**Infosys Internship 4.0 Project**

**Documentation**

**Title: Project Documentation: PlantAI - Automated Plant Disease Detection using AI Vision Algorithm**

•**Introduction:**

**Overview:** The Plant Disease Detection System aims to leverage AI to identify diseases in plants, enhancing early detection and timely intervention. This project is crucial for supporting precision agriculture and reducing the reliance on manual inspections. The PlantAI project aims to develop an automated system for detecting plant diseases using convolutional neural networks (CNNs). The project leverages image processing and deep learning techniques to identify and classify various plant diseases from leaf images

**Objectives:**

* To build a CNN model capable of classifying plant diseases.
* To develop a web application for users to upload leaf images and receive disease predictions
* To provide a user-friendly interface for farmers and agricultural experts.

**Team Members and Roles:**

**Phani Manoj**: Project Developer

**Group A: Project Demonstration Group B: Project Presentation**

1. Chaithanya 1. Abhay Thakur

2. Abhiram Koppuravuri 2. Phani manoj

3. Bazzurla Aishwarya 3. Puranasree

4. Bhukya Bhushnam 4. Sahil shukla

5. Chinmaya Kumarpalo 5. Yogesh Patil

6. Dhanashree 6. Badrinath Reddy

7. Dhrumi 7. Vikrant Kulkarni

8. Gautham sankar

9. Jineshn shah

**Mentors**: Mr. Deepak Sir & Mrs. Madhupriya Ma’am, Infosys

**•Project Scope:**

**Included in the Project:**

1. Machine Learning Model Development
2. Dataset Collection and Annotation
3. Application Development
4. Performance Evaluation and Testing
5. Documentation and Deployment Plan
6. Research Contribution

**Not Included in the Project:**

1. Real-time Monitoring Systems
2. Multi-crop Disease Detection
3. Extensive User Training Programs

**Limitations and Constraints:**

* Computational Resources
* Field Testing Conditions
* User Accessibility

**•Requirements:**

**Functional Requirements:**

* Users can upload images of tea leaves through the application.
* The system identifies and classifies diseases present in the uploaded tea leaf images.
* The application displays the diagnostic results, including the type and severity of the disease.

**Non-Functional Requirements:**

* The system must process and analyze images quickly, providing diagnostic results within seconds
* The application must be user-friendly, with an intuitive interface that is easy to navigate.

**User Stories and Use Cases**

User Story: As a user, I want to upload images of tea leaves to receive diagnostic results.

Use Case: user uploads image -> System processes image -> System detects disease -> System displays results.

User Story: As a user, I want to see detailed information about the detected disease and recommended actions.

Use Case: Disease detected -> System provides detailed information and recommendations -> user views and takes action.

**•Technical Stack:**

**Programming Languages:**

* Python
* Java script
* Html
* Css

**Frameworks/Libraries:**

* TensorFlow/Keras for model development
* Flask for web application
* Open cv
* Scikit-learn

**Tools/Platforms:**

* Jupyter Notebook for model training
* VS Code for development
* Anaconda for environment management
* Git for version control

**•Architecture/Design:**

**System Architecture:**

* + **Components:**
* **Data Processing:** Prepares and augments the dataset for training.
* **Model Training:** Trains the CNN model on the processed dataset.
* **Web Application:** Provides an interface for users to upload images and view predictions.
  + **Design Decisions:**
* Chose CNN for its effectiveness in image classification tasks.
* Used Flask for its simplicity and ease of integration with machine learning models.
* Opted for local deployment to avoid complexities of cloud infrastructure

Diagrams:

