**1. Introduction**

1.1 Application

This web application is part of a broader research initiative aimed at improving plant disease detection in agricultural settings. Plant diseases significantly impact crop yields and can lead to massive economic losses. Early detection is critical to mitigate these impacts. By leveraging Convolutional Neural Networks (CNNs), this app allows realtime classification and detection of diseases from plant leaf images, providing a vital tool for agricultural professionals to identify issues early and take action.

1.2 Significance

The app bridges theory and practice by applying CNN based deep learning models to solve a realworld problem: plant disease detection. It demonstrates how advanced image classification techniques can provide tangible benefits for farmers, agricultural consultants, and researchers. The app's realtime capabilities, combined with its user friendly interface, offer a practical, accessible tool for diagnosing plant diseases in the field.

**2. Technologies Used**

2.1 Python: Python was selected due to its extensive ecosystem of machine learning and image processing libraries, making it ideal for handling realtime image classification tasks in agricultural settings.

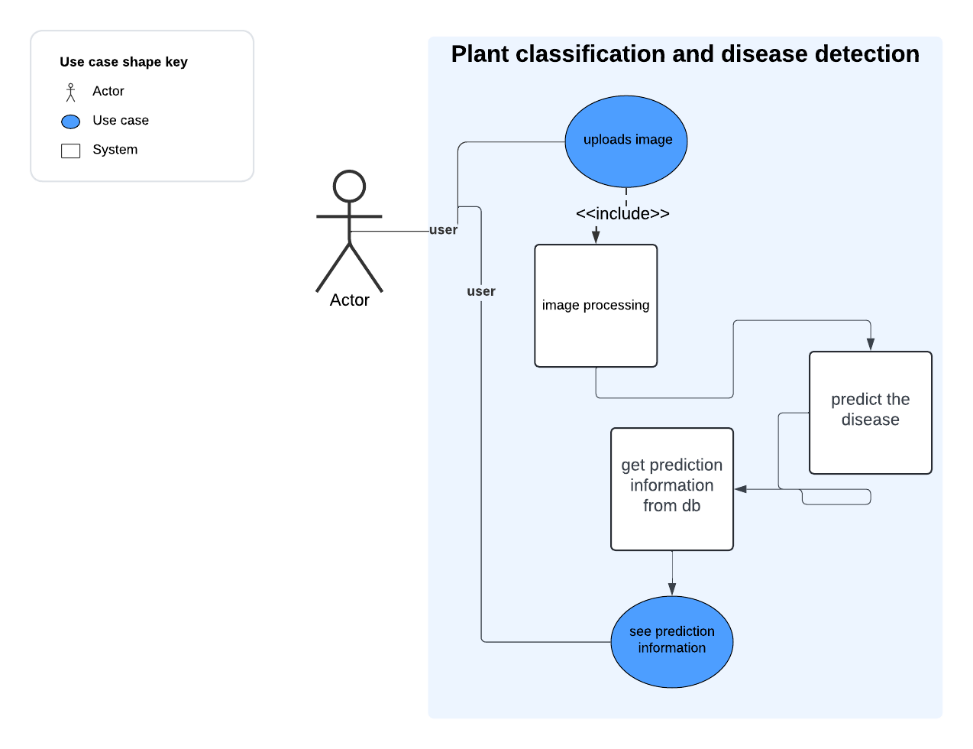
2.2 TensorFlow: TensorFlow was the primary framework used to build and train the CNN models. It provides efficient, scalable deep learning capabilities, making it suitable for handling large datasets and complex models, essential in detecting and classifying plant diseases from images.

2.3 Streamlit: Streamlit was used to build the web interface, providing a simple yet powerful platform for interacting with the model. In agricultural contexts, the ease of use is critical, and Streamlit allows farmers and other nontechnical users to upload images, view results, and make decisions without needing to understand the underlying technology.

