

Instructions to Run App and Details about the Application

Team Members:

Vishwani Bodapatla (Product Owner, Researcher and developer)

Arvind Devarajan (Reviewer, Tester and part of presentation)

Sunil Peela (Reviewer, Tester, Presentation Owner)

Run Streamlit Application:

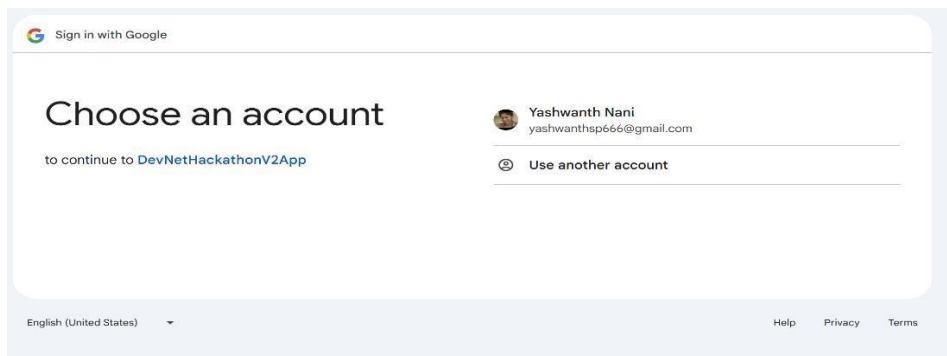
1. Install streamlit in your computer (in my case I used anaconda prompt).
2. Activate streamlit by using command: streamlit activate
3. Run streamlit app by using command: streamlit run (path) /DevNetHackathonV2App.py

Authenticate (You need to authenticate for the first Time):

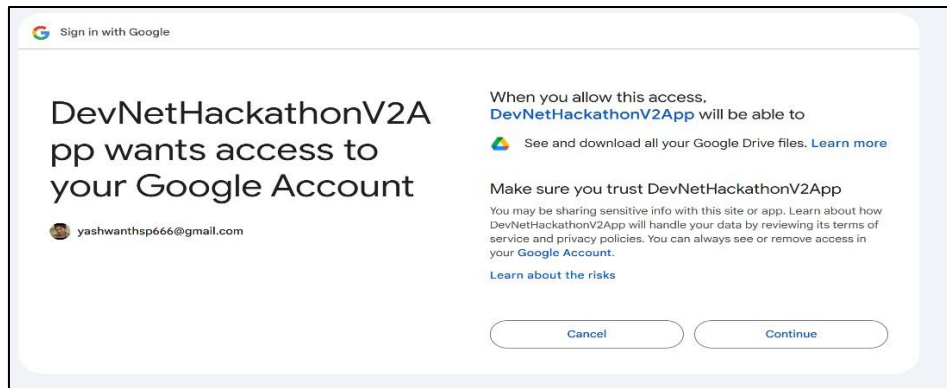
1. Once you run the streamlit application in your Anaconda prompt, you will be redirected to this page. Provide your email address and click on the 'Authenticate and Download Images' button.

The screenshot shows a web application titled "Google Drive Image Downloader and Classifier". It has a text input field labeled "Enter your email for authentication" with the email "yashwanthsp66@gmail.com" entered. Below the input field is a button labeled "Authenticate and Download Images". The interface is clean and modern, with a white background and a dark header.

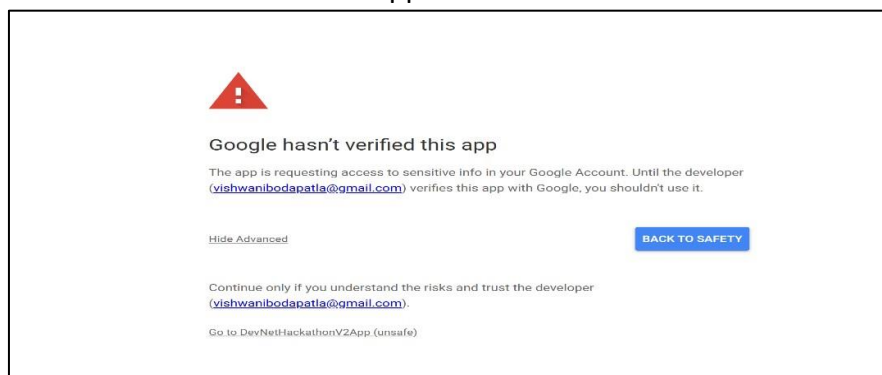
2. Next click on your preferred email address.