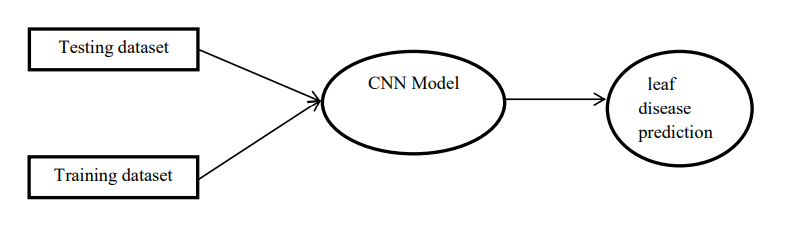


Fig 1:data flow Diagram

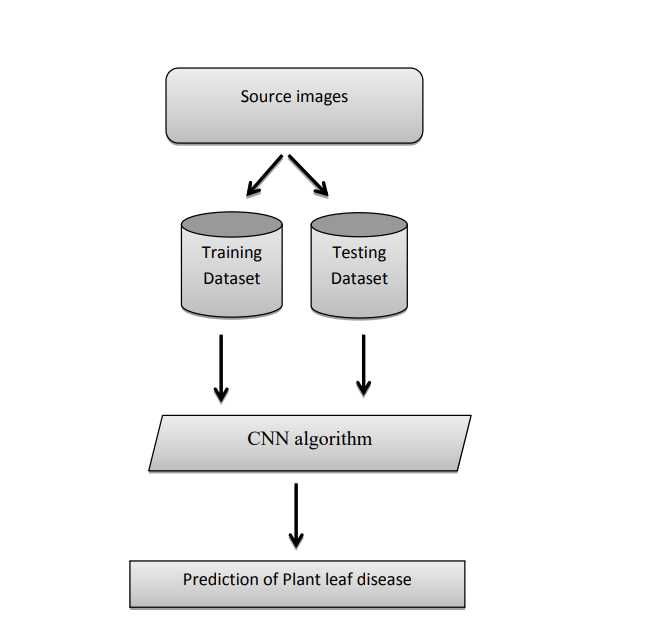


Fig 2:WorkFlow Diagram

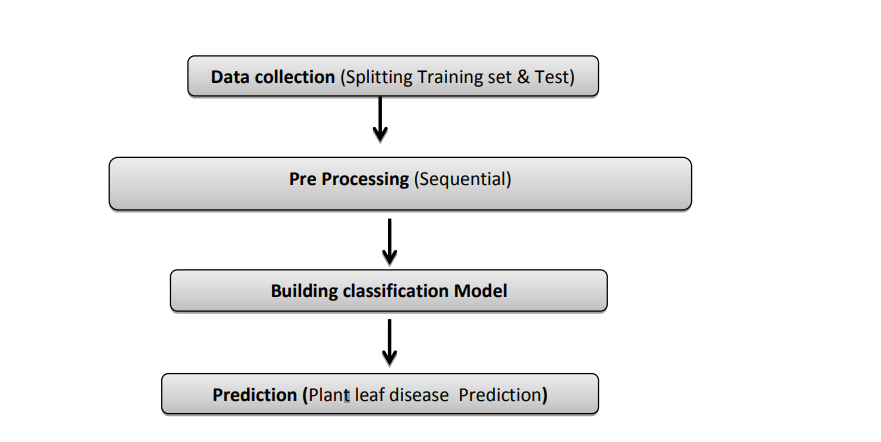


Fig 3:data flow diagram for CNN model

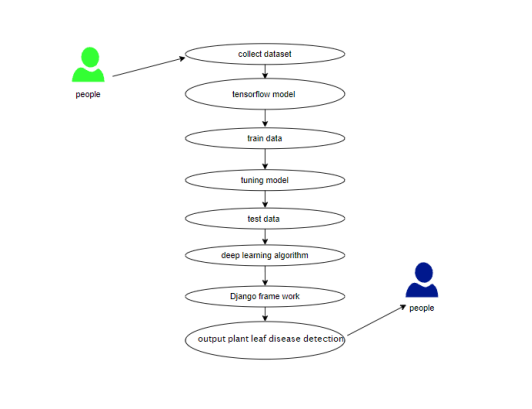


Fig 4:use case diagram

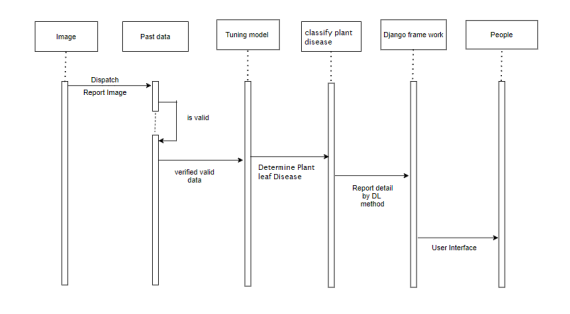


Fig 5:Sequence Diagram

Over of the entire system is shown in the below figure

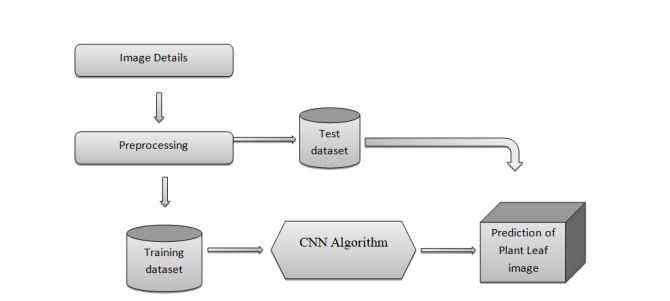


Fig 6:overview of the system

**•Development:**

**Programming Languages:**

Python: Used for backend server logic, image processing, and machine learning model development.

JavaScript: Used for front-end development of the web application.

HTML/CSS: Used for structuring and styling the web application.

**Frameworks:**

Flask/Django: Frameworks for building the backend server and RESTful API.

Pandas/Numpy: Libraries for data manipulation and analysis.

OpenCV: Library for image processing and computer vision tasks.

**Challenges Encountered and Solutions**

1)Data Quality and Diversity

**Issue:** Ensuring a diverse and high-quality dataset for training the machine learning model.

**Solution:** Collaborated with tea farmers and researchers to collect a wide range of tea leaf images from different regions and conditions. Applied data augmentation techniques to increase dataset diversity.

2) Model Accuracy and Performance

**Issue:** Achieving high accuracy and performance in disease detection.

**Solution:** Iteratively tuned hyperparameters and experimented with different architectures to improve model accuracy. Used transfer learning with pre-trained models to enhance performance.

**•Testing:**

**Testing Approach:**

* **Unit Testing:** Ensure that individual components or functions work correctly in isolation.
* **Integration Testing:** Verify that combined components work together as expected.
* **System Testing:** Ensure the entire system works as a whole.
* **Performance Testing:**Assess the system's performance under load and stress conditions.

