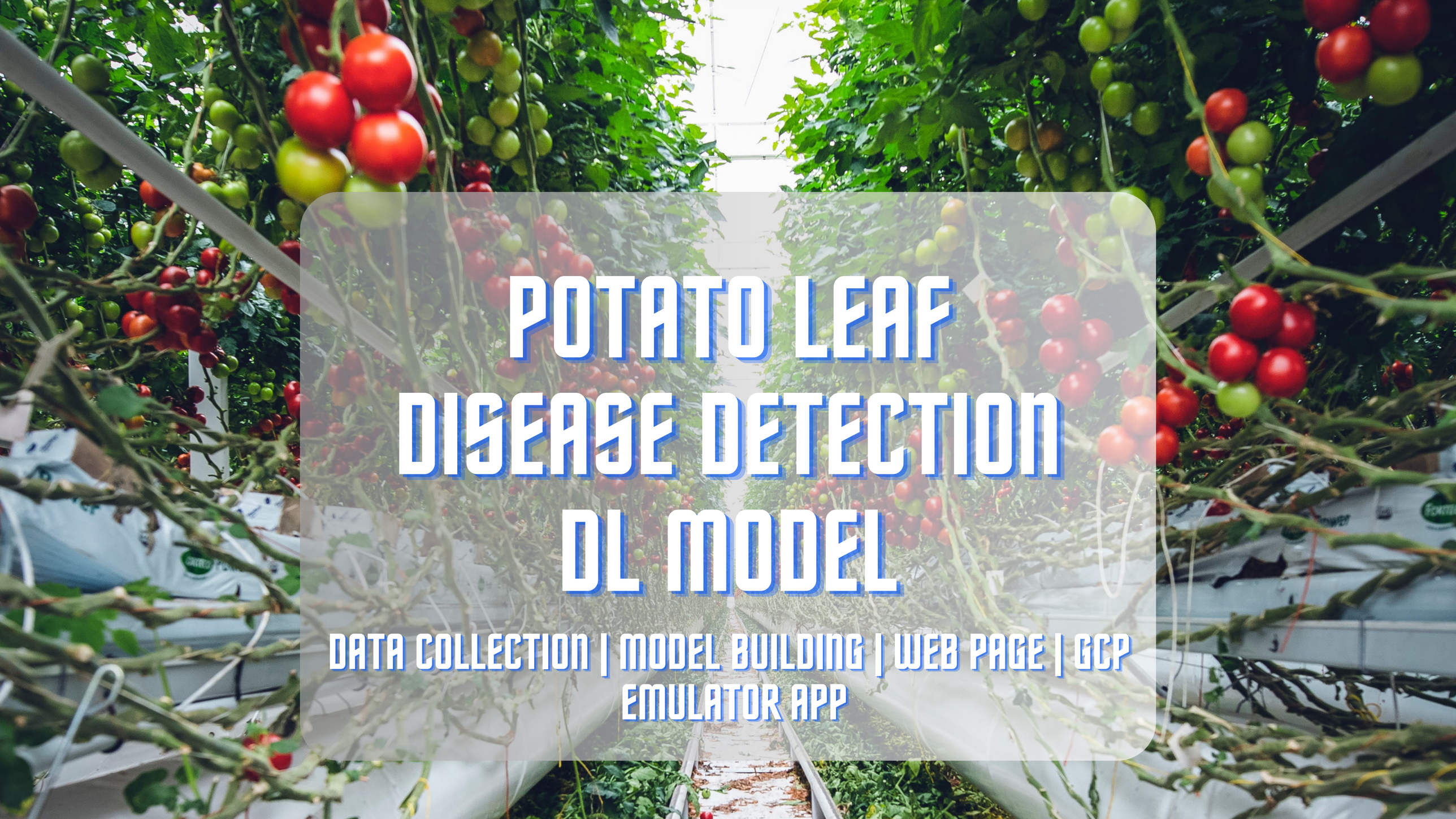


POTATO LEAF DISEASE DETECTION DL MODEL

DATA COLLECTION | MODEL BUILDING | WEB PAGE | GCP EMULATOR APP

Problem Statement

POTATO LEAF DISEASES SUCH AS EARLY AND LATE BLIGHT CAUSE SIGNIFICANT ECONOMIC LOSSES FOR FARMERS AROUND THE WORLD. TO ADDRESS THIS PROBLEM, THE PROJECT AIMS TO HARNESS INNOVATIVE TECHNOLOGY TO DEVELOP EFFECTIVE AND SUSTAINABLE SOLUTIONS FOR POTATO DISEASE MANAGEMENT