The screenshot shows the Google account selection screen. It has a header "Sign in with Google" and a main heading "Choose an account". Below the heading, it says "to continue to DevNetHackathonV2App". On the right, there is a list of accounts with a profile picture and the name "Yashwanth Nani" and email "yashwanthsp66@gmail.com". Below the list is a button labeled "Use another account". The interface is clean and modern, with a white background and a dark header.

3. Next click on continue.



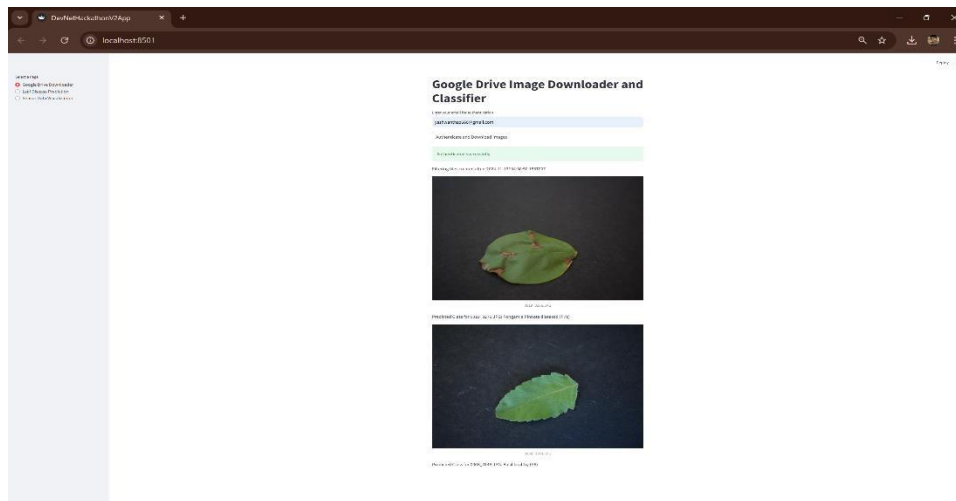
4. Click on advance in the next page. Under Advanced, click on 'Go to DevNetHackathonV2App'



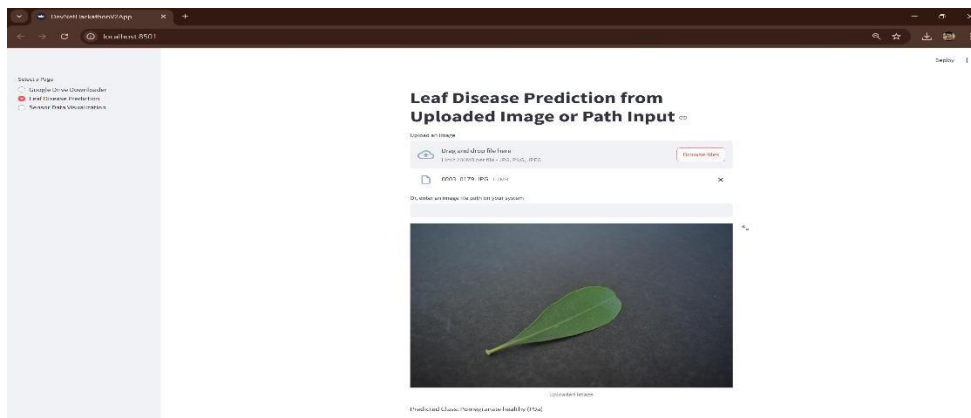
How does this Streamlit App is Working:

Page1(Google Drive Downloader): The program first authenticates the user using Google OAuth to allow access to Google Drive. After successful authentication, it retrieves the newly added images from the user's Drive. The images are then downloaded into a temporary file called **tempDownloadFile**.