**Results:**

* Identified and fixed issues related to input validation, data processing, and minor logical errors.
* Fixed issues related to API communication, data format inconsistencies, and state management in the frontend
* Enhanced image processing pipeline for faster execution.

**•Deployment:**

**Deployment Process:**

* Created a virtual environment using Anaconda.
* Installed required dependencies from requirements.txt.
* Deployed the Flask application locally for testing.

**Deployment Instructions:**

1. Clone the repository.
2. Create and activate a virtual environment.
3. Install dependencies using pip install -r requirements.txt.
4. Run python app.py to start the Flask server.
5. Access the application at http://127.0.0.1:5000/.

By following these deployment steps and configuring services accordingly, the application can be effectively deployed and scaled across different environments, ensuring reliable operation and efficient management.

**•User Guide:**

**Setup Instructions:**

* Ensure all dependencies are installed.
* Run the Flask application as described in the deployment section.

**Using the Application:**

* Open the web application in a browser.
* Upload a leaf image using the provided interface.
* View the prediction results displayed on the screen.

**Troubleshooting Tips:**

* Ensure you have a stable internet connection.
* Check permissions for accessing the device's camera or gallery
* Processing times may vary depending on server load. Wait a moment and try again if processing takes longer than usual.
* If you believe the results are inaccurate, ensure the image is clear and shows the affected tea leaves properly

**•Conclusion:**

**Project Outcomes and Achievements**

* The application accurately identifies various diseases affecting tea leaves, providing timely diagnostic results.
* Designed intuitive interfaces for both web and mobile platforms, ensuring accessibility and ease of use for farmers.
* Optimized image processing and machine learning models to handle real-time requests effectively, enhancing application performance.

