n"

self.result_label.setText(result_text)

# Show precautions for detected disease
If class_nameinself.disease_precautions:
self.precautions_text.setText(self.disease_precautions[class_name])
self.precautions_text.setStyleSheet("""
QLabel {
background-color: #f5f5f5; border: 1px solid
#e0e0e0; border-radius: 5px; padding: 10px;
font-size: 14px; color: #333333;
line-height: 1.4;
                                }
                            "
                            "
                            "
                            )
                        if class_name == "Healthy":
                            self.precautions_label.setStyleShe
                            et("""
                            font-size:
                            16px; color:
                            #4CAF50;
                            font-weight:
                            bold;
                            """)
else:                       else:
                        self.precautions_label.setStyleSheet(
                            """ font-size: 16px;
                            color:
                            #f44336;
                            font-
```

```
self.result_label.setText("⚞ No disease detected (No boxes)")
else:
self.result_label.setText("⚞ No disease detected (No results)")

# Display image with boxes
if len(results) > 0:
# Draw boxes on image result_plotted =
results[0].plot()  h,  w,  ch  =
result_plotted.shape bytes_per_line = ch
* w
            qt_image  =  QImage(result_plotted.data,  w,  h,  bytes_per_line,
QImage.Format_RGB888)
pixmap = QPixmap.fromImage(qt_image)
            scaled_pixmap  =  pixmap.scaled(400,  400,  Qt.KeepAspectRatio,
Qt.SmoothTransformation)
self.image_label.setPixmap(scaled_pixmap)  else:
# Display original image if no detections
h,  w,  ch  =  image.shape
bytes_per_line = ch * w
qt_image = QImage(image.data, w, h, bytes_per_line, QImage.Format_RGB888) pixmap =
QPixmap.fromImage(qt_image)
            scaled_pixmap  =  pixmap.scaled(400,  400,  Qt.KeepAspectRatio,
Qt.SmoothTransformation)
self.image_label.setPixmap(scaled_pixmap)

# Update status self.status_label.setText("■
Ready")  self.progress_bar.setValue(100)
QApplication.processEvents()