USING DEEP LEARNING, THIS AGRO-TECH SOLUTION PREDICTS AND PREVENTS POTATO LEAF DISEASES, ENSURING CROP HEALTH AND FARMER PROSPERITY

Data Collection & Model Building
Dataset : Kaggle Dataset | Plant Village

Data Preprocessing | Resizing, Augmentation

Dataset Partitioning | Splitting for subsets

Model Architecture | CNN with ReLU

Model Compilation | Compile with settings.

Model Training | Train for learning

Model Evaluation | Assess model performance

Results Visualization | Display training results

Image Prediction | Make predictions on images

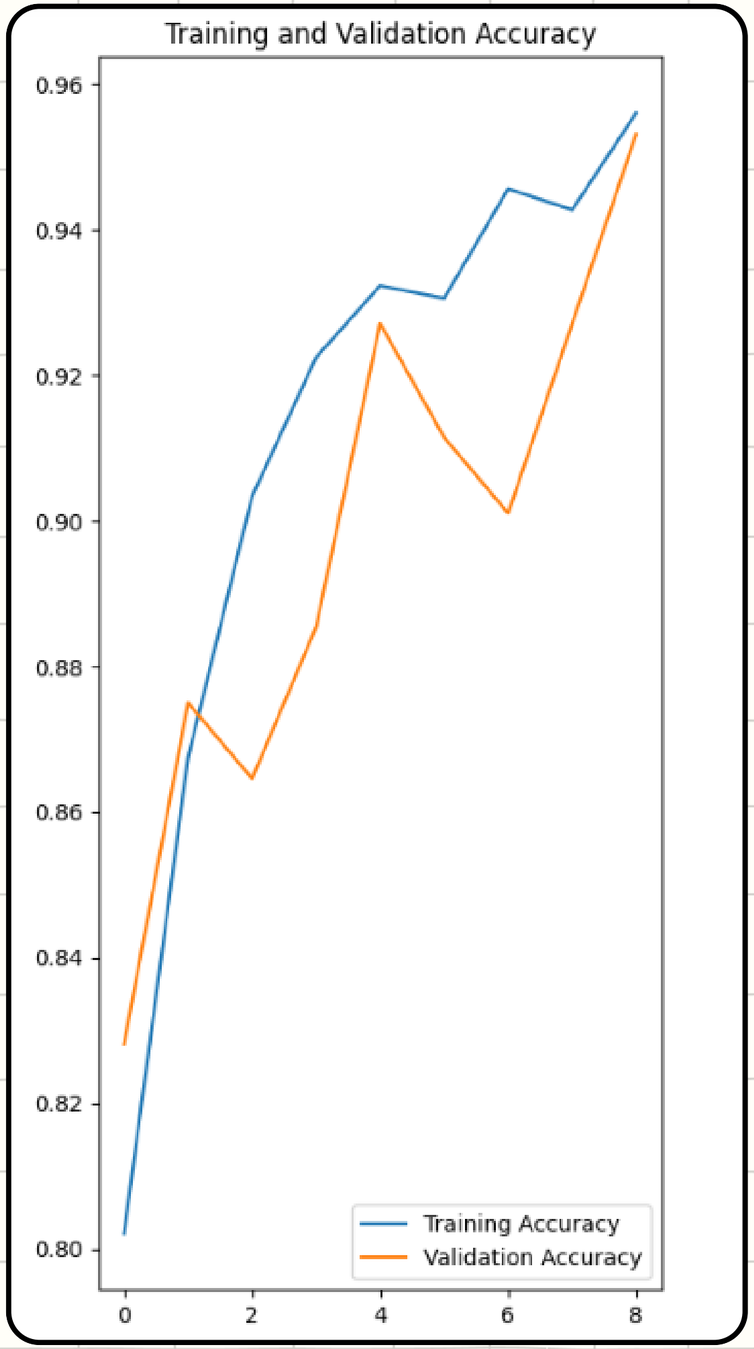
Model Export | Save the trained model

FAST API | Create a user interface

Load Trained Model | Load the saved model

Make Predictions | Use the model for predictions

Training & Validation Accuracy



Model Prediction

CNN (Convolutional Neural Network) | Deep learning architecture for image processing.
ReLU (Rectified Linear Unit) | Activation function that introduces non-linearity.
Softmax | Activation function for multi-class classification.
Data Augmentation | Expands training data with image transformations.

Back End - FAST API

