ACKNOWLEDGEMENT

First and foremost I thank almighty God for the strength and blessings showered, for successfully completing Major Project. I would like to express my gratitude to all people who guided, inspired and motivated me during this work.

I express my sincere gratitude and thanks to our college Principal **Dr. B. Uma Maheswari M.sc., M.Phil., Ph.D.,** for providing me an opportunity to do my degree, Master of Computer Science in this esteemed Institution.

My heartfelt thanks to our respected Head of the Department **Mrs. K. Sasikala M.C.A., M.Phil., PGDBA.,** for her sincere endeavour in educating us. I thank my guide **Mrs. S. Leelalakshmi M.C.A., M.Phil.,** for providing me the necessary facilities to complete this Major Project and guidance me throughout this work. Without her valuable help and support this Major Project has never been possible.

I thank my staff members **Dr.(Mrs). M. Antony Ammal M.C.A., M.Phil., Ph.D., Dr.(Mrs). R. Janaki M.C.A., M.Phil., Ph.D., Dr. (Mrs) .S. Lakshmi Prabha M.Sc., M.Phil., M.Tech., Ph.D., Dr.(Mrs).K. Banu M.C.A., M.Phil., Ph.D., Dr. (Mrs). C.L. BrindhaDevi M.C.A., Ph.D., and Mrs. G. Priyadharshini M.C.A ., M.Phil.,** for their constant support to complete this Major Project.

I also thank **Mrs**. **Gayathri** Project Programmer, Pantech Technologies for providing me with the necessary technical support during the Project Programme.

I take this opportunity to acknowledge the help, guidance and co-operation given by my family and friends to complete this Project work in a successful manner.

ABSTRACT

Pest detection Technique is an important topic in modern intelligent agriculture.The agriculture is affected from serious challenges including complexity of wild environment, detection of tiny size pest and classification into multiple classes of pest. A tentative design of a deep learning based on essential role for predicting the pest is developed in this project. The system employs a deep neural network architecture to automate pest detection and anticipate potential diseases associated with the detected pests. Additionally, it provides remedies for identified pest diseases based on expert knowledge. In this project, pest detection system using various algorithm such as MobileNet Algorithm is used for designing the project. MobileNet algorithm is a computer vision model open-sourced by Google and designed for training classifier. It is used to recognized the Image of the project. The Image of pests has a two classes like yelow and incertulas classes. The dataset are collected from Kaggle Website. Accuracy was utilized to visualize the effectiveness of the model across different evaluation criteria. This project aims to enhance agricultural productivity and food security by facilitating proactive pest management strategies. The main objective of the project is to detect the pest and pest disease and also detect the remedies of the disease. This project very helpful for Farmers and Agricultural environment.

INDEX

1. INTRODUCTION

1.1 System Overview

Pest infestations pose significant threats to agricultural productivity and food security worldwide. Early detection and timely intervention are crucial for effectively managing pest outbreaks and minimizing crop damage. Traditional methods of pest detection often rely on manual observation, which can be time-consuming, labor-intensive, and prone to human error. In recent years, advancements in deep learning technology have opened up new possibilities for automating pest detection processes. Deep learning algorithms, particularly convolutional neural networks (CNNs), have shown remarkable performance in various image recognition tasks, making them promising candidates for pest detection in agricultural settings. In this context, this project aims to explore the application of deep learning techniques, specifically leveraging the MobileNet architecture, for pest detection in crop images captured using mobile devices. By developing a robust and efficient deep learning model, this project seeks to provide farmers with a practical tool for early pest detection, enabling proactive pest management strategies and ultimately contributing to enhanced agricultural sustainability and productivity.

1.2 About the organization

Pantech e Learning leading software services company focusing on Consulting Enterprises, Internet Applications, IT services, System Software, Networking and Telecom and Software Testing, Verification and Validation. This organization combines business and technical knowledge based on requirements of the client and ensure maximum Customer Satisfaction. They are Equipped with a team comprising of experienced and dedicated professionals. They use documented system to provide full satisfaction on the quality of products. A company of excellence is a team, a shared facility or an entity that provides leaderships, best practice, research, support and training for a focus area. It might also refer to network of institutions collaborating with each other to pursue excellence in particular area. Their aim is to provide high usability software that complies to the given specification and deliver it within the stipulated time frame.

2. REQUIREMENT ANALYSIS

2.1 Existing System:

In the existing system, Support Vector Machine (SVM) models are commonly used for pest detection in agricultural settings. SVM is a supervised machine learning algorithm that is effective for classification tasks, including distinguishing between healthy crops and those affected by pests or diseases. SVM works by finding the hyperplane that best separates different classes of data points in a high-dimensional space. In the context of pest detection, SVM models are trained on features extracted from crop images, such as color histograms, texture descriptors, or edge information. These models then classify new images as either healthy or pest-infested based on the learned patterns. While SVM-based approaches have shown promising results in some cases, they may struggle with complex image representations and require manual feature engineering, limiting their performance and scalability compared to deep learning-based methods.

Disadvantages:

- Limited Scalability
- Sensitivity to Parameter Tuning
- Binary Classification
- Interpretability

2.2 Proposed System:

The proposed system utilizes Convolutional Neural Networks (CNNs) for pest detection, leveraging the capabilities of deep learning to automatically extract relevant features from input images. By training a CNN model on a large dataset of pest images, the system aims to learn intricate patterns and features indicative of pests, enabling accurate classification. Unlike traditional SVM approaches, CNNs can automatically learn hierarchical representations of features from raw pixel values, eliminating the need for manual feature engineering. Additionally, CNNs inherently support multi-class classification, making them suitable for scenarios with multiple pest types. The proposed CNN-based system offers improved scalability, robustness to noise, and interpretability compared to traditional SVM approaches, thus providing a more effective solution for pest detection tasks.

Advantages:

- High Accuracy
- Scalability
- Robustness
- Interpretability

2.3 System Requirements

2.3.1 Hardware Requirement

- Operating System : Windows 10
- RAM : 4GB
- Hard Disk : 250GB
- Processor : AMD PRO A4-4350B R4,5 COMPUTE CORES 2.50 GHz

2.3.2 Software Requirement

- Language : Python
- Packages : Streamlit
- Frontend : Python IDLE
- Backend : Anaconda Navigator

2.4 Software Description

Python

Python software offers dynamic adaptability and scalability, evolving seamlessly with business needs. With flexible customization and seamless integration, it optimizes workflows and enhances productivity. Security is paramount, ensuring robust protection for sensitive data through encryption and access controls. Intuitive user interfaces and streamlined navigation prioritize user experience, minimizing learning curves. Automation streamlines tasks, freeing resources for strategic initiatives, while analytics provide actionable insights for informed decision-making. Supported by reliable maintenance services, our Python software empowers businesses to thrive in a dynamic digital landscape.