**3 Relating Tools to User Experience**

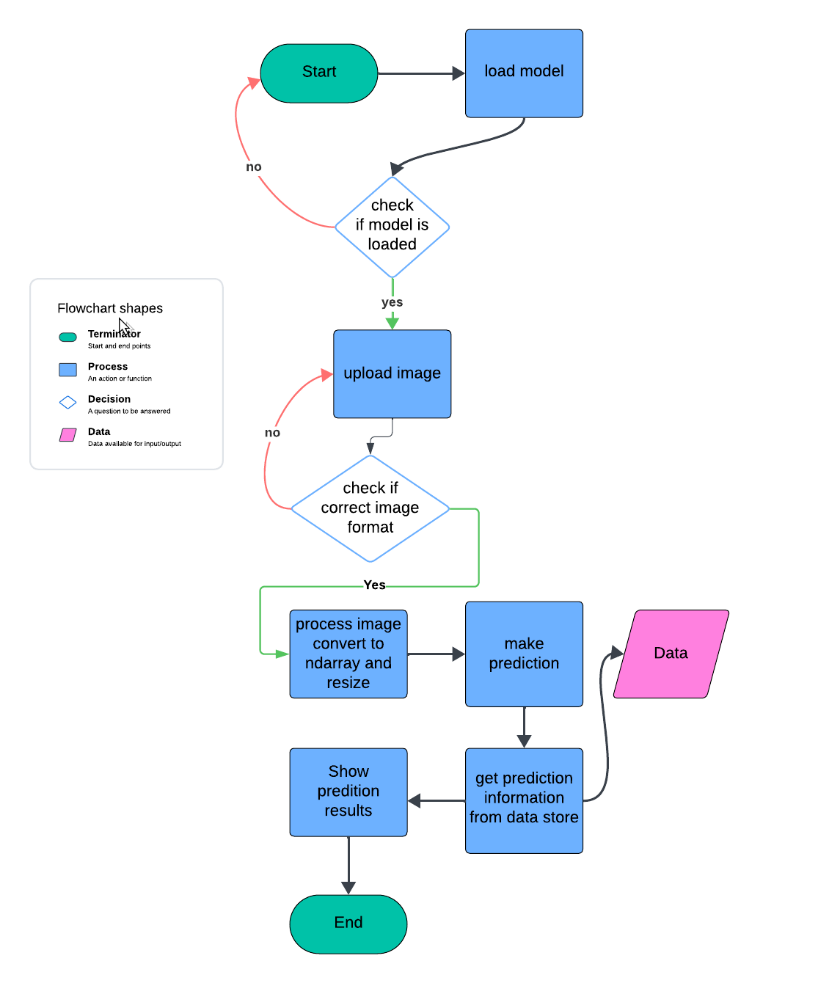
The intuitive design of Streamlit ensures that users, even those with minimal technical knowledge, can easily interact with the app. Farmers can upload images of plant leaves and receive immediate feedback on potential diseases, along with prevention and treatment options. This accessibility ensures that the app can be used in diverse environments, from research institutions to rural farming communities.

3.1 Use case diagram



In this system, there is only one actor: the user. The user interacts with the system by uploading an image. The system processes the uploaded image and then passes it to the model for predictions. Once the model generates the results, the system sends the predictions back to the user, providing them with the diagnosis and relevant information.

3.4 Flowchart



The application begins by loading the trained CNN model from storage. If the model loads successfully, the app proceeds to the next step, which is the image upload interface. If the model fails to load, the application will not run and will require a restart.

Once an image is uploaded, the app checks to ensure the image is in the correct format. If the format is incorrect, the user is prompted to upload a valid image. When the uploaded image is in the correct format, it is processed using the Pillow library to load it. The image is then converted into an array using NumPy for further manipulation.