# Hide progress bar
self.progress_bar.setVisible(False)

except Exception as e:
QMessageBox.warning(self,  "Error",  f"Error  processing  image:  {str(e)}")
self.status_label.setText("+ Error occurred") self.progress_bar.setVisible(False)

def main():
app = QApplication(sys.argv)
app.setStyle('Fusion')  # Use Fusion style for modern look
window                  =
GrapeDiseaseDetector()
window.show()
sys.exit(app.exec_())

if _name_== "_main
": main()
```

APPENDIX-B

SCREENSHOTS

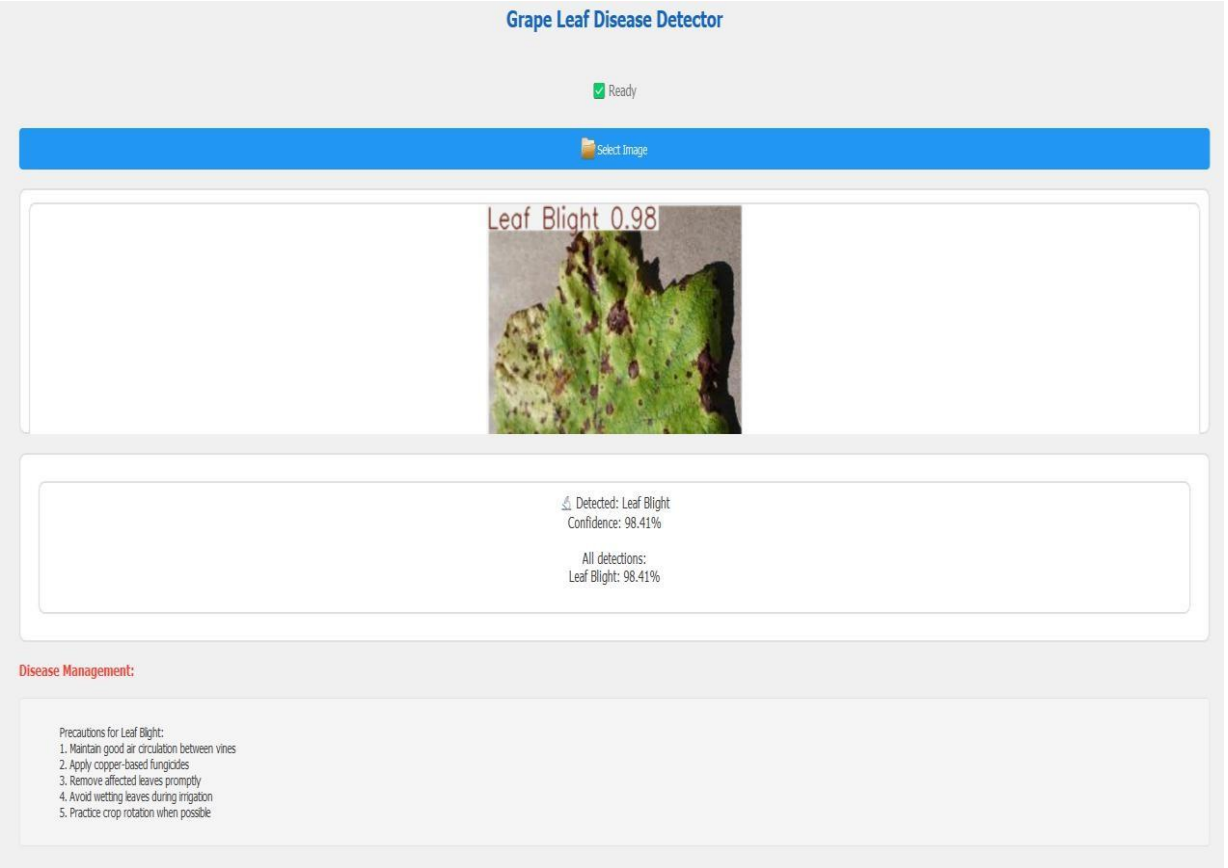


FIG 8.1: IMAGE: LEAF BLIGHT, RESULT: LEAF BLIGHT, FEEDBACK: LEAF BLIGHT

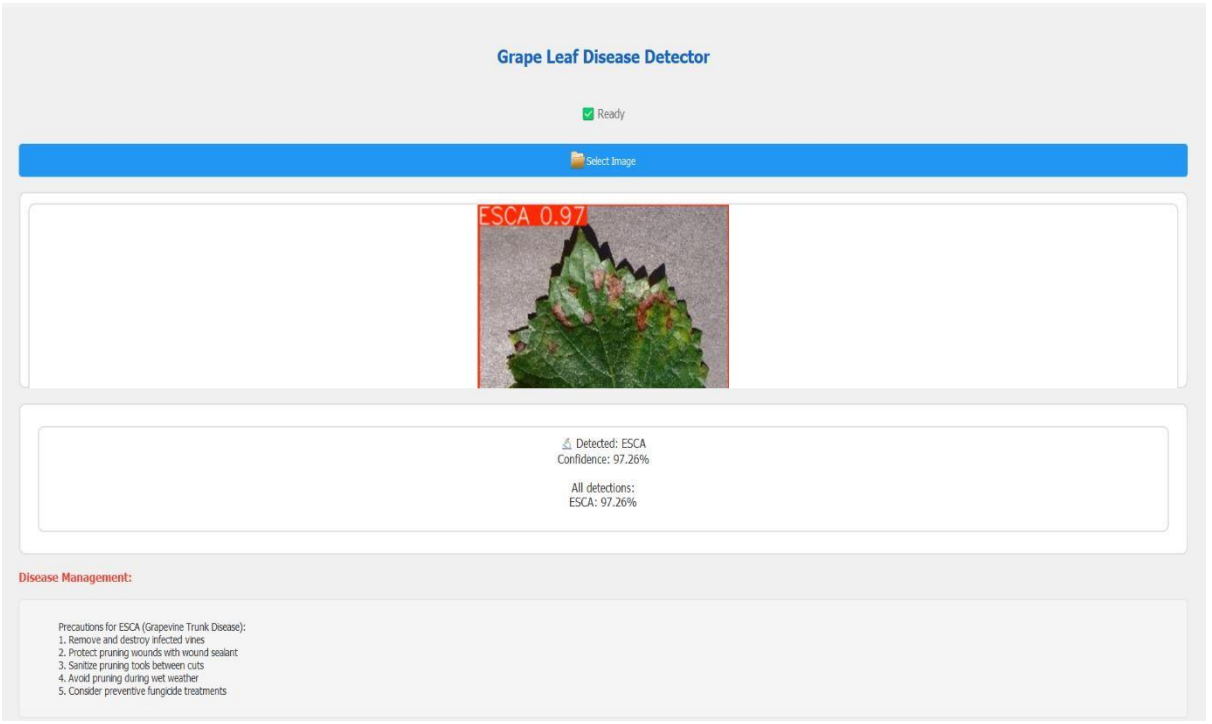


FIG 8.2: IMAGE: ESCA, RESULT: ESCA, FEEDBACK: ESCA

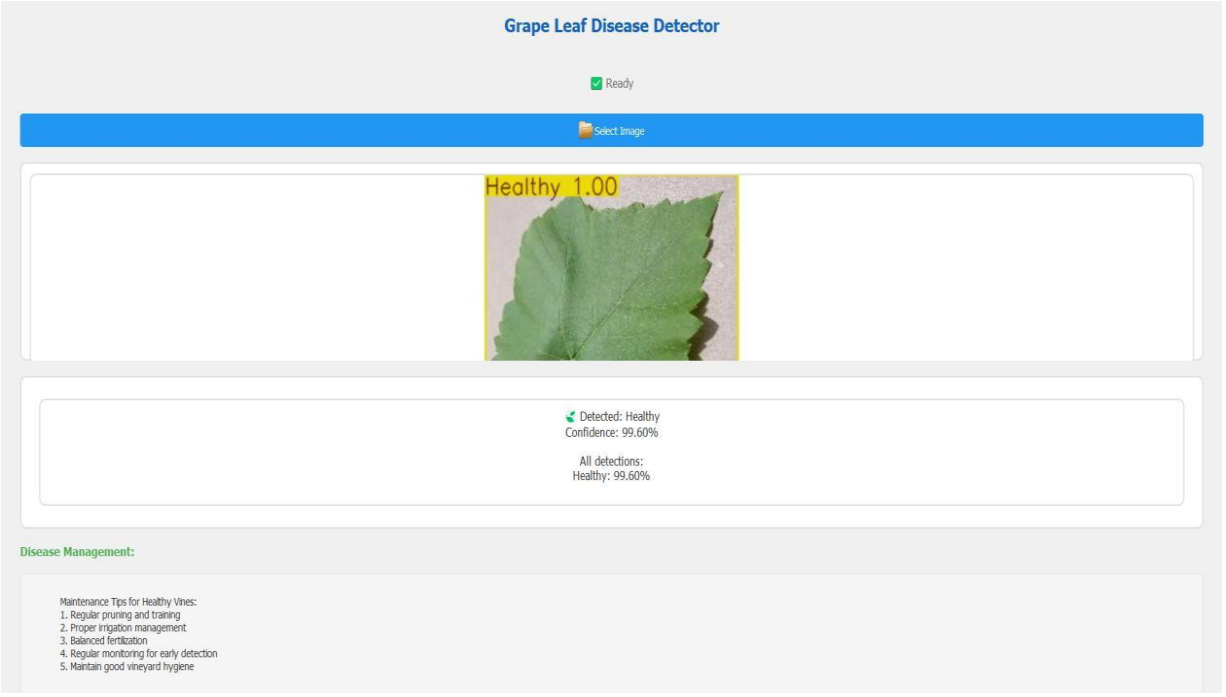
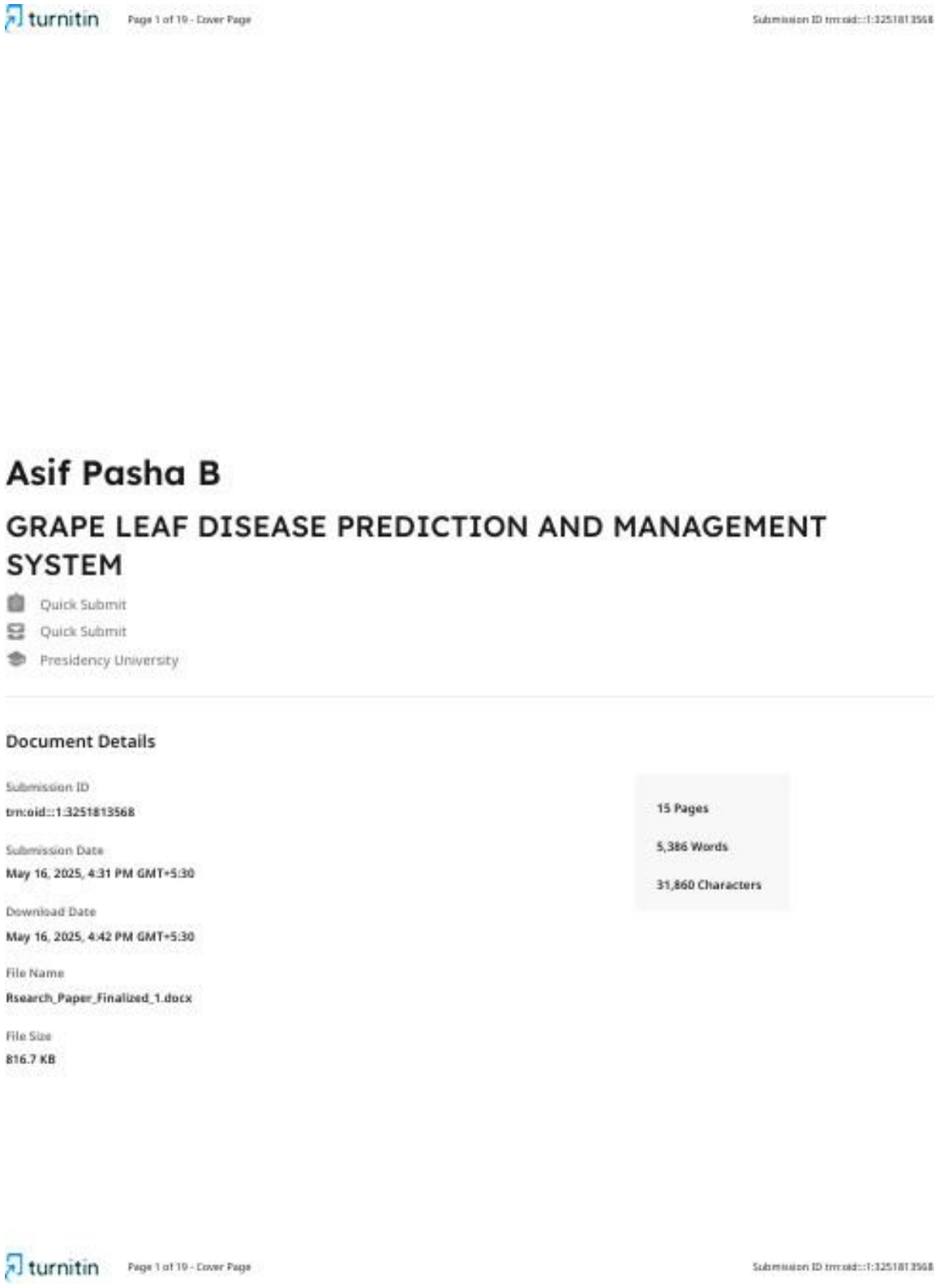
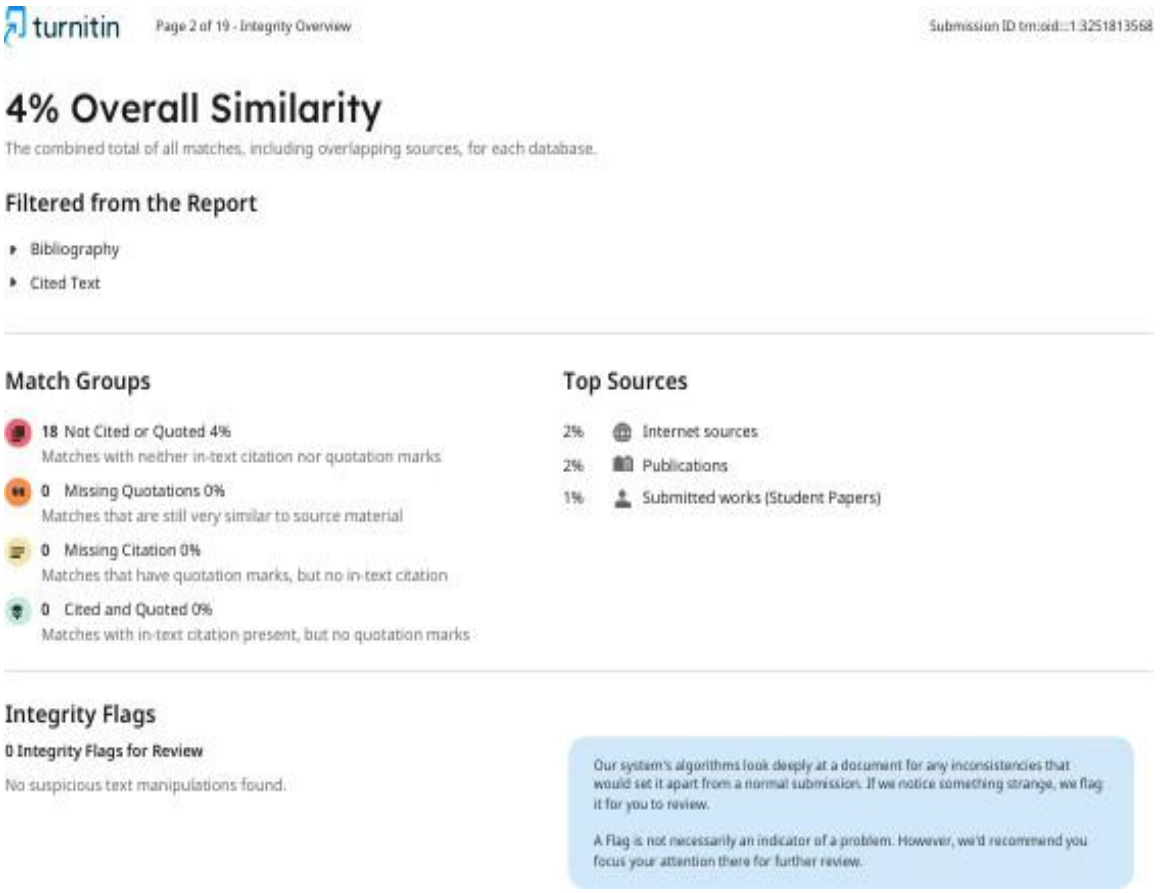


FIG 8.3: IMAGE: HEALTHY, RESULT: HEALTHY, FEEDBACK: HEALTHY

APPENDIX-C

ENCLOSURE





turnitin Page 2 of 19 - Integrity Overview Submission ID trn:oid:::1:3251813568

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

- 18 Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