**Lessons Learned:**

* Ensuring diverse and high-quality datasets is essential for training accurate machine learning models.
* Continuous user feedback and iterative design processes are critical for developing applications that meet user needs effectively.

**Areas for Improvement:** .

* Increase dataset size and diversity.
* Implement more advanced techniques for augmenting datasets to further improve model accuracy and robustness
* Optimize the model for faster predictions.
* Consider deploying the application on cloud platforms for scalability.

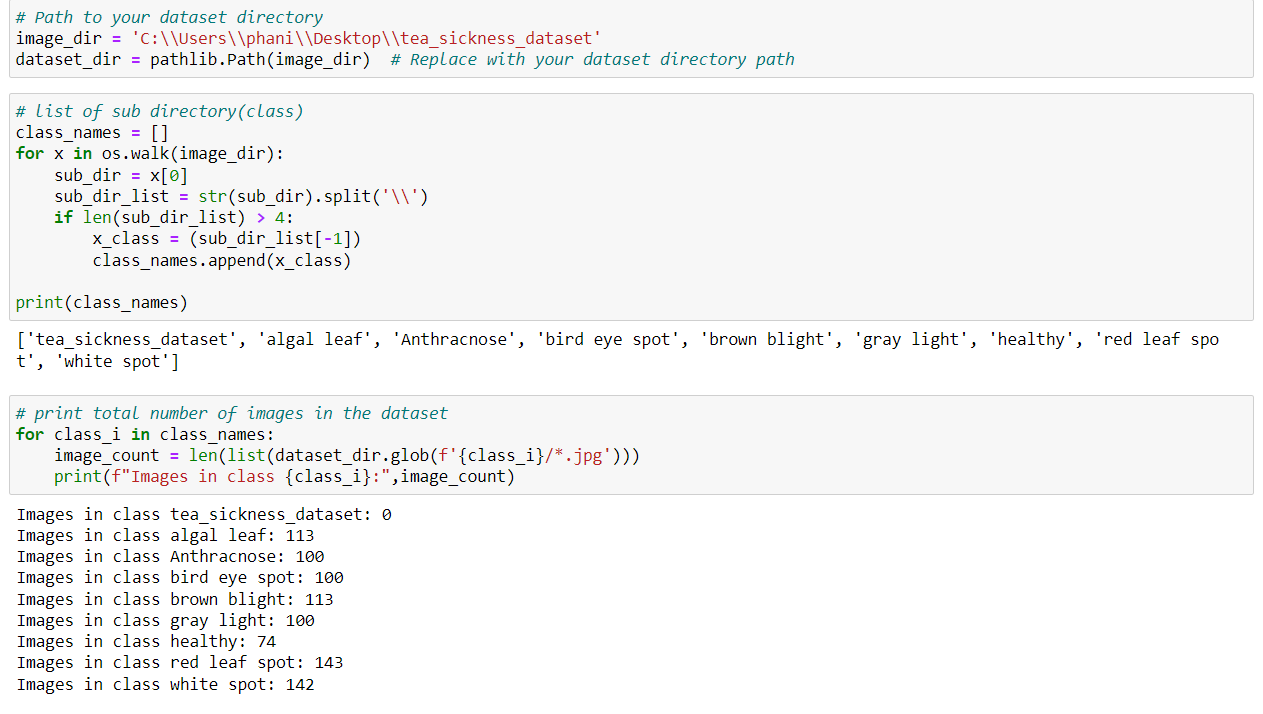
**Future Prospects:**

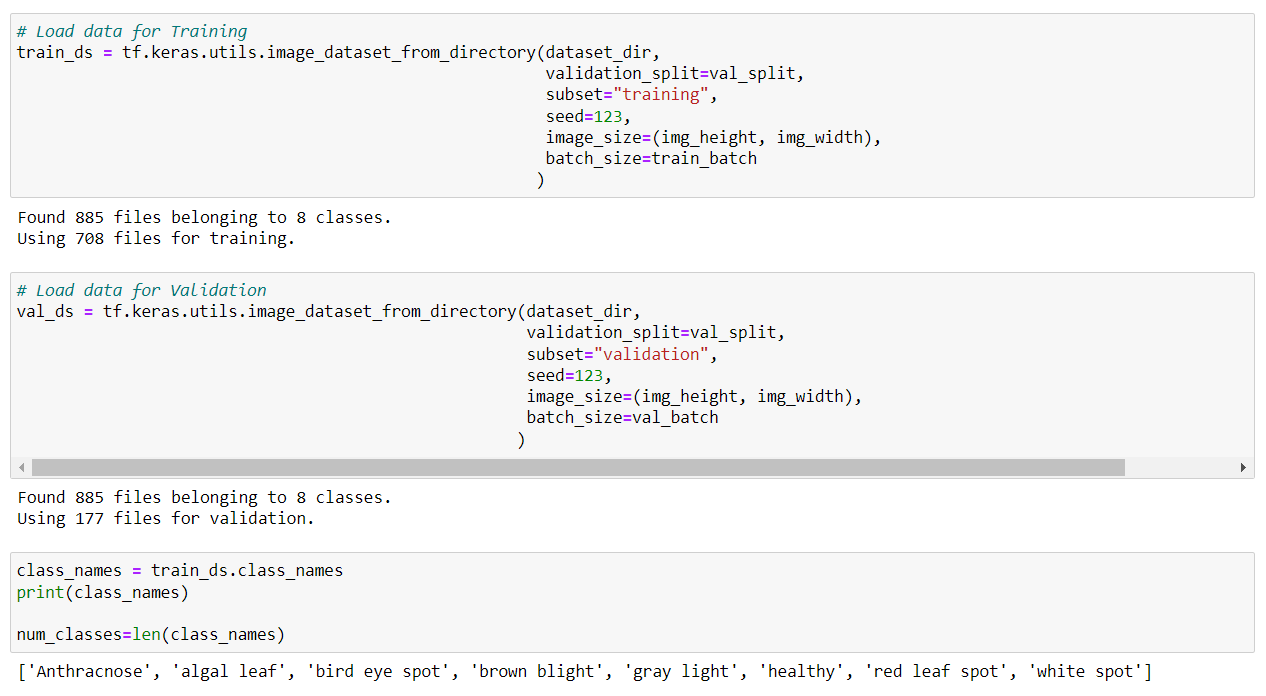
The success of the Tea Leaf Disease Detection project opens avenues for applying similar technologies to other agricultural sectors, promoting sustainable farming practices and crop health management globally. By learning from current achievements and focusing on areas for improvement, future projects can further advance agricultural innovation and support farming communities worldwide.