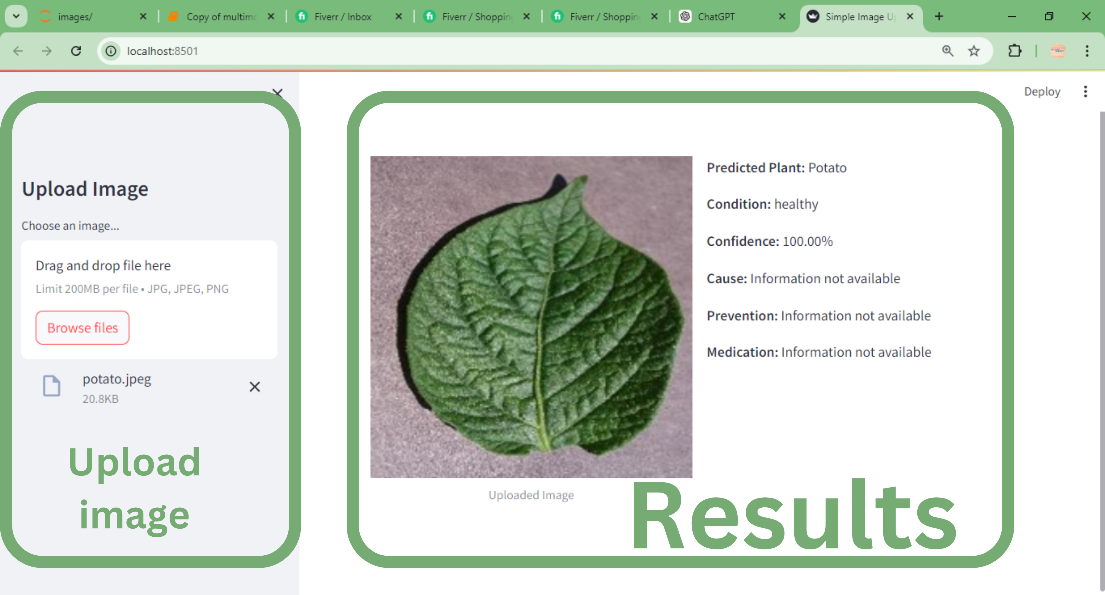
After conversion, the image is resized to 244,244 the dimensions expected by the model. Once resized, the processed image is sent to the model for predictions.

4. User Interaction with the App

How Users Interact with the app

Users interact with the app through a simple and intuitive web interface. After launching the app locally, they are presented with an option to upload an image of a plant leaf. Once the image is uploaded, the app preprocesses it and passes it to the CNN models for plant identification and disease classification. The results, including the plant name, disease condition, confidence score, and suggested prevention or treatment measures, are then displayed in realtime.

