Automated Agriculture System: Accurate Water Management and Disease Analysis

24-25J-164

Our Team

Sharan U.	IT21381690
Nakulabasgaran Y.	IT21228230
Sankeethan Y.	IT21228162
Mayootharan P.	IT21290206

Content

Background

Develop an automated agriculture system integrating sensors, data analytics, and machine learning for optimized irrigation and disease management.

What is Automated Agriculture System: Accurate Water Management and Disease Analysis

Why are we focusing on this topic?

What are the main problems we identified?

Research Problem

01

Irregular rainfall patterns, including droughts and floods, can disrupt planting schedules, affect germination, and reduce crop yield.

02

Extreme temperatures, both hot and cold, can stress crops, disrupt flowering and fruit set, and lead to yield losses.

03

Limited water availability can lead to reduced irrigation, affecting crop growth and yield.

04

Farmers often lack awareness of crop diseases and their symptoms, hindering effective outbreak management.

Objectives

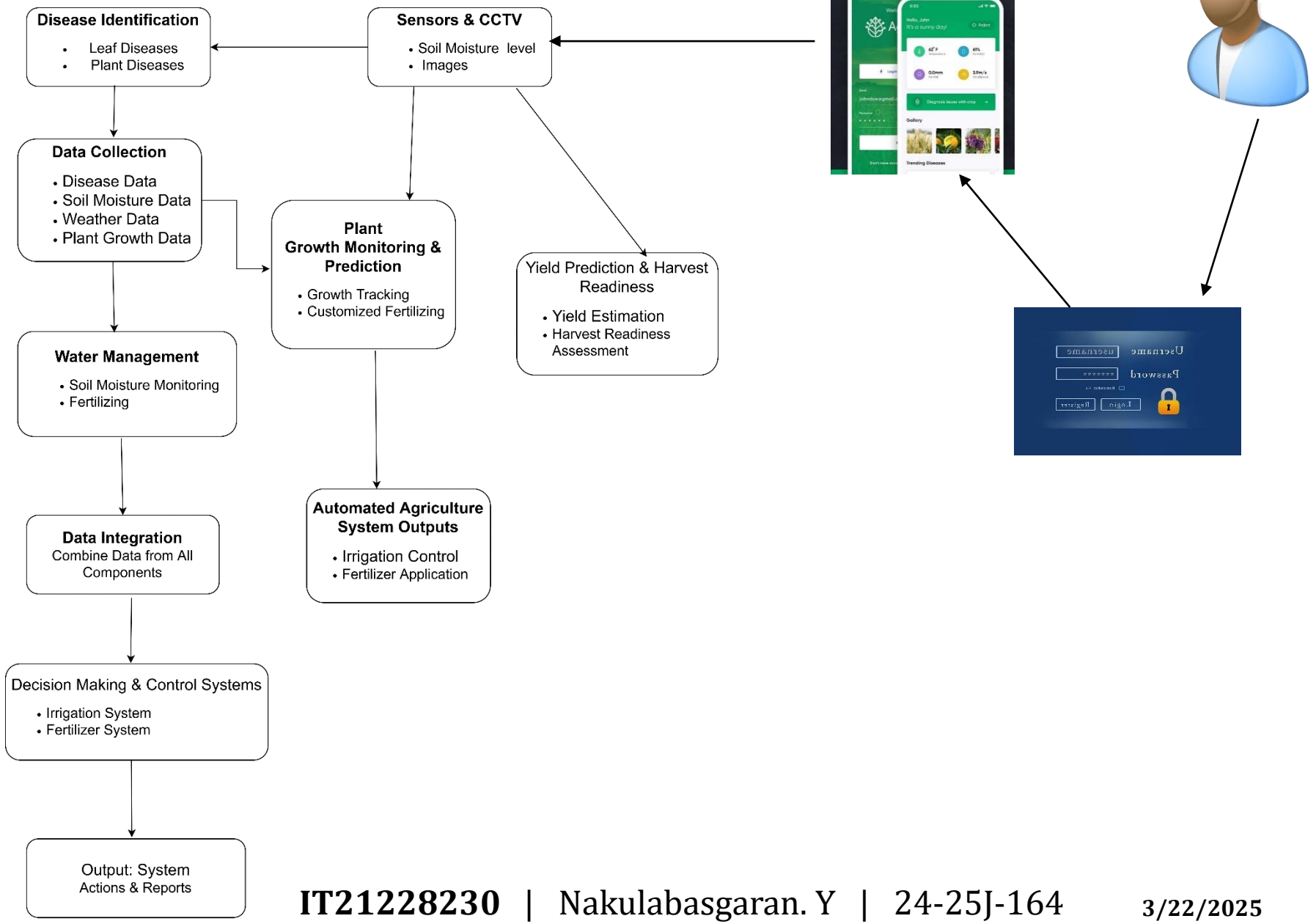
DEVELOP A DISEASE IDENTIFICATION SYSTEM

IMPLEMENT ADVANCED WATER MANAGEMENT TECHNOLOGY

INTEGRATED WEATHER FORECASTING AND PLANT GROWTH MONITORING SYSTEM BASED ON SOIL MOISTURE ANALYSIS

DEVELOP A MODEL FOR EFFICIENT YIELD PREDICTION, HARVEST READINESS, AND FERTILIZER OPTIMIZATION

Overall System Diagram



IT21228230 - Nakulabasgaran. Y

Information Technology

Water Management Technology

Introduction

01 Background

02 Research Problem

03 Objectives

Background

This component focuses on integrating advanced technologies to optimize water usage in agricultural systems, ensuring sustainable practices and reducing water wastage while maintaining crop health.

Research Problem

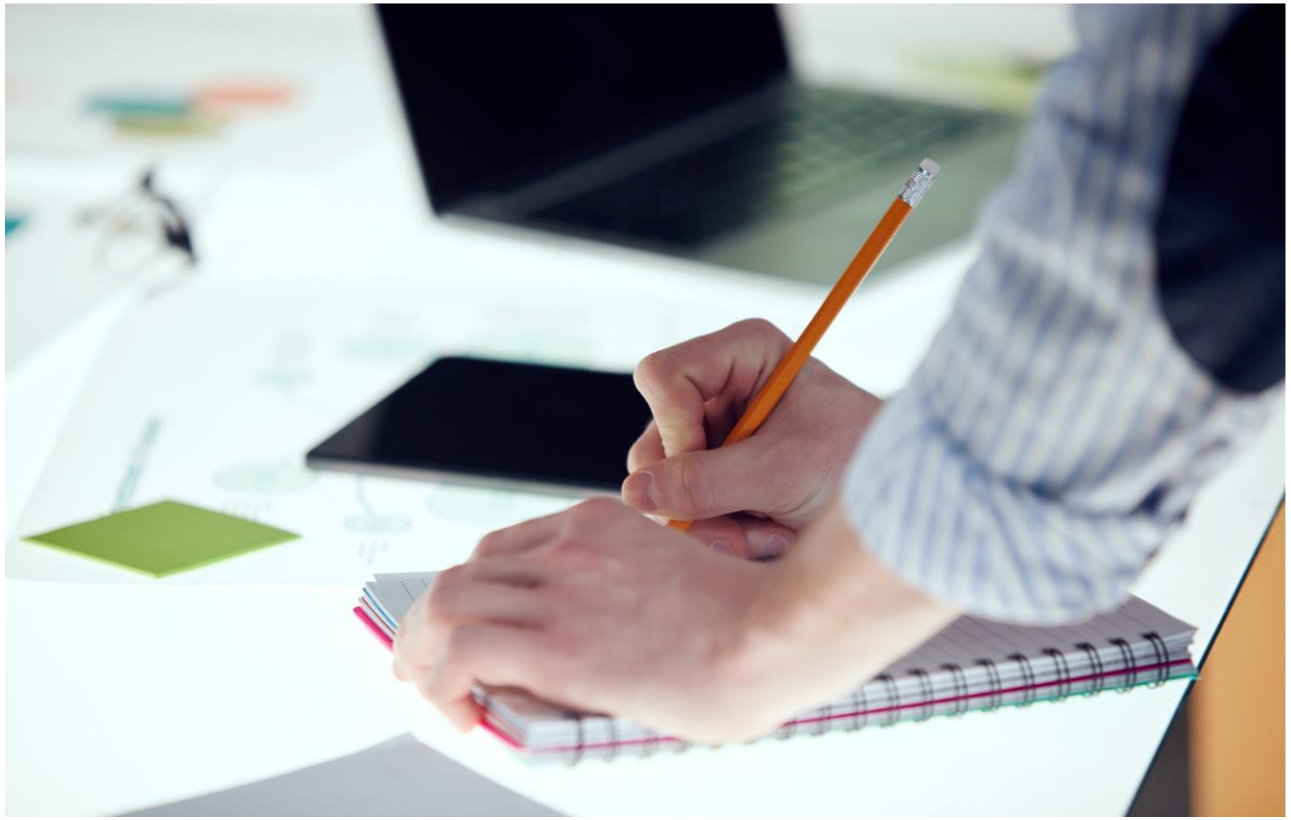
Inconsistent watering practices can lead to over- or under-watering, affecting crop health and yield

OBJECTIVES

Water Conservation: Minimize water waste by implementing precision irrigation techniques.

Automation: Automate irrigation systems to deliver water precisely when and where needed.