**Appendices:**

**Sample Code Snippets:**

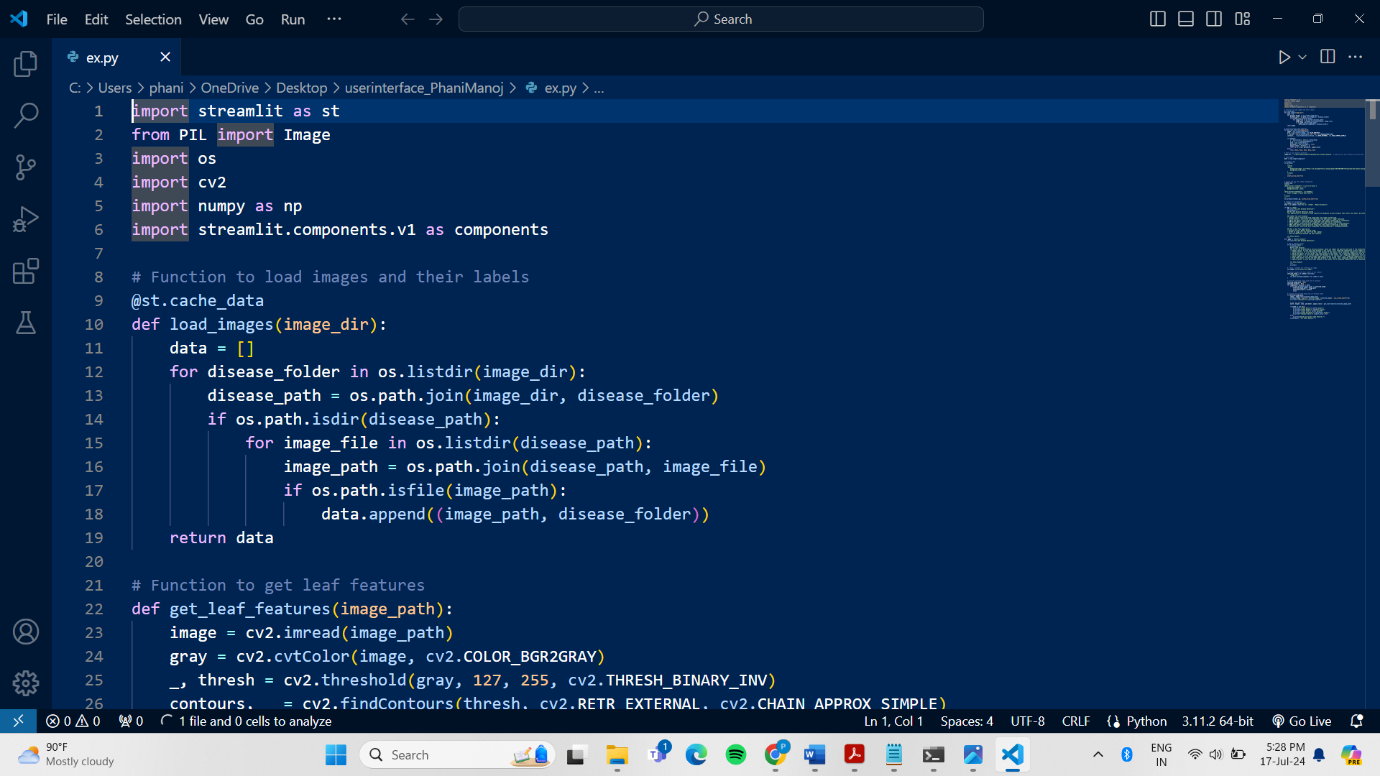
Model Training & Prediction: Python code snippets demonstrating the setup and training of the convolutional neural network (CNN) model using TensorFlow/Keras.