Deep Learning

Deep learning utilizes neural networks to analyze complex data patterns, revolutionizing fields like computer vision and natural language processing. Its hierarchical layers facilitate accurate predictions, but it demands substantial computational resources and extensive data. Ongoing research aims to enhance efficiency and generalization. As a transformative technology, deep learning powers innovations in healthcare, finance, and beyond. Its scalability and adaptability make it potent for solving real-world challenges. Despite challenges, deep learning stands at the forefront of AI innovation, offering unprecedented opportunities.

Python IDLE

Python IDLE is an integrated development environment (IDE) bundled with Python, providing a user-friendly interface for coding and debugging Python scripts. It offers features like syntax highlighting, code completion, and interactive shell, facilitating efficient development. IDLE supports various operating systems, making it accessible to a wide range of users. Its simplicity makes it ideal for beginners learning Python programming. It enables quick script execution and experimentation, aiding in rapid prototyping and debugging. As an integral part of the Python ecosystem, IDLE continues to play a significant role in Python development workflows.

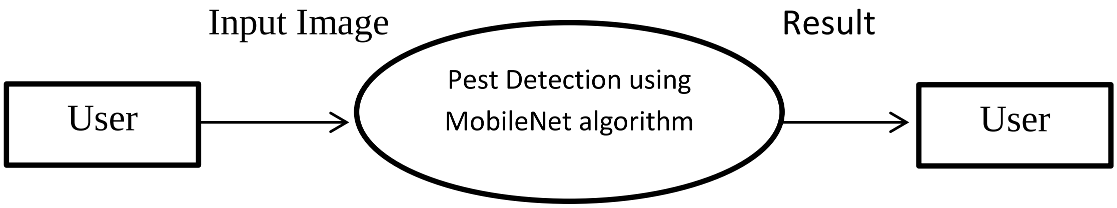
Anaconda Navigator

Anaconda Navigator is a graphical user interface bundled with the Anaconda distribution, facilitating management of Python packages, environments, and applications. It simplifies the process of installing and updating packages, making it ideal for data science and scientific computing projects. Navigator offers access to a vast library of pre-installed packages and tools, including Jupyter Notebook and Spyder IDE. With its intuitive interface, users can create and manage virtual environments effortlessly. Nonetheless, it remains a popular choice among data scientists and Python enthusiasts for its convenience and functionality.