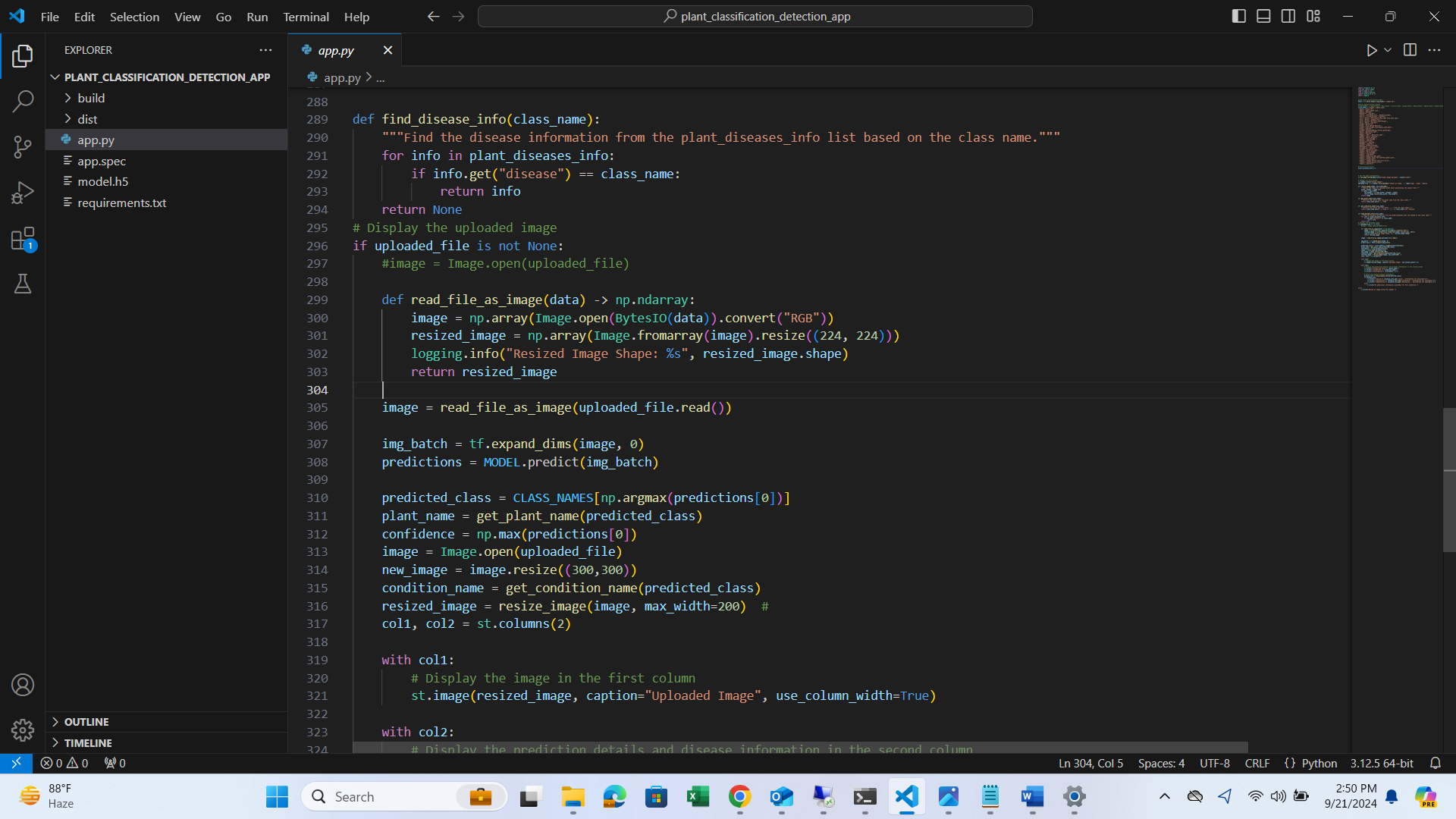
Screenshots of User Interface



**5. Backend Integration with CNN Models**

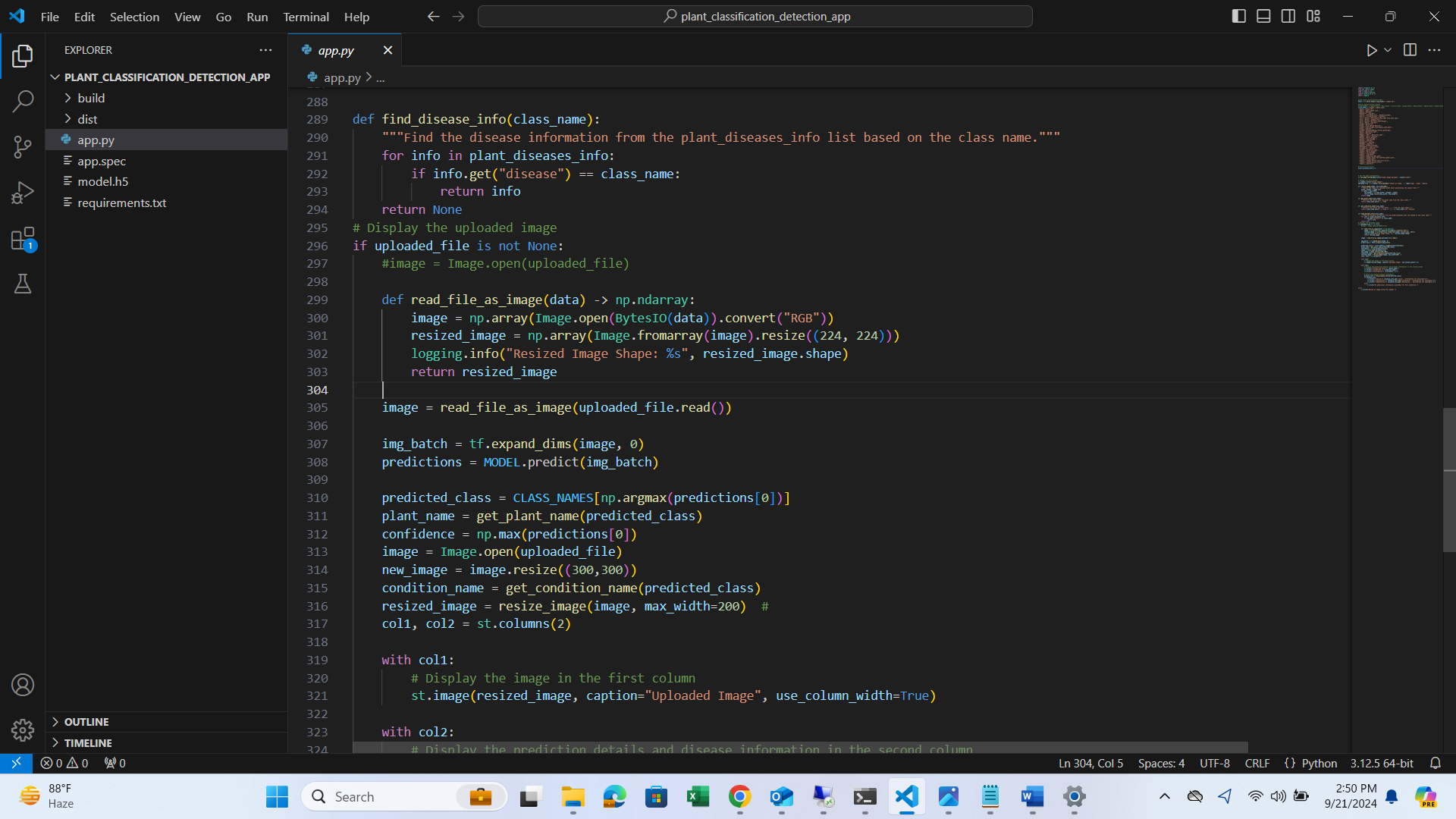
5.1 Image Processing and Model Integration