1% Submitted works (Student Papers)

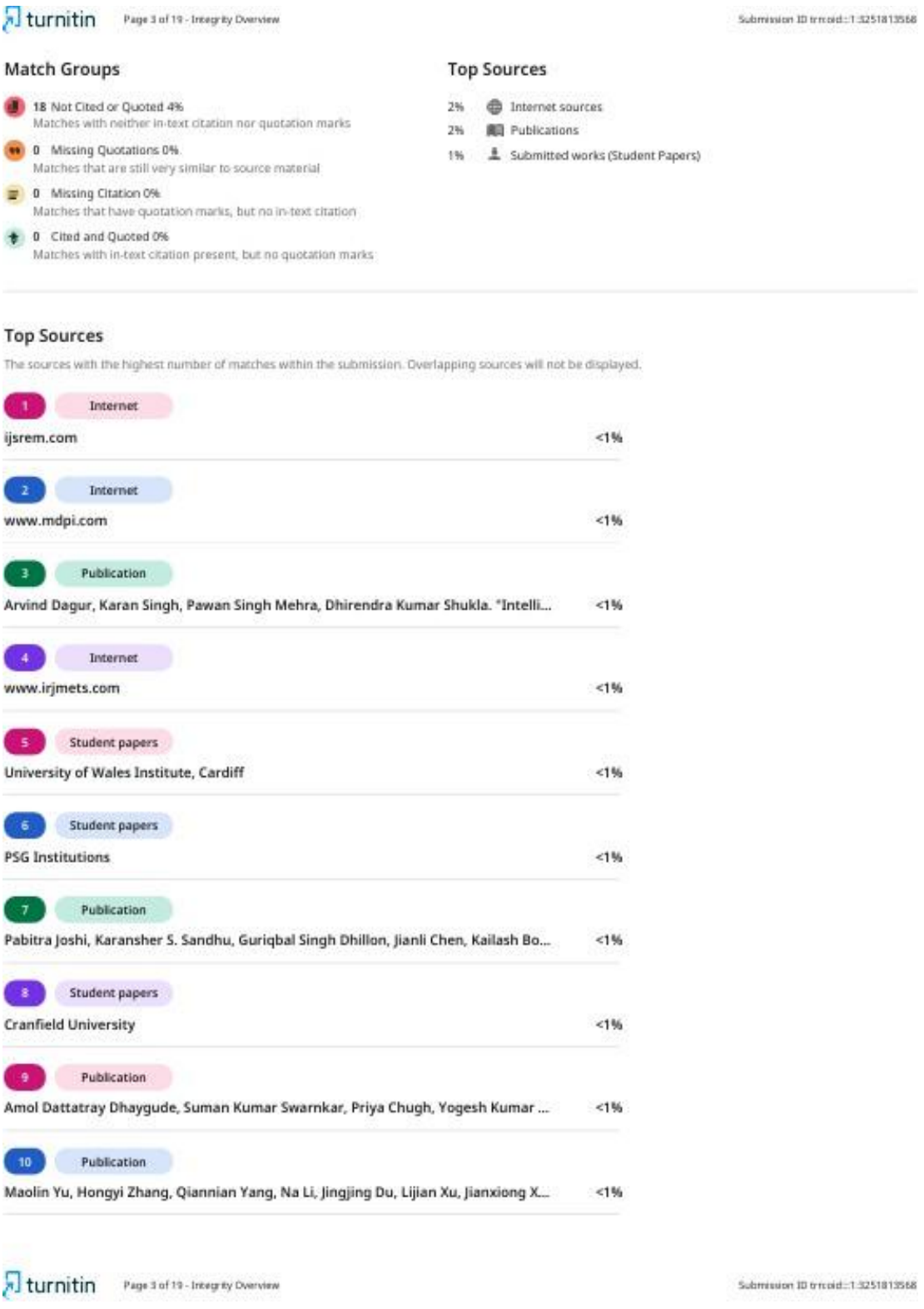
Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



turnitin Page 3 of 19 - Integrity Overview Submission ID trn:oid:::1:3251813568

Match Groups

- 18 Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

2% Internet sources
2% Publications
1% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
ijsrem.com		<1%
2	Internet	
www.mdpi.com		<1%
3	Publication	
Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...		<1%
4	Internet	
www.irjmets.com		<1%
5	Student papers	
University of Wales Institute, Cardiff		<1%
6	Student papers	
PSG Institutions		<1%
7	Publication	
Pabitra Joshi, Karansher S. Sandhu, Guriqbal Singh Dhillon, Jianli Chen, Kailash Bo...		<1%
8	Student papers	
Cranfield University		<1%
9	Publication	
Amol Dattatray Dhaygude, Suman Kumar Swarnkar, Priya Chugh, Yogesh Kumar ...		<1%
10	Publication	
Maolin Yu, Hongyi Zhang, Qiannian Yang, Na Li, Jingjing Du, Lijian Xu, Jianxiong X...		<1%

turnitin Page 3 of 19 - Integrity Overview Submission ID trn:oid:::1:3251813568

11	Publication	