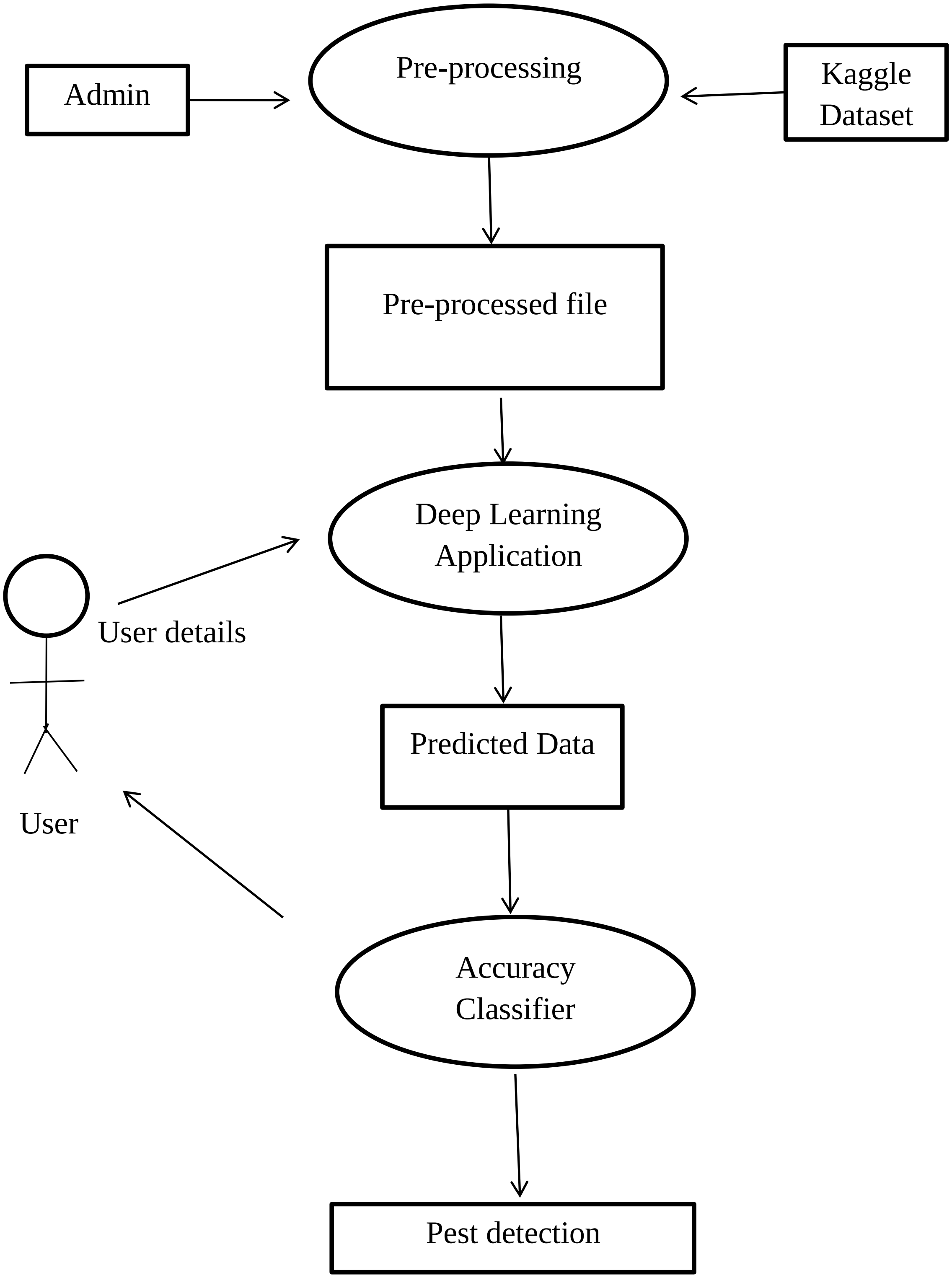
3. SYSTEM ANALYSIS

3.1 Data Flow Diagram

LEVEL – 0

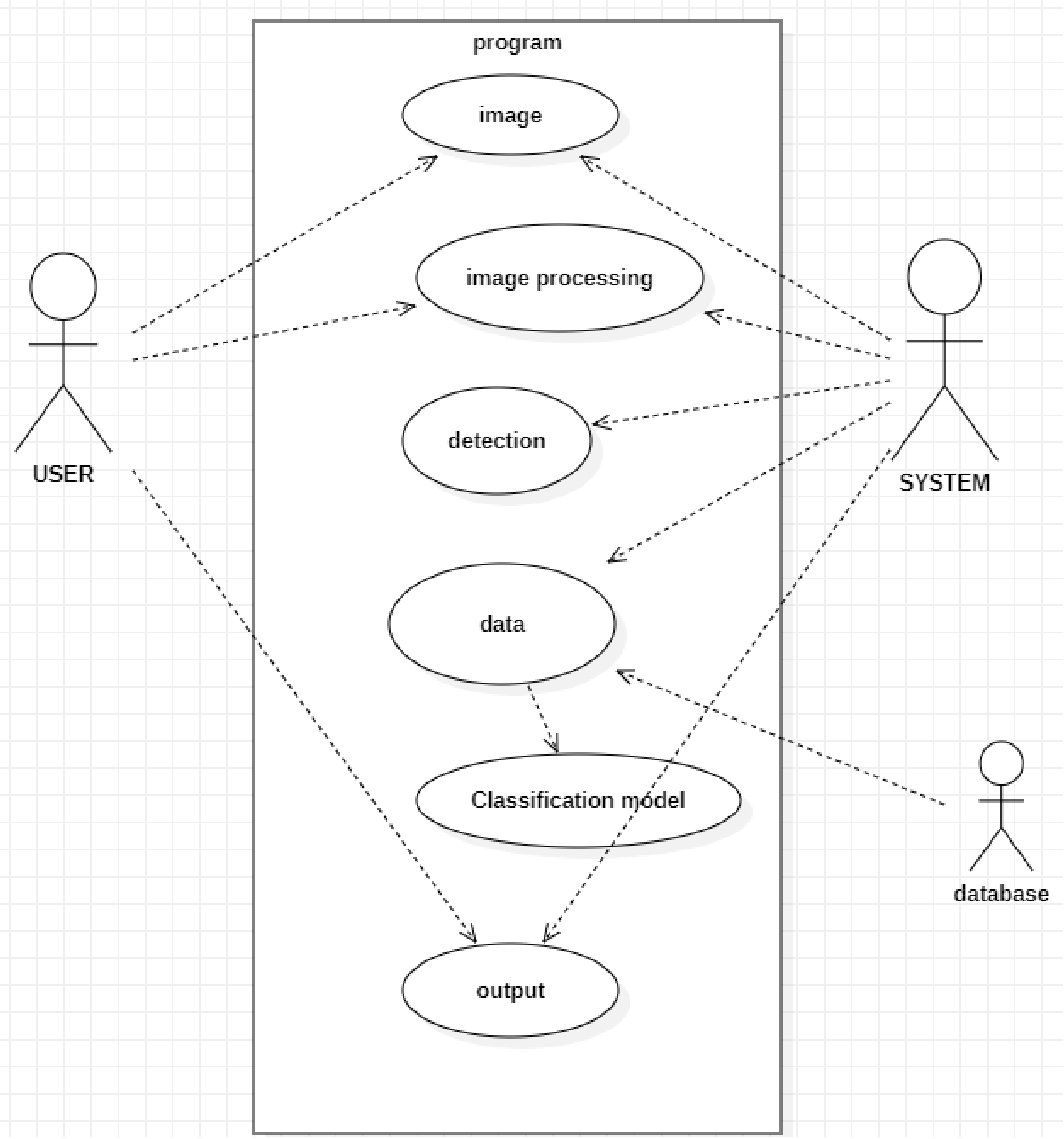


LEVEL 1:



Admin
Pre-processing
Kaggle Dataset
Pre-processed file
Deep Learning Application
User details
User
Predicted Data
Accuracy Classifier
Pest detection

3.2 Use Case Diagram



4. SYSTEM DESIGN

4.1 Modules Description

- Data Collection for pest images in Kaggle
- Data Pre-processing for Dataset
- Data Visualization
- Testing and Training
- Pest Detection Using Streamlit

4.1.1 Data Collection for pest imges

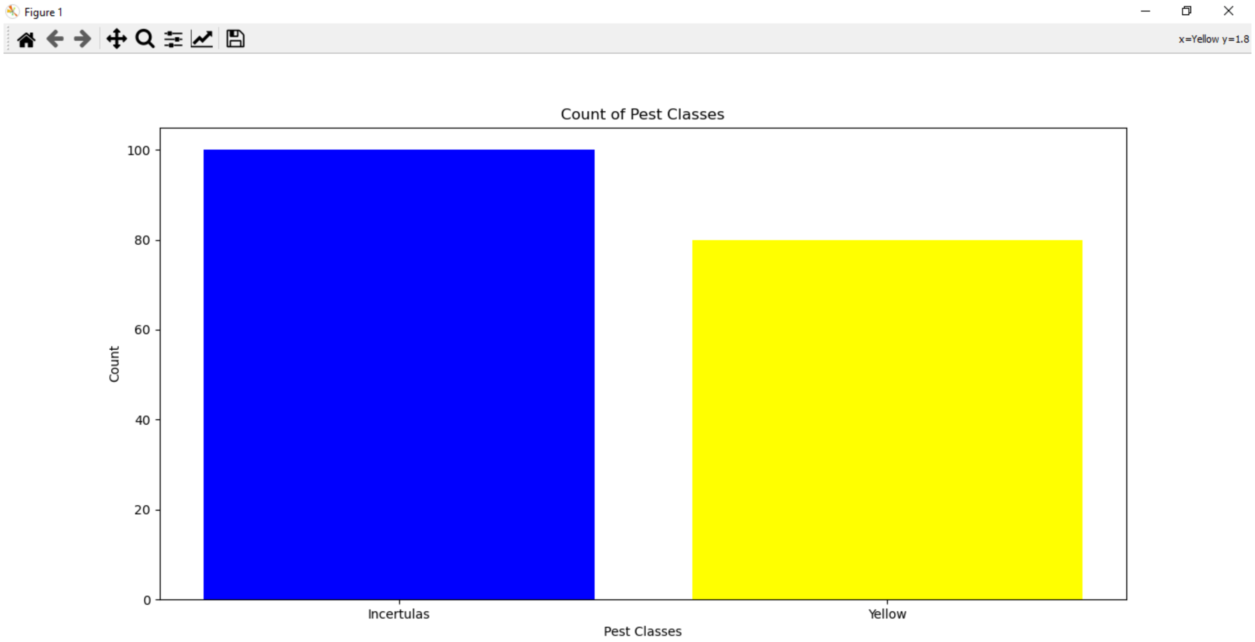
In a pest detection project using deep learning and data collected from Kaggle, comprehensive image datasets are gathered, annotated, and augmented. High-resolution images of crops affected by pests are obtained, covering various infestation stages and angles. Annotations marking pest locations are meticulously added, ensuring accurate training of deep learning models. Dataset augmentation techniques, such as rotation and flipping, enhance variability and model robustness. Quality assurance measures validate data integrity, while dataset splitting enables effective model evaluation. Continuous updates ensure adaptability to changing pest patterns, fostering accurate pest detection in agricultural settings.