Preprocessing: After an image is uploaded, it undergoes preprocessing using the PIL library to ensure it is resized, normalized, and formatted correctly for input into the CNN models. The image is passed through two models: one for plant identification and another for disease classification.



The code above shows the function that is called when an image is uploaded so that the image is first converted to an ndarray then resized to 224, 244 which are the dimensions that the model understands

5.2 Model Workflow: The CNN model identifies the plant species based on the leaf image. And returns the results



The above show how the processed image is passed to the model for prediction by calling the ***model.predict()*** methodResults Presentation: The results are presented to the user in an easytoread format, including the plant name, disease condition, confidence score, condition cause and recommendations for prevention and treatment.

After we get these results we then pass them to the streamlit ui for the user to see

5.3 Data Flow Diagram

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step 1 | Step 2 | Step 3 | Step 4 | Step 5 | Step 6 |
| Image is uploaded and sent to backend | Image is processed by converting to ndarray and resized by 244,244 | The resized image is passed to the model for prediction | Model returns the results | The result is sent to the data store to return more information | The results I sent t the frontend |

**6 Performance Metrics**

Processing Time: On average, the app processes and returns results within 25 seconds after the image is uploaded, depending on the system's specifications (e.g., CPU and RAM). This fast response time makes the app highly usable in realtime agricultural settings where quick decisions are critical.