FastAPI for Building APIs

FastAPI is a Python web framework designed for building APIs quickly and efficiently. It allows developers to create RESTful web services with minimal code while maintaining high performance.

Automatic Data Validation

FastAPI provides automatic data validation and serialization, reducing the risk of common API errors. It uses Python-type hints to validate incoming data and generate accurate API documentation.

Async Support

FastAPI fully supports asynchronous programming, making it suitable for handling concurrent requests and I/O-bound operations. This feature enhances the scalability and responsiveness of web applications built with FastAPI.

Front End React

Material-UI for UI Components

The script employs Material-UI, a UI framework for React, integrating components like AppBar, Toolbar, and Card to create a modern and visually appealing user interface.

Image Processing with Axios

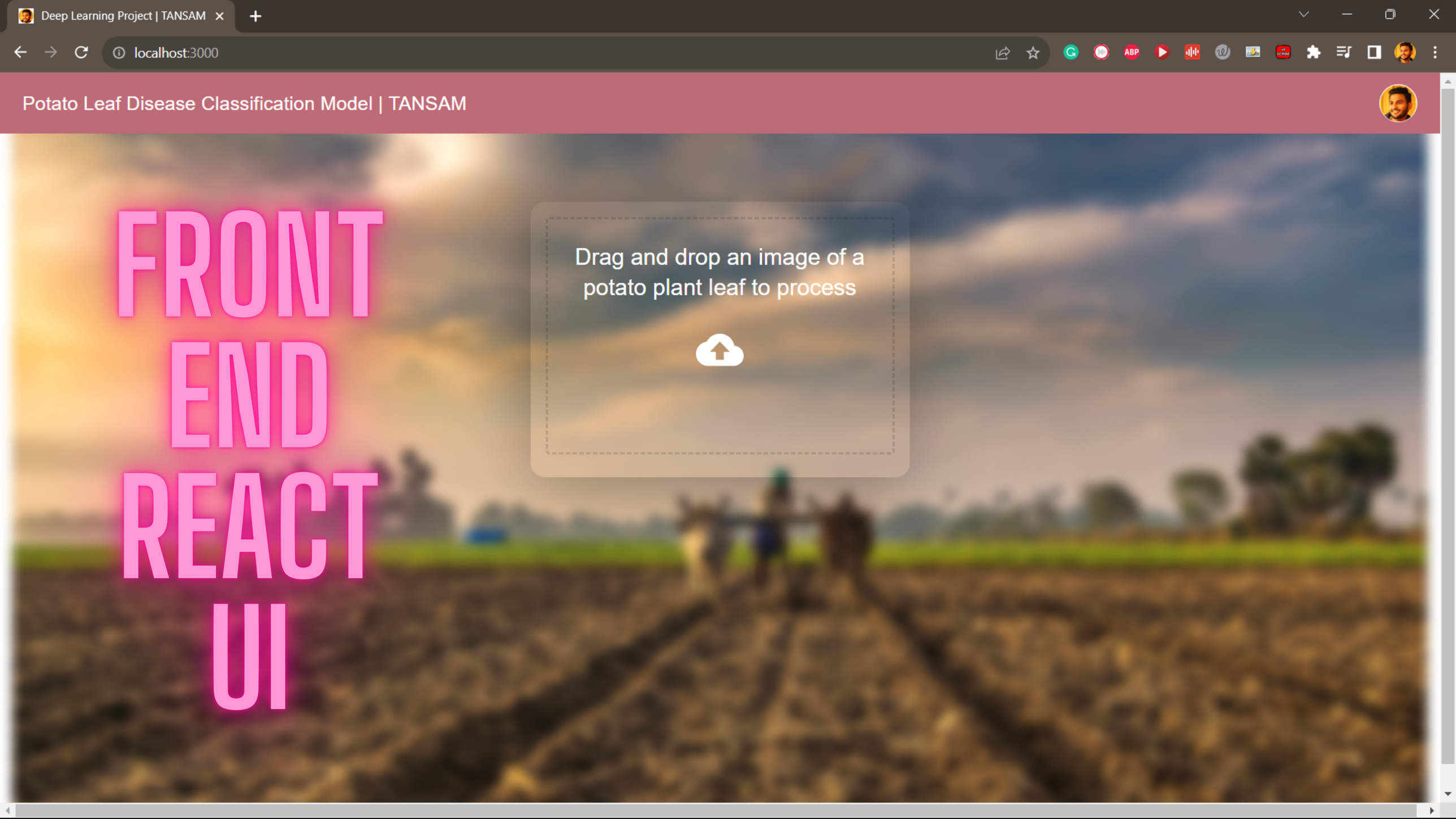
It utilizes Axios, a promise-based HTTP client, to send image data to a local server for image classification, showcasing an asynchronous approach to handling data.