4.1.2 Data Pre-processing for pest images

In preparing data for a pest detection project with MobileNet algorithm in deep learning, start by collecting a diverse dataset of pest images and non-pest images. Ensure proper labeling and augmentation techniques to balance the classes and enhance model generalization. Resize all images to a consistent input size compatible with MobileNet architecture. Normalize pixel values to a range suitable for training. Finally, split the dataset into training, validation, and test sets to evaluate model performance accurately.

4.1.3 Data Visualization for dataset

Data visualization plays a crucial role in understanding and communicating insights from data. In this example, we utilize Matplotlib, a widely-used Python library, to create a bar chart illustrating the counts of two pest classes, namely 'Incertulas' and 'Yellow'. Each bar represents the count of occurrences for the respective pest class. The choice of colors, blue for 'Incertulas' and yellow for 'Yellow', aids in visually distinguishing between the classes. The x-axis denotes the pest classes, while the y-axis represents the count. With clear labeling of axes and a descriptive title, the plot becomes self-explanatory, facilitating easy interpretation of the data. This visualization serves not only to present the data effectively but also to facilitate further analysis and decision-making processes related to pest management strategies. Such visual representations are invaluable tools in various domains, enabling stakeholders to grasp insights swiftly and make informed decisions based on data-driven evidence.



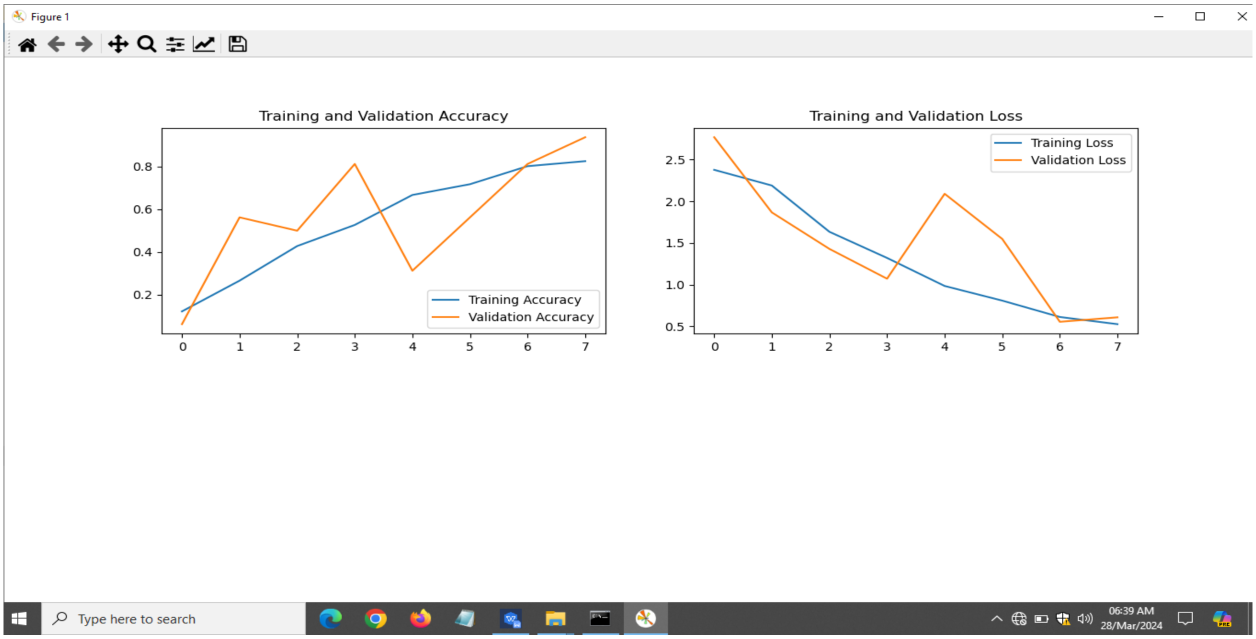
4.1.4 Training and Testing

Training

The Pest Disease Classifier was trained using the Keras library, leveraging its intuitive interface atop TensorFlow. The dataset, containing images of 'incertulas' and 'yelow' pest diseases, was split into training and validation sets. Preprocessing included resizing and normalization, with data augmentation techniques applied for enhanced variability. The model, comprising convolutional and dense layers, underwent iterative parameter updates via optimization algorithms. Training involved multiple epochs to enable the model to learn to distinguish between different pest diseases.

Testing

For evaluation, a separate testing dataset was reserved to assess the model's generalization. Deploying the trained model on this unseen data, predictions were generated and compared with ground truth labels. Evaluation metrics including accuracy, precision, recall, and F1-score were computed, providing insights into the model's performance. This rigorous testing phase validated the Pest Disease Classifier's ability to accurately identify pest diseases based on input images, offering valuable feedback for potential refinements.



4.1.5 Pest Detection Using Streamlit

In the system designed for pest detection, Streamlit serves as the interface, facilitating user interaction with the deep learning model. Uploaded crop images undergo real-time processing by the deep learning algorithm, such as MobileNet, enabling instant pest detection feedback. The interface displays detection results, empowering stakeholders to analyze and visualize pest presence efficiently. Continuous model updates and feedback mechanisms ensure ongoing improvement and adaptability for accurate pest identification.