**7. Application Results**

Expanded Results Discussion

7.1 The results page offers the following information:

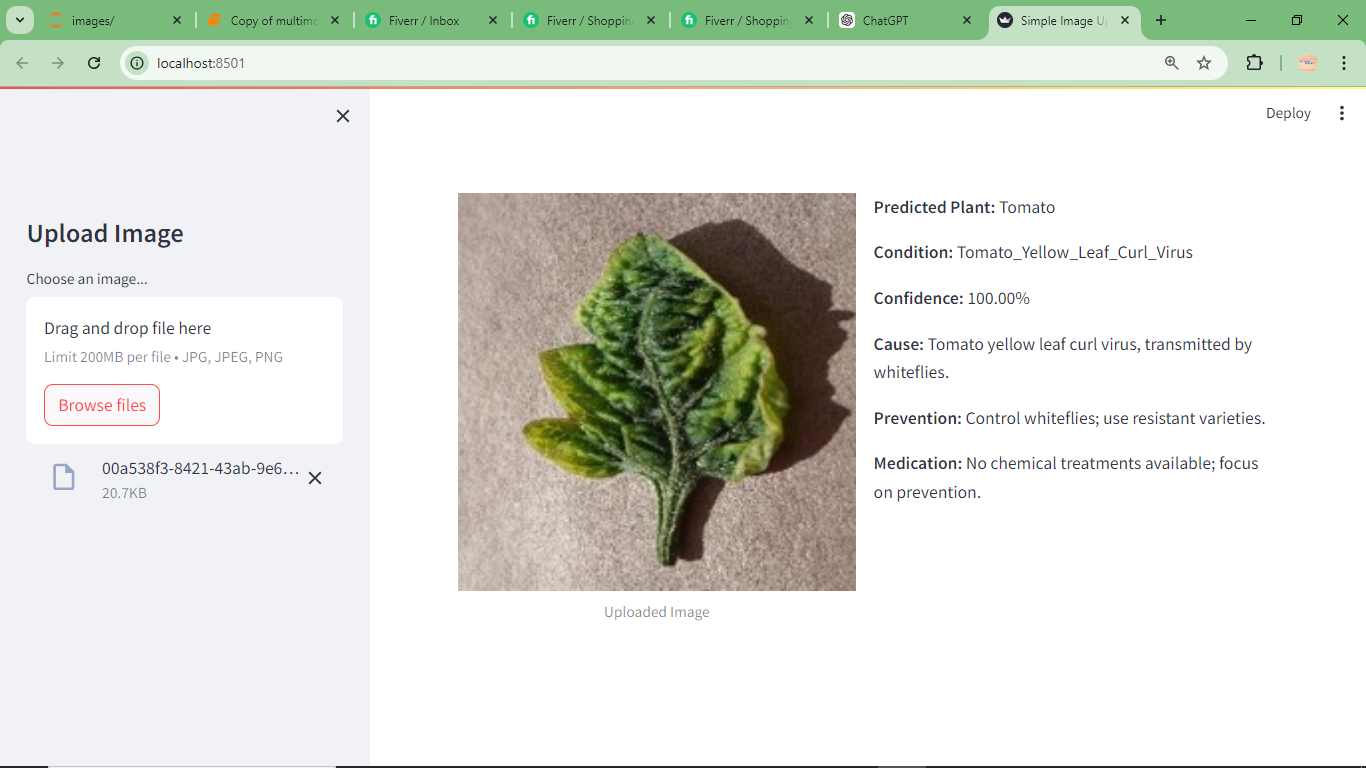
Plant Name: The app identifies the plant species from the image, which is crucial for accurate disease diagnosis.

7.2 Disease Condition: The app provides a detailed diagnosis of the disease, if present, along with the probability (confidence score) of the diagnosis.

Confidence Score: The app shows a percentage that reflects the model’s confidence in its prediction.

7.3 Prevention and Treatment: Users are given actionable advice on how to prevent the disease from spreading or how to treat it.

These results allow farmers and agricultural experts to make informed decisions quickly, improving crop management and yield protection.



Example of how results are presented to the user.

**8. Performance and Deployment**

Current Local Setup

The app is currently deployed locally, meaning it can only be accessed on the local machine or network where it is installed. This setup is beneficial in remote or rural areas where internet connectivity may be limited. The app's performance depends on the local machine's specifications, including CPU, RAM, and GPU if available, which influences the speed of image processing and result delivery.

8.1 Advantages of Local Deployment:

1. Offline Capability: Since it runs locally, the app doesn't require an internet connection, making it suitable for rural environments where network access is limited or unreliable.

Development and Testing: Local deployment is ideal for development and testing purposes, where a single user can test the app without needing cloud infrastructure.

2. Scalability and Future Deployment

While the current local setup limits scalability and user access to a single machine, the app has been designed with scalability in mind. Future versions could be deployed on cloud platforms such as render, streamlit, hugging face , own server and other platforms that can support streamlit app

3. Multiuser Access: Cloud deployment would allow multiple users to access the app simultaneously from different locations.

4. Realtime Inference at Scale: Cloudbased deployment would enable faster processing and more efficient handling of large datasets, making the app suitable for broader use in larger agricultural organizations or cooperatives.

9. Conclusion

This project demonstrates the practical application of CNN models in detecting plant diseases in realtime through a simple web interface. The use of Python, TensorFlow, and Streamlit has enabled the creation of a tool that is both powerful and accessible to nontechnical users such as farmers. The current local deployment serves as a robust prototype for field use in rural settings, with future potential for cloud deployment to scale its usability across multiple users. By offering realtime diagnostic results, actionable prevention measures, and offline capabilities, this app has the potential to significantly improve disease management in agriculture, leading to healthier crops and higher yields.