Responsive Design Principles

The script follows responsive design principles, ensuring the web application's adaptability to various screen sizes, enhancing user experience across devices.

State Management with useState

React's useState hook manages component state, enabling dynamic updates of image data, previews, and loading indicators as users interact with the application.


Deep Learning Project | TANSAM
localhost:3000
Potato Leaf Disease Classification Model | TANSAM
FRONT END REACT UI
Drag and drop an image of a potato plant leaf to process

Deep Learning Project | TANSAM

localhost:3000

Potato Leaf Disease Classification Model | TANSAM

SAMPLE : EARLY BLIGHT DISEASE AFFECTED LEAF

CONFIDENCE PERCENTAGE OF 98.99% RETURNED

MODEL PREDICTION IS GOOD

Label:	Confidence:
Early Blight	98.99%

× CLEAR

GCP Cluod Deployemnt & Testing
Project ID | GCP Bucket | .h5 Model

GCP | Google Cloud Platform | Setting up the environment

Started creating a GCP account and project, setting up a unique bucket for model storage, and installing the Google Cloud SDK.

Code Development

A Python script for a cloud function that downloads the pre-trained ML model, preprocesses input images and performs predictions using TensorFlow/Keras.

Deployment

Deployed the cloud function using the Google Cloud SDK, ensuring necessary libraries were included, and enabled the Google Cloud API whenever prompted.