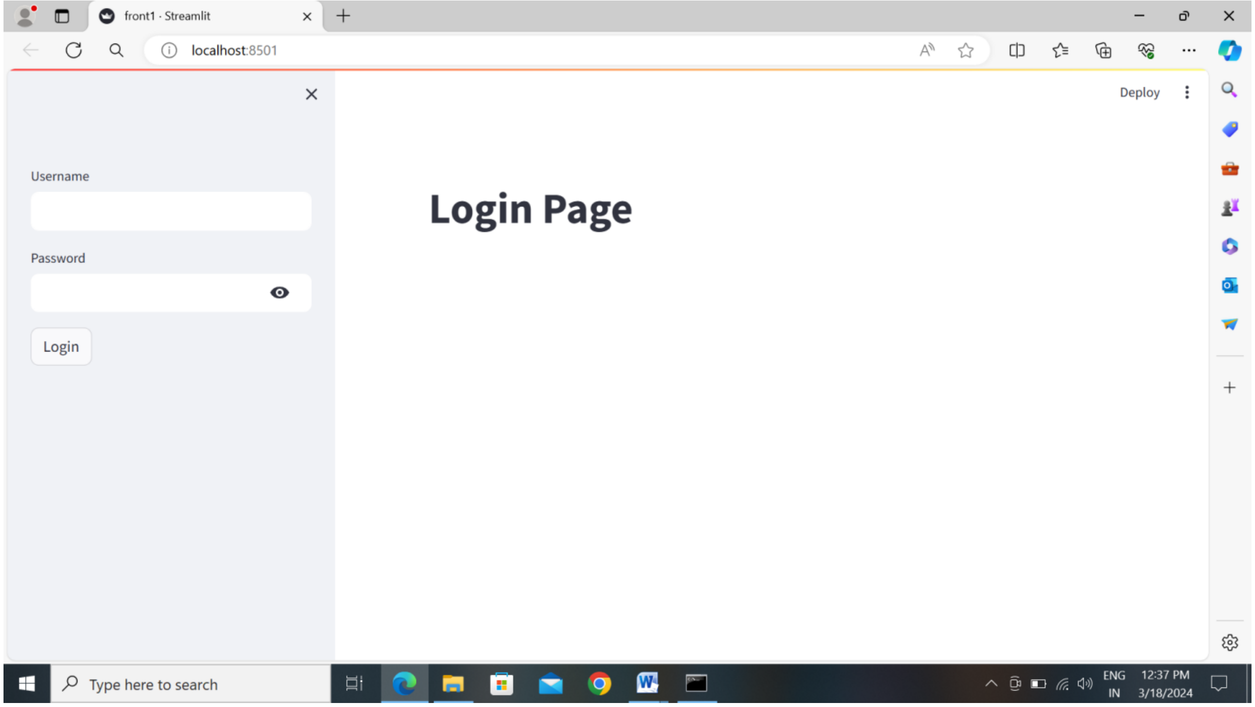
MobileNet Algorithm

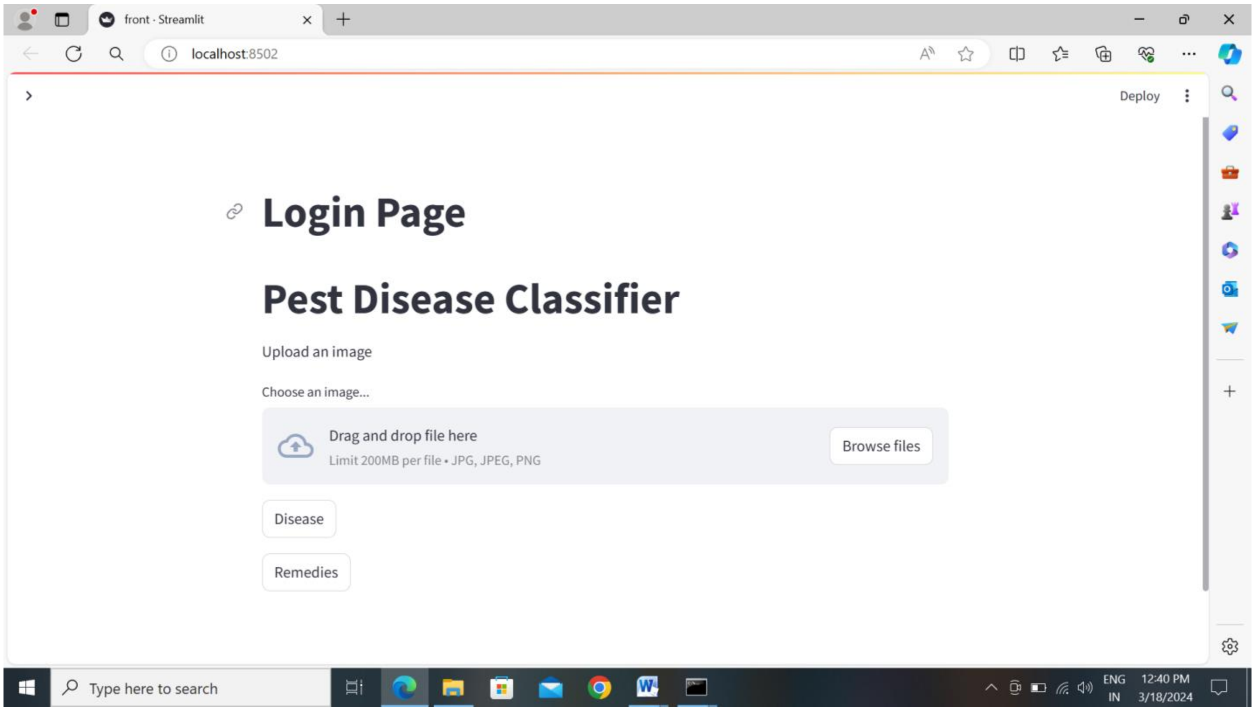
In this pest detection project, the MobileNet algorithm serves as the backbone for image classification. Leveraging the pre-trained MobileNet model, our system efficiently analyzes uploaded images to predict pest diseases with high accuracy. By harnessing MobileNet's deep convolutional neural network architecture, we can discern subtle patterns and features crucial for distinguishing between different pest classes. This integration not only enhances the model's predictive capabilities but also enables rapid inference, making it suitable for real-time pest detection applications. Moreover, MobileNet's lightweight design ensures optimal performance, facilitating seamless deployment across various platforms, including web-based interfaces like Streamlit..