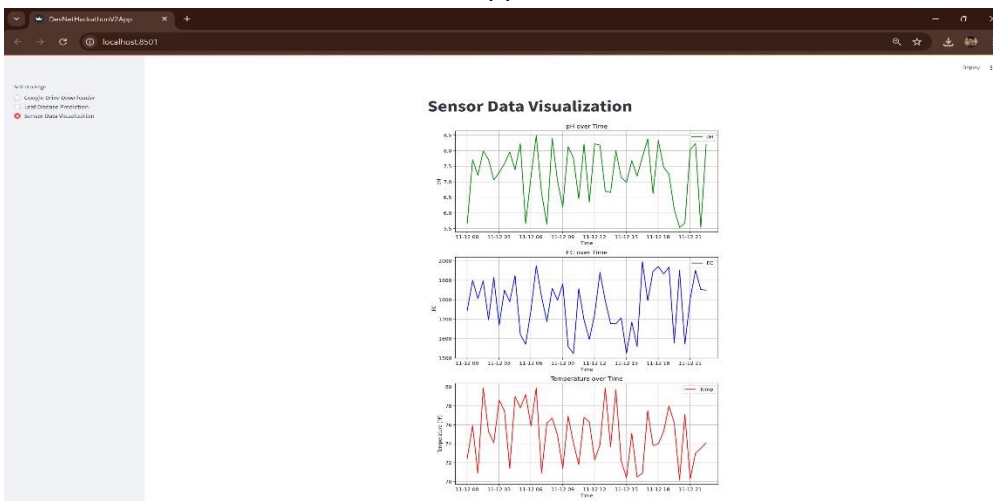
Once the images are downloaded, the program uses the trained transfer learning model to classify each image. The predicted classes are then displayed alongside the images on the page. This entire process allows for seamless integration of Google OAuth for authentication, image retrieval from Google Drive, classification using a pre-trained model, and displaying the results to the user.



Page2: In the second page, you can upload the individual file and let the trained model classify if the leaf is healthy or diseased.



Page3: In sensor data Visualization page, you view the pH over time, EC over time and Temperature over temperature. This data is fake and autogenerated every 30 minutes from 12:00 AM to the time, the streamlit application is executed.



About the Trained Model (Technical points):

Session Management and GPU Configuration:

- The code begins by ensuring that any existing TensorFlow sessions are properly closed to prevent session conflicts. It then configures the GPU memory usage, setting the fraction of GPU memory for TensorFlow and enabling dynamic memory growth to optimize resource utilization during training.

2. Image Preprocessing and Augmentation:

- The program uses ImageDataGenerator to preprocess and augment the images for training and validation. For training data, it applies various transformations like rotation, width/height shift, shear, zoom, and horizontal flipping to improve model generalization. For validation data, only rescaling is applied to ensure consistent evaluation.

3. Loading and Configuring InceptionV3 Model:

- The InceptionV3 model is loaded with pre-trained ImageNet weights, excluding the top classification layer (include_top=False). The base model layers are frozen to prevent updates during training. A custom classifier, consisting of a global average pooling layer, a fully connected dense layer, dropout for regularization, and a softmax output layer, is added on top.

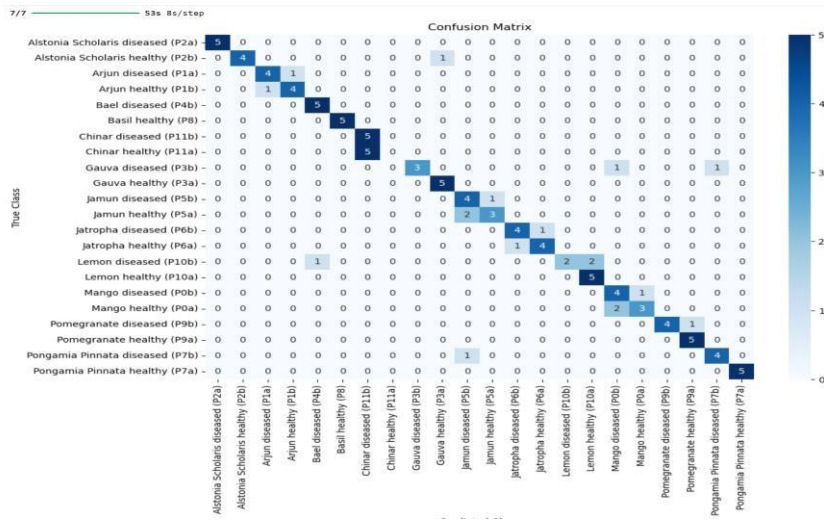
4. Model Compilation and Training:

- The model is compiled with the Adam optimizer (with a learning rate of 0.001) and categorical cross-entropy loss for multi-class classification. Early stopping is set to monitor the validation loss and stop training if no improvement is seen after 5 epochs. Additionally, ModelCheckpoint is used to save the best model based on validation performance.

5. Model Evaluation and Visualization:

- After training, the model's performance is evaluated by plotting the training and validation loss/accuracy over epochs. A confusion matrix is generated to visualize the performance of the model on the validation set. The classification report, including

precision, recall, and F1-score for each class, is also printed to assess the model's effectiveness.



6. Model Saving:

- The trained model is saved in two formats: .keras and .h5 to ensure compatibility with different TensorFlow versions and frameworks. The saved model can later be reloaded for inference or fine-tuning.