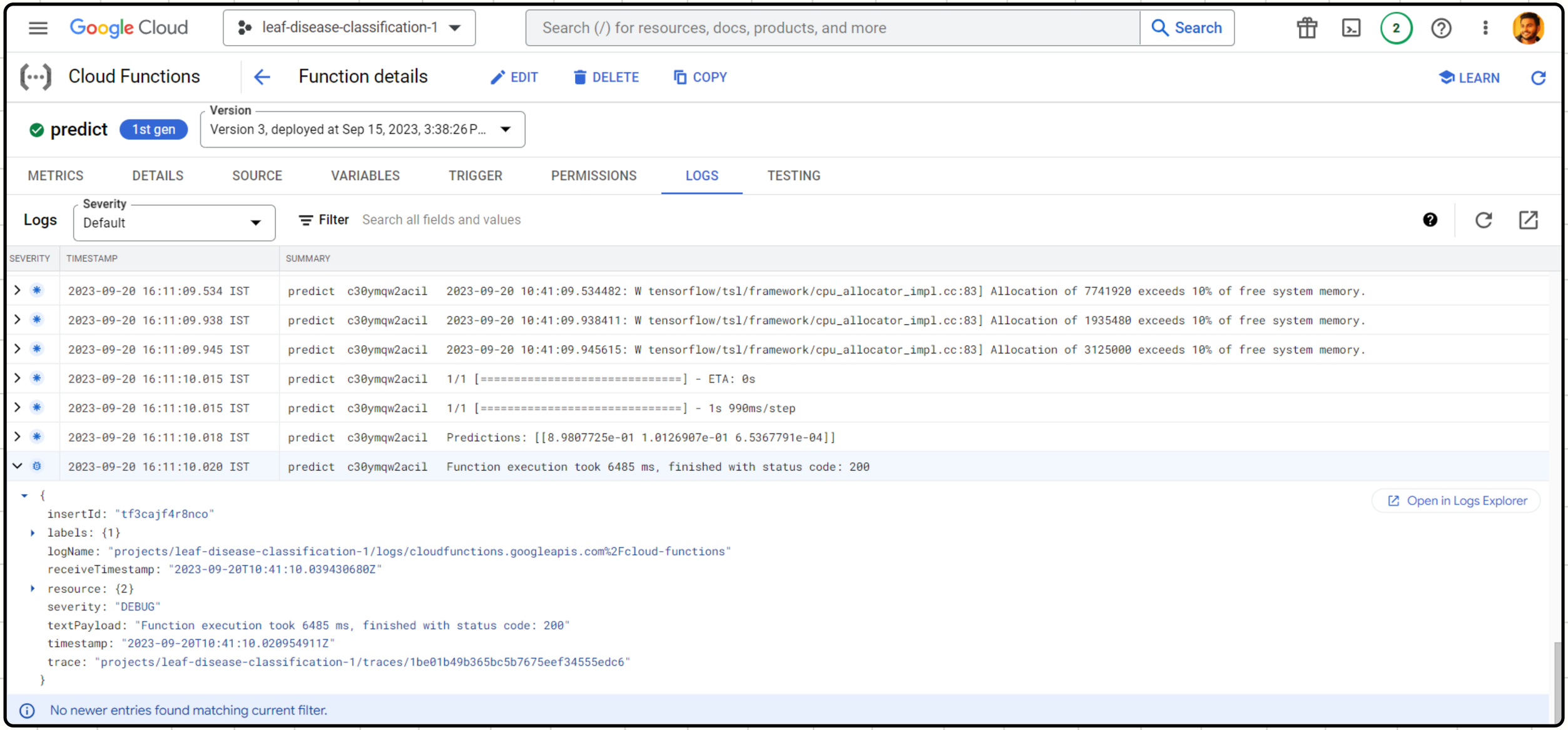
Testing

Used Postman to send image data as form data to the deployed function, that analyze the response for predictions, and ensures the function works correctly.

Function & URL

predict & https://us-central1-leaf-disease-classification-1.cloudfunctions.net/predict

GCP Deployment



Postman Invoke

https://us-central1-leaf-disease-classification-1.cloudfunctions.net/predict

Save

POST https://us-central1-leaf-disease-classification-1.cloudfunctions.net/predict

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
file	early_blight_1.jpg		
Key	Value	Description	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 34.54 s Size: 332 B Save as Example

Pretty Raw Preview Visualize JSON


```
{
    "class": "Early Blight",
    "confidence": 89.81
}
```

Mobile App Deployment

React Native Setup

Prepared the development environment by installing the necessary tools and dependencies for React Native, ensuring the platform-specific instructions.

Configuring Mobile App

Cloned the mobile app repository from GitHub, configure environment variables (like the Google Cloud Functions link), and install required Node.js modules using "yarn install."