4.2 TABLE DESIGN

Field Name	**Data Type**
Username	String
Password	String(password)
Class names	String
Pest disease	String
Pesticide Recommendations	string

4.3 FORM DESIGN





5. SAMPLE CODING

```
import numpy as np

import streamlit as st

import cv2

from keras.models import load_model


# Loading the Model

model = load_model('model.H5')


# Name of Classes

CLASS_NAMES = [

    "incertulas",

    "yelow"

]


# Pesticide recommendations for each class

PESTICIDE_RECOMMENDATIONS = {

    "incertulas": "Pesticides: Chlorantraniliprole, Emamectin benzoate, Indoxacarb",

    "yelow": "Buprofezin, Fipronil, Thiamethoxam"

}


# Login Page

def login():

    st.title("Login Page")
    username = st.sidebar.text_input("Username")
    password = st.sidebar.text_input("Password", type="password")
    if st.sidebar.button("Login"):
        if username == "vennila" and password == "0109":
            st.sidebar.success("Logged in as {}".format(username))
            home()
        else:
            st.sidebar.error("Incorrect username or password")
    return False

# Home Page
def home():
    st.title("Pest Disease Classifier")
    st.markdown("Upload an image ")

# Uploading the image
    uploaded_image = st.file_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

# On predict button click
    if uploaded_image is not None:
    # Convert the file to an opencv image.
        file_bytes = np.asarray(bytearray(uploaded_image.read()), dtype=np.uint8)
        opencv_image = cv2.imdecode(file_bytes, 1)

        # Display the image
        st.image(opencv_image, channels="BGR", caption='Uploaded Image')


        # Preprocess the image
        resized_image = cv2.resize(opencv_image, (200, 200))  # Resize to the appropriate input size of the model
        resized_image = resized_image / 255.0  # Normalize the image


        # Make prediction
        prediction = model.predict(np.expand_dims(resized_image, axis=0))[0]


        # Display the result
        predicted_class_index = np.argmax(prediction)
        predicted_class = CLASS_NAMES[predicted_class_index]
        confidence = prediction[predicted_class_index]


        # Get pesticide recommendations
        pesticide_recommendation = PESTICIDE_RECOMMENDATIONS.get(predicted_class, "No recommendation available")


    if st.button("Disease"):
        st.write(f"Predicted Disease: {predicted_class} (Confidence: {confidence:.2f})")
    if st.button("Remedies"):
        st.write(f"Recommended Pesticide(s): {pesticide_recommendation}")
login()
home()
```

6. TESTING

The purpose of testing is to discover errors. Testing is the purpose of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 Unit Testing

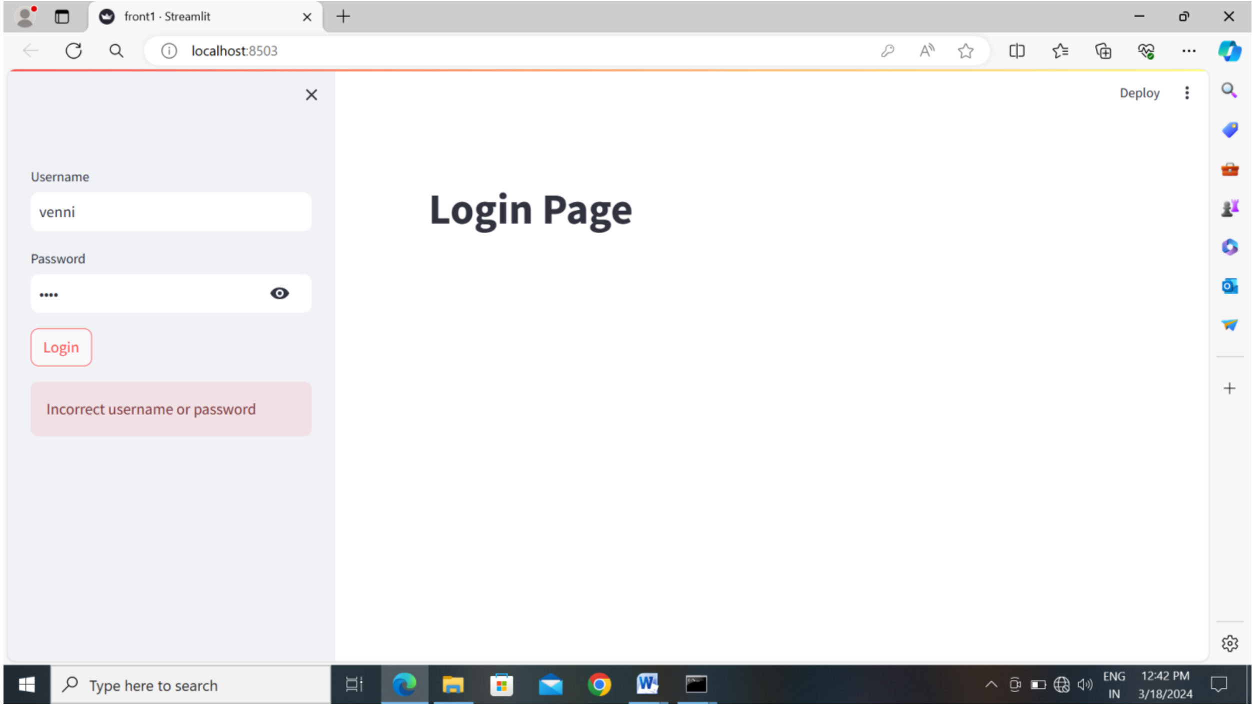
Unit Testing involves the design of the test cases that validate that the internal program logic is functioning properly, and the program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process application, and system configuration. Unit testing has been done for each modules like Data collection, Pre-processing, Data Visuallization of the modules provides correctly. Proper inputs are given and corresponding outputs are checked.

6.2 Integration Testing

Integration testing for a pest detection project using Mobile Net algorithm in Python with deep learning is crucial for ensuring the seamless interaction of various components. It involves testing the integration of data preprocessing, model inference, and result visualization modules to validate their interoperability. Furthermore, it verifies the compatibility of the system with different mobile platforms and environments to ensure consistent performance across devices. Rigorous testing scenarios should encompass diverse datasets, representing various pest species and environmental conditions, to validate the model's robustness and generalization capabilities. Continuous integration practices can streamline this process, facilitating rapid iteration and deployment of improvements.

6.3 Validation Testing

Validation Testing is when your program checks the data to make sure it meets some rules or restrictions. It is important to make sure that valid data is passed and used within the program and that there can be no security or data corruption.



Username

venni

Password

••••

Login

Incorrect username or password

Login Page

7. CONCLUSION & FUTURE SCOPE

In conclusion, the deep learning-based pest detection system utilizing the Mobile Net algorithm offers a promising solution for accurately and efficiently identifying pests in agricultural settings. By leveraging the power of convolutional neural networks, the system achieves high accuracy, scalability, and robustness, enabling real-time processing and reducing dependency on manual intervention. With further optimization and deployment, this technology has the potential to revolutionize pest management practices, leading to improved crop yields and agricultural sustainability.

The future scope for a pest detection project utilizing MobileNet algorithm in Python with deep learning is promising. By harnessing the power of deep learning, the project can evolve to accurately identify and classify various pests in real-time through mobile devices, aiding farmers in pest management. Integration with cloud computing could enhance scalability and allow for continuous model improvement through data aggregation. Expansion into multi-modal detection, incorporating additional sensory data such as infrared or ultraviolet imagery, could further enhance accuracy and robustness. Moreover, collaboration with agricultural experts and stakeholders could ensure practical applicability and widespread adoption of the solution.