Minh Dang, Hanxiang Wang, Yanfen Li, Tri-Hai Nguyen, Lilia Tightiz, Nguyen Xua...		<1%
12	Internet	
arxiv.org		<1%
13	Internet	
ijrpr.com		<1%
14	Publication	
I. Johnson, X Anitha Mary, A. Peniel Winifred Raj, J Chalmers, M. Karthikeyan, And...		<1%

Sustainable Development Goals (SDG) Report

Project: Grape Disease Detection System using YOLOv8 and Machine Learning

1. Introduction

Agricuiture is a backpower of a global food security Wthih the grow ing casacivaces smart technnology into farming practices. This project enhances smart technology intro farming practices. A smart technology into intentify and manage grapovine diseasees with hicaccuracy a ɜ iendam.

2. Alignment with

Sustainable Devevelopmennt ont Goals (SDGs)

Zero Hunger
Target 2.3: Essinel·1 sustainable productivivity and incomes of smail-acaale food producers, instroduces a smart'technology into targetining grapovine diseases efficienity up- using image processing, preelictive anaivitcs, and potential IoT integration as a werlila mpact

3. Alignment with Sustainable Development Goals (SDGs)

Industry, Innovation and Infrastructure
Target 9.5; Enharo– scientific research and upgrade technological capabilities capabilities.

➡ **Target 12.2:** Reduce chemjca angest through targeted use of agricultural inputs, reducing chamical waste through targeted use of agricultural inputs

Climate Alicon
Target 9.1: Strengo– pron resilience to climate-related hazards

➡ billghted. telutgenogı to Inıtimi- ragricul ural yiel∃ empfrovs. farmers pri moining resource conservation.

■ **Autchomous Crop Management**
Robots and drones for precision spcqg· and harvesting.

3. Project Highlights and SDG Integration

- **Expansion to Other Crops (SDG 2, 9)**
  Retrainable model for wider agricultural yield impact onbınzlinceas.
- **Integration with Irrigation & Nutrient Systems (SDG 8, 12), butomated** water and tertilizer control through IoT and sensor networks.
- **Autonomius Preditiion (SDG 9, 12)**
  Useif-roendly, rhmoible əpaɡolb əmaying and forecasting.
- **Autonomous: Crop Management**
  Robote and drones for precision spraying and harvestion promoting resource conservation and minmuring chernical

CERTIFICATE