Emulator Setup

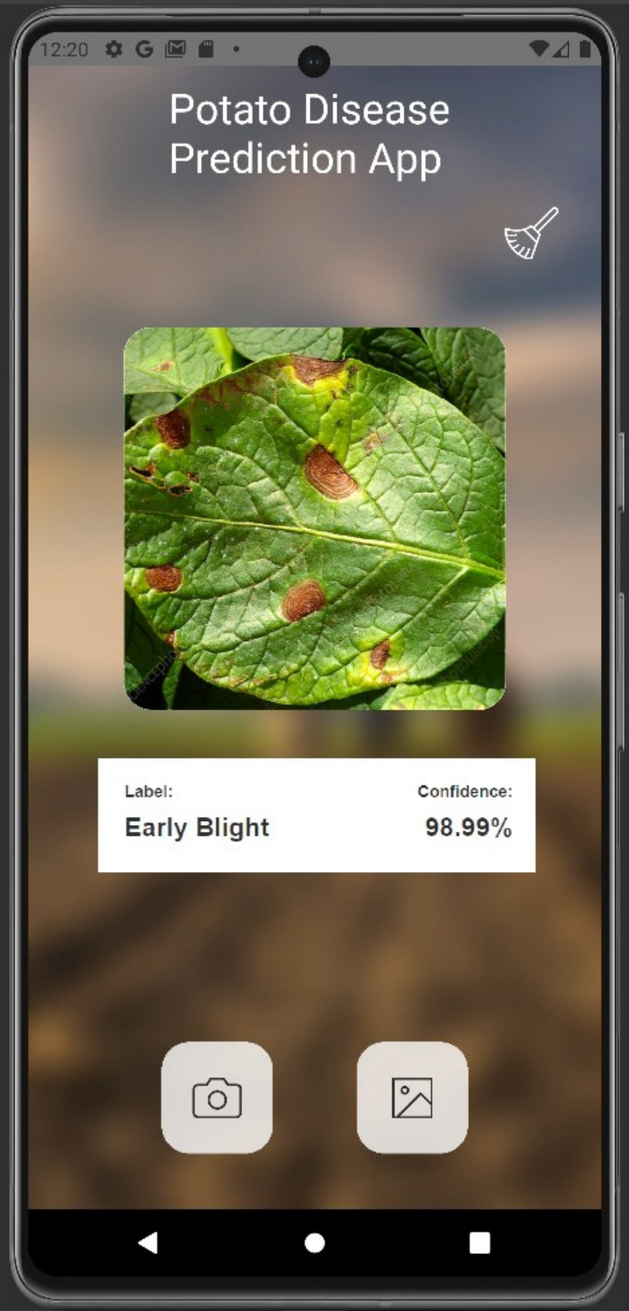
Created a virtual Android device using Android Studio's AVD Manager (for Windows users) and ran the emulator.

Mobile App Testing

Launched the React Native mobile app on the emulator, which will communicate with the deployed Google Cloud Function for image prediction, enabling real-time testing and interaction.

Running Devices - mlDemo

Running Devices: Pixel 7 Pro API 29



MOBILE APP DEPLOYMENT IN ANDROID EMULATOR

npm run android

```
info Starting JS server...
info Installing the app...

> Task :app:installDebug
Installing APK 'app-debug.apk' on 'Pixel_7_Pro_API_29(AVD) - 10' for :app:de
bug
Installed on 1 device.

BUILD SUCCESSFUL in 1m 14s
89 actionable tasks: 2 executed, 87 up-to-date
info Connecting to the development server...
8081
info Starting the app on "emulator-5554"...
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.catego
ry.LAUNCHER] cmp=com.mldemo/.MainActivity }
```

node "C:\Users\SIVA SRINIVA

```
> Task :react-native-config:copyDebugJniLibsProjectOnly UP-TO-DATE
> Task :react-native-image-picker:mergeDebugJniLibFolders UP-TO-DATE
> Task :react-native-image-picker:mergeDebugNativeLibs NO-SOURCE
> Task :react-native-image-picker:copyDebugJniLibsProjectOnly UP-TO-DATE
> Task :react-native-permissions:mergeDebugJniLibFolders UP-TO-DATE
> Task :react-native-permissions:mergeDebugNativeLibs NO-SOURCE
> Task :react-native-permissions:copyDebugJniLibsProjectOnly UP-TO-DATE
> Task :app:mergeDebugNativeLibs UP-TO-DATE
> Task :app:stripDebugDebugSymbols UP-TO-DATE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:writeDebugAppMetadata UP-TO-DATE
> Task :app:writeDebugSigningConfigVersions UP-TO-DATE
> Task :app:packageDebug UP-TO-DATE
> Task :app:createDebugApkListingFileRedirect UP-TO-DATE

> Task :app:installDebug
Installing APK 'app-debug.apk' on 'Pixel_7_Pro_API_29(AVD) - 10' for :app:de
bug
```

Data Collection - Preprocessing - Cleansing

Model Building using CNN

Data Prep: Gather, clean, label, and preprocess a diverse leaf image dataset.