REFERENCES

1. Author: Vijayakumar, V. Book: "Deep Learning for Image Processing Applications: With MatLab Examples" Published Year: 2019

2. Author: Manogaran, G. Book: "Deep Learning Techniques and Optimization Strategies in Big Data Analytics" Published Year: 2020

3. Author: Subramanian, S. Book: "Deep Learning: Fundamentals and Applications" Published Year: 2019

4. Author: Zhou, J. Book: "Deep Learning in Agriculture" Published Year: 2020

5. Author: Khan, M. K. Book: "Deep Learning Applications and Intelligent Decision Making in Engineering and Healthcare" Published Year: 2020

WEBSITES

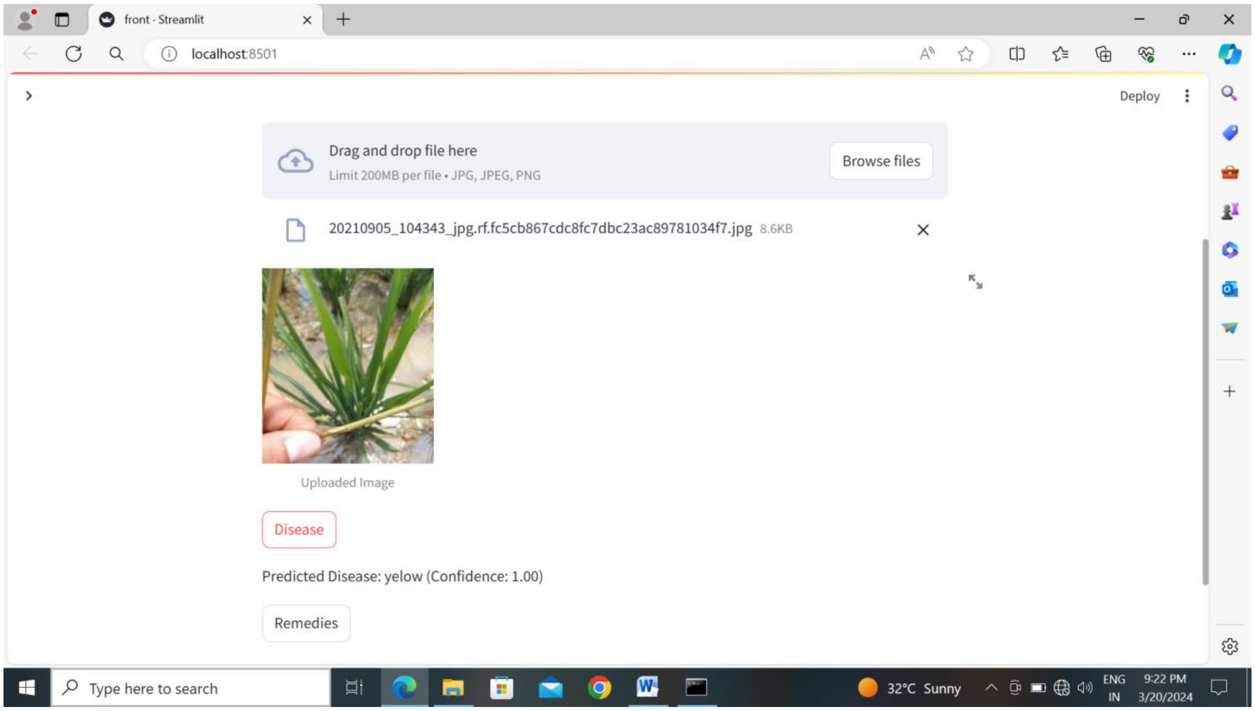
www.kaggle.com

http://www.geeksforgeeks.org/pest-detection-using-deep-learning-in-python/

http://www.researchgate.net/publication/34409903-Employing-Deep-Learning-for-Pest-disease-Detection/