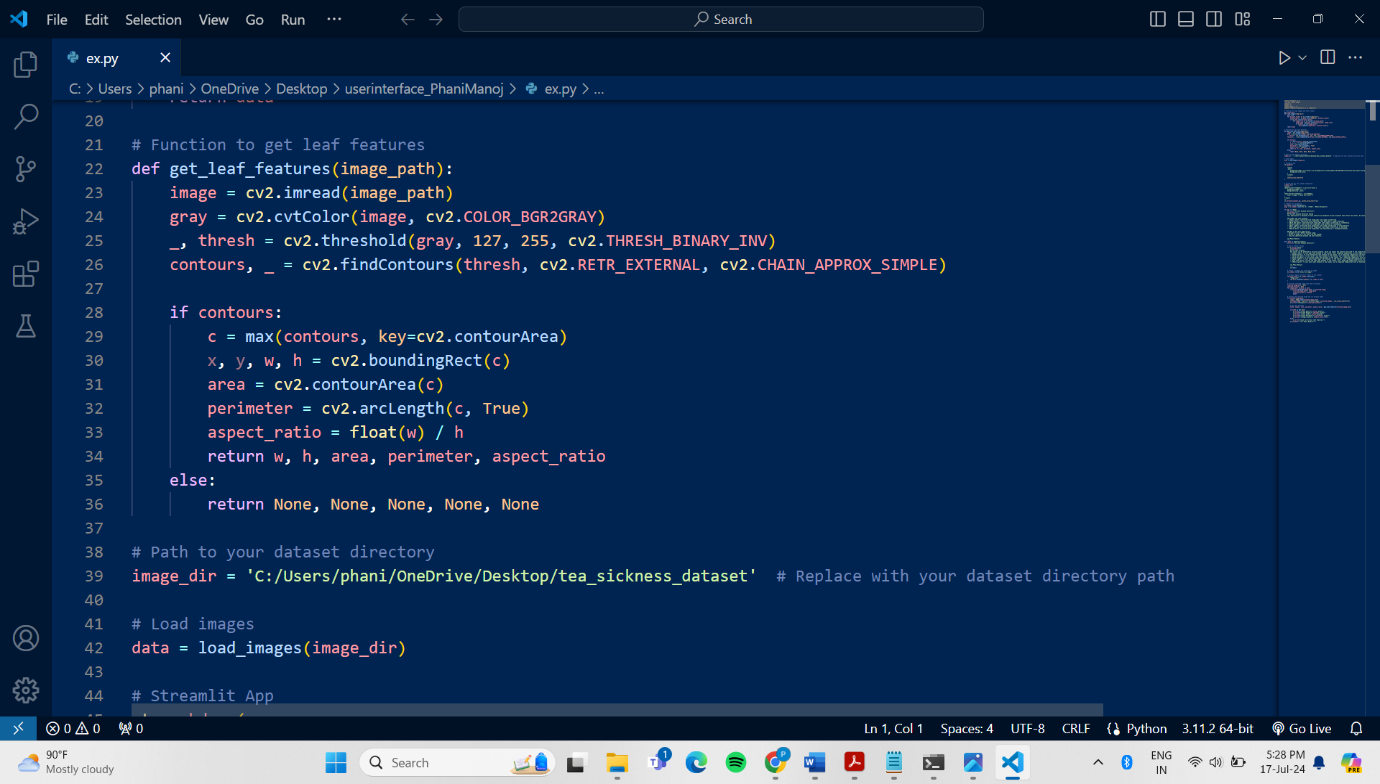


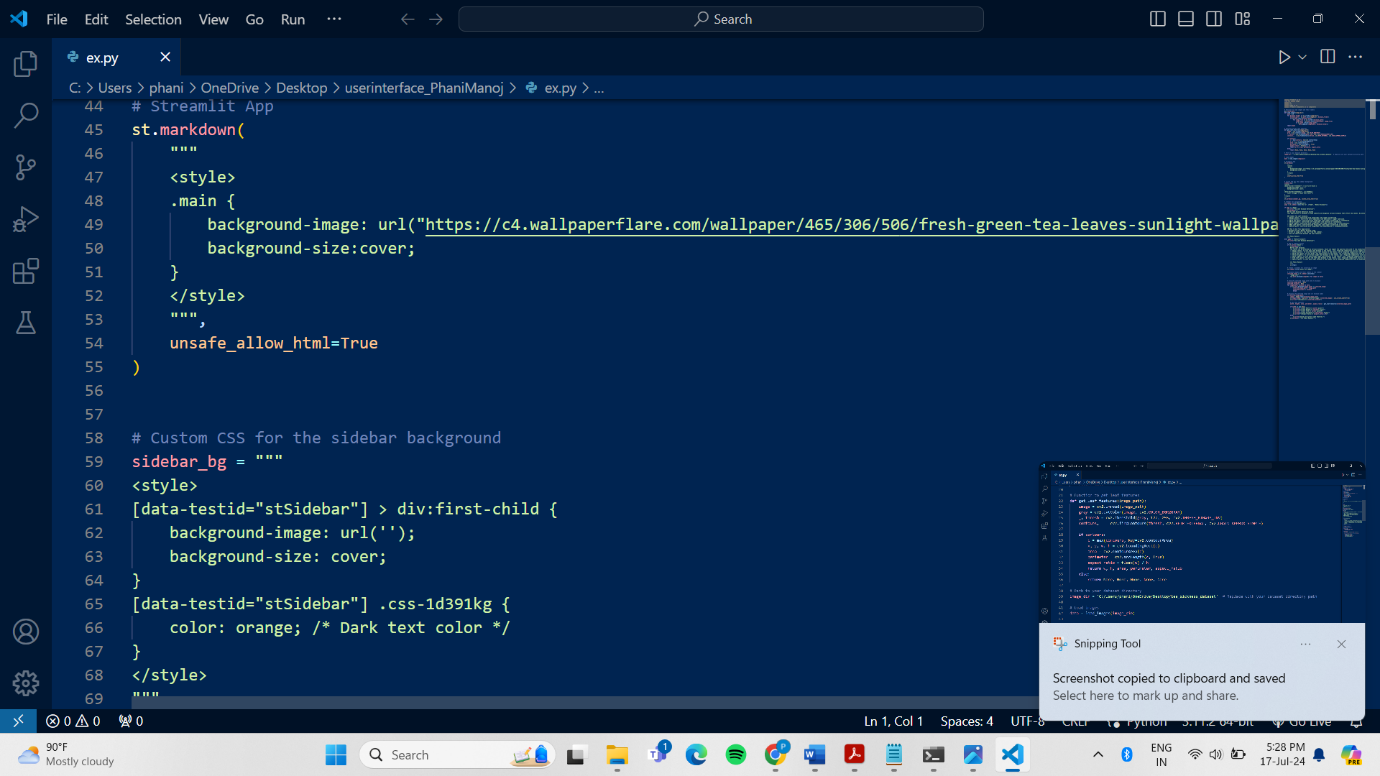




**Streamlit Application Setup:** Instructions and code snippets for setting up the streamlit web application to serve the trained model.

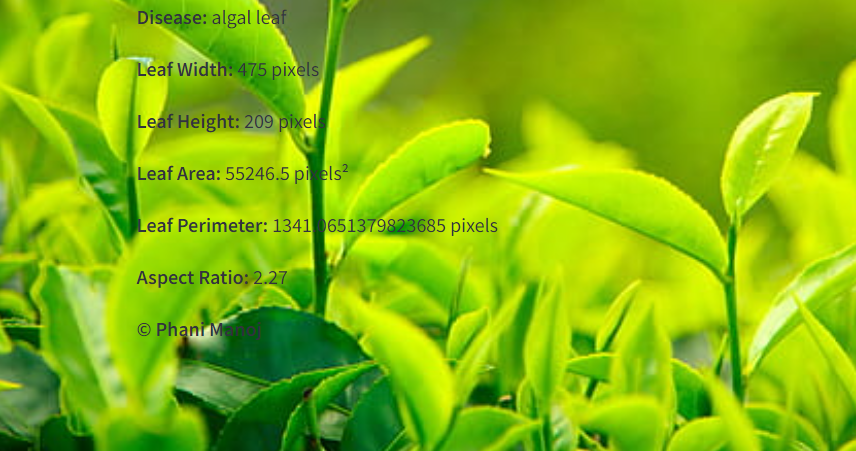






**User interface:**





**Research References:**

* Sally A. Miller, Fen D. Beed, & Carrie Lapaire Harmon. (2009). Plant Disease Diagnostic Capabilities and Networks. *Annual Review of Phytopathology*, 47, 15-38. https://doi.org/10.1146/annurev-phyto-080508-081743
* Vijai Singh, Namita Sharma, Shikha Singh. (2020). A review of imaging techniques for plant disease detection. *Artificial Intelligence in Agriculture*, 3, 10-22. https://doi.org/10.1016/j.aiia.2020.10.002
* <https://www.kaggle.com/code/rizqyad/tea-leaves-disease-classification>
* https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00863-9
* https://acsess.onlinelibrary.wiley.com/doi/full/10.1002/uar2.20053
* https://issuu.com/irjet/docs/irjet-v10i3171

**Thank You!!**