Model Design: Craft a CNN architecture with convolutional and pooling layers.

Hyperparameter Tuning: Fine-tune settings like learning rate and batch size.

Transfer Learning: Leverage pre-trained CNN models and adapt them.

Loss & Metrics: Define appropriate loss functions and evaluation metrics.

Regularization: Combat overfitting with dropout and L2 regularization.

Optimizers: Choose optimizers and learning rate strategies.

Deployment: Scale, optimize, and integrate the model into applications.

Monitoring: Continuously assess model performance in production.

Interpretability: Employ techniques to explain model decisions, enhancing the trust

FAST API

GCP

POST MAN

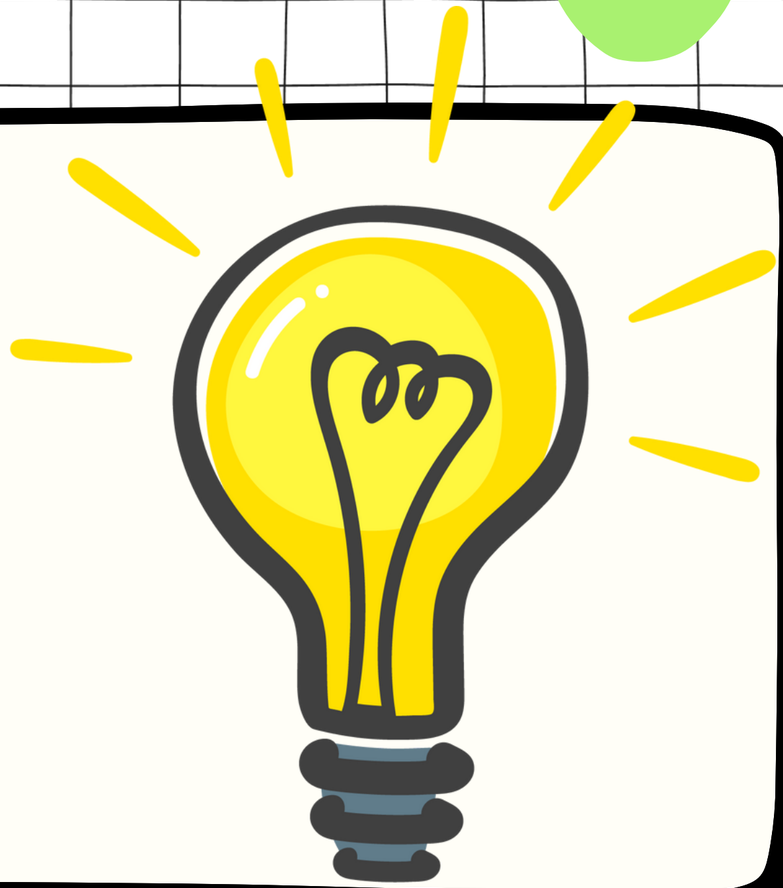
REACT NATIVE

I am so grateful for the opportunity and your faith on me
Willing to furnish my finest on this endeavor

Resources Involved

Android Studio | Postman | React | React Native | Jupyter Notebook | PyCharm Microsoft Visual Studio Code | Chatgpt Youtube Tutorials

Watched | Learned | Deployed on my own



"In the process of working on this project, I've had the opportunity to gather valuable insights from Chat GPT and various web and YouTube tutorials.
These resources have played a crucial role in shaping my approach.
It's worth mentioning that I come from a non-CS background, so I've actively sought out tutorials and received support from Chat GPT to navigate areas like front-end, back-end, API, and GCP. It's been a great learning journey.