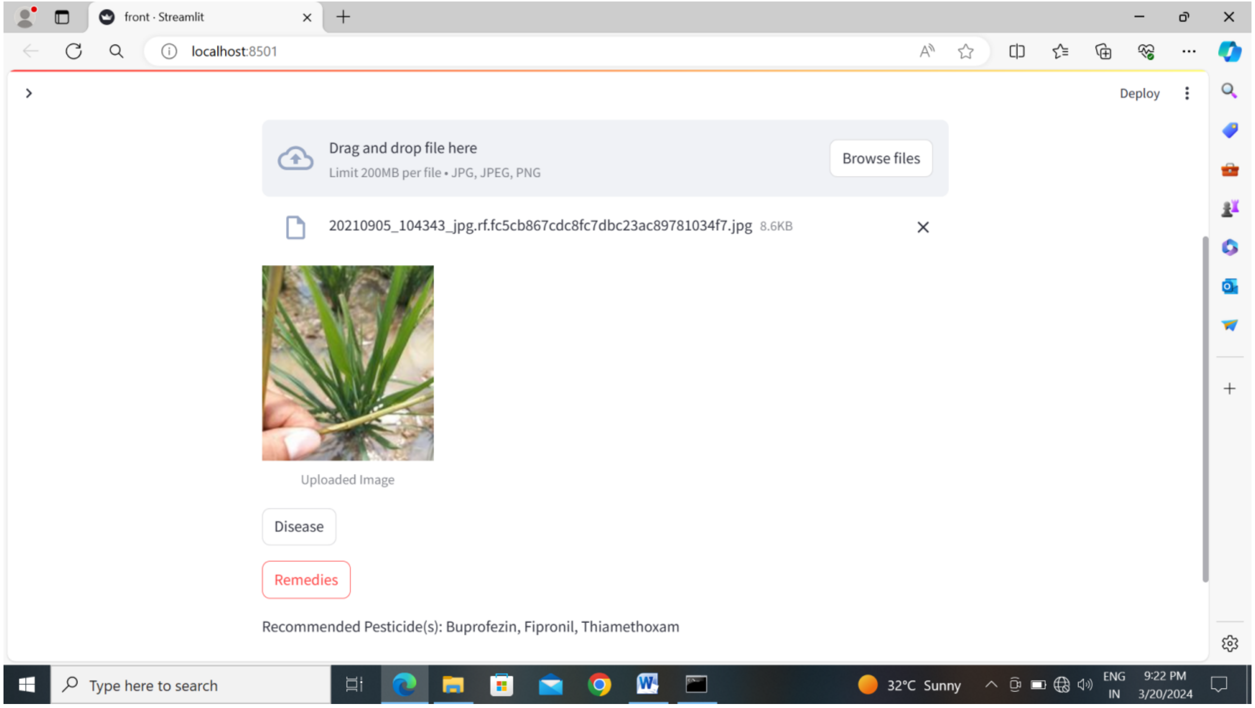
APPENDIES

SCREENSHOT



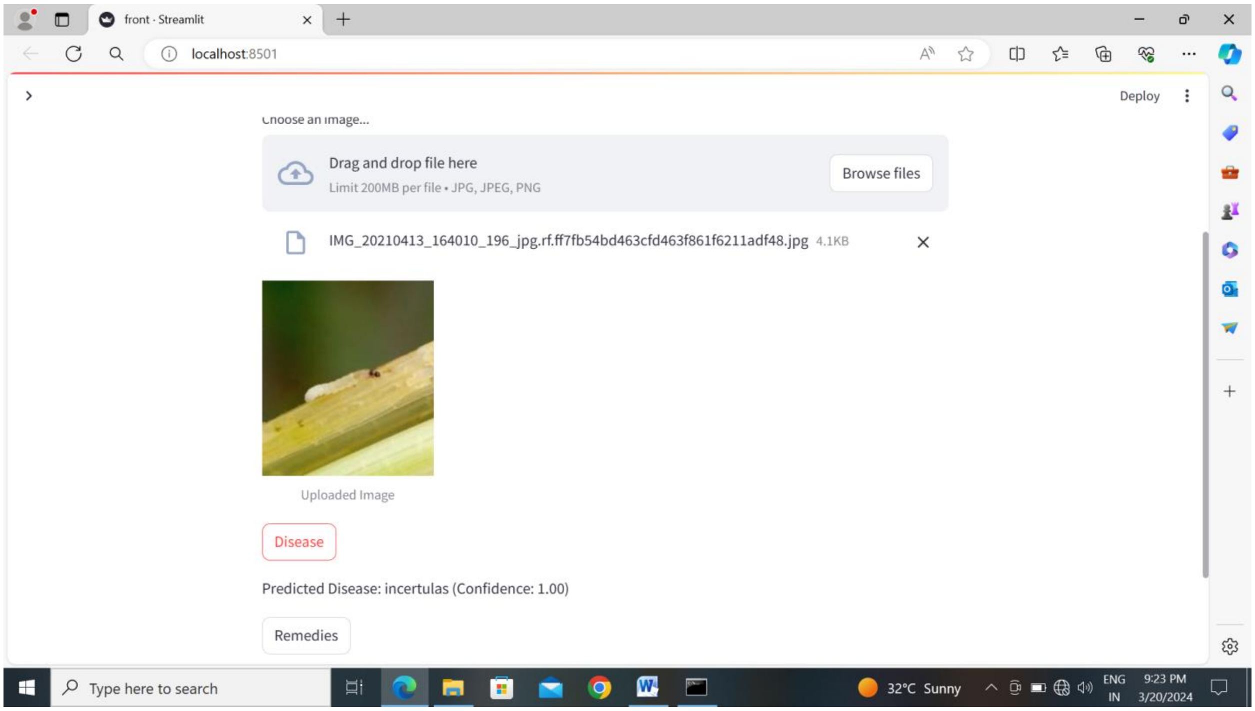
20210905_104343_jpg.rf.fc5cb867cdc8fc7dbc23ac89781034f7.jpg 8.6KB

Predicted Disease: yelow (Confidence: 1.00)

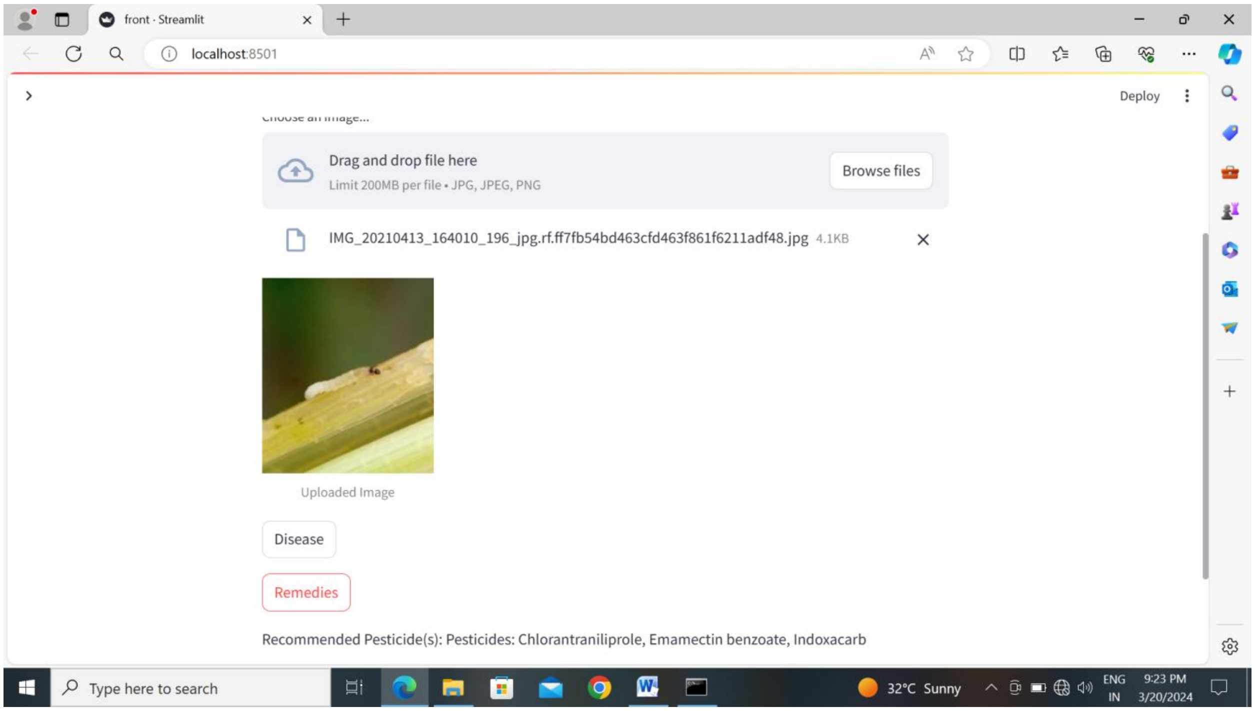


20210905_104343_jpg.rf.fc5cb867cdc8fc7dbc23ac89781034f7.jpg 8.6KB

Recommended Pesticide(s): Buprofezin, Fipronil, Thiamethoxam



Predicted Disease: incertulas (Confidence: 1.00)



Recommended Pesticide(s): Pesticides: Chlorantraniliprole, Emamectin benzoate, Indoxacarb