Core Objectives

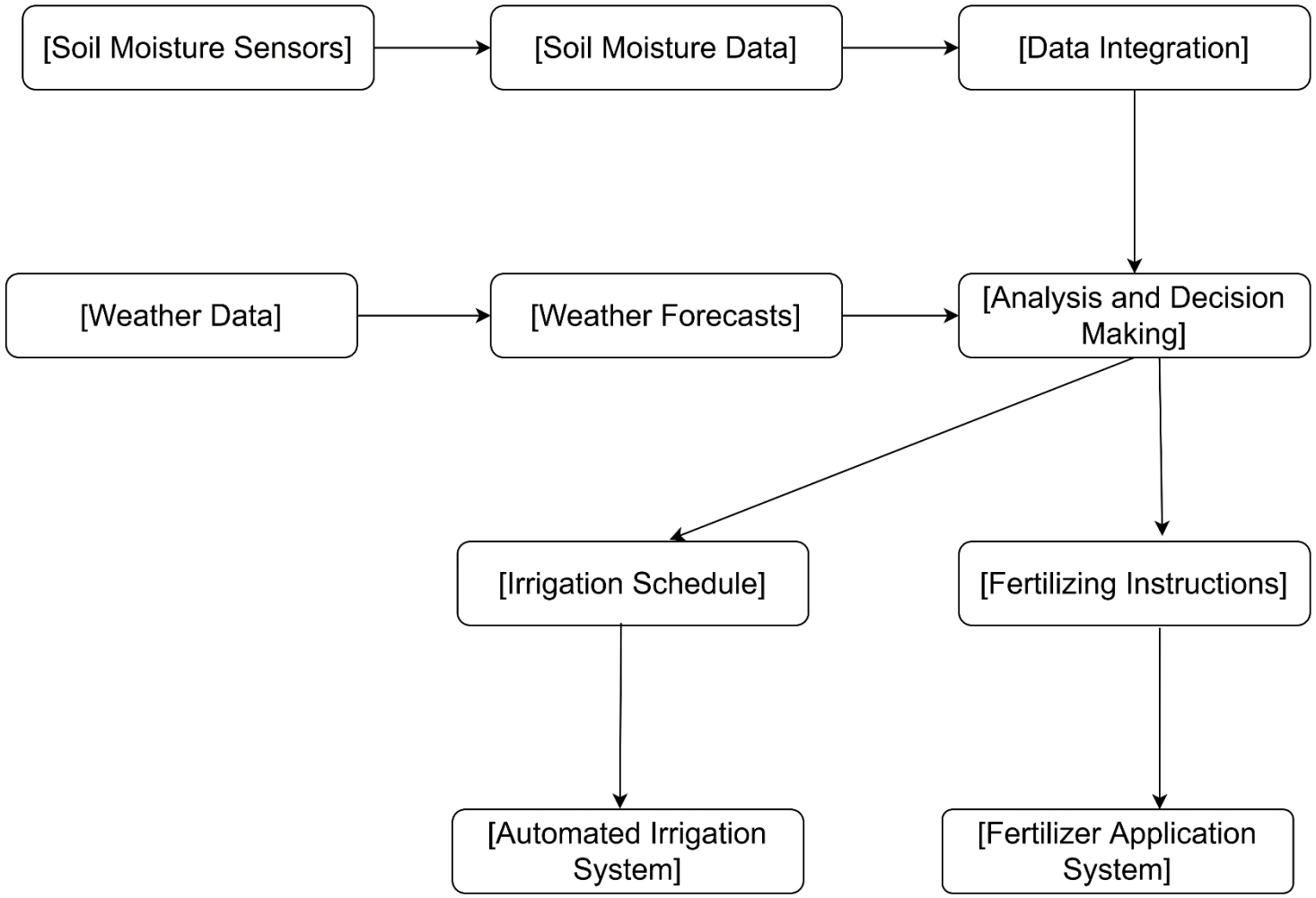


01 Soil Moisture-Based Watering

02 Weather Forecast Integration

03 Automated Watering System

System Diagram



Technologies

PROGRAMMING LANGUAGES

- Python

FRAMEWORK

- Flask

OTHER APPLICATION

- Arduino IDE
- PyCharm

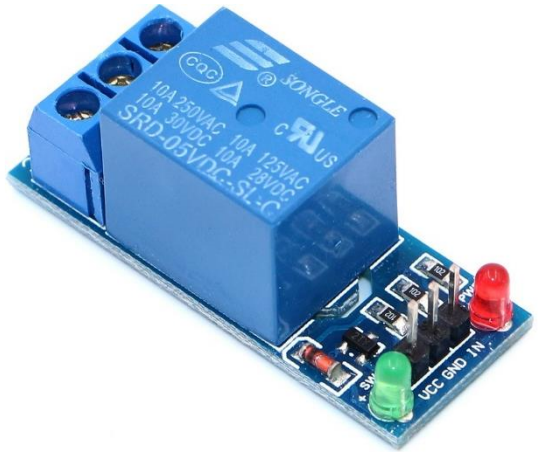
DATABASE

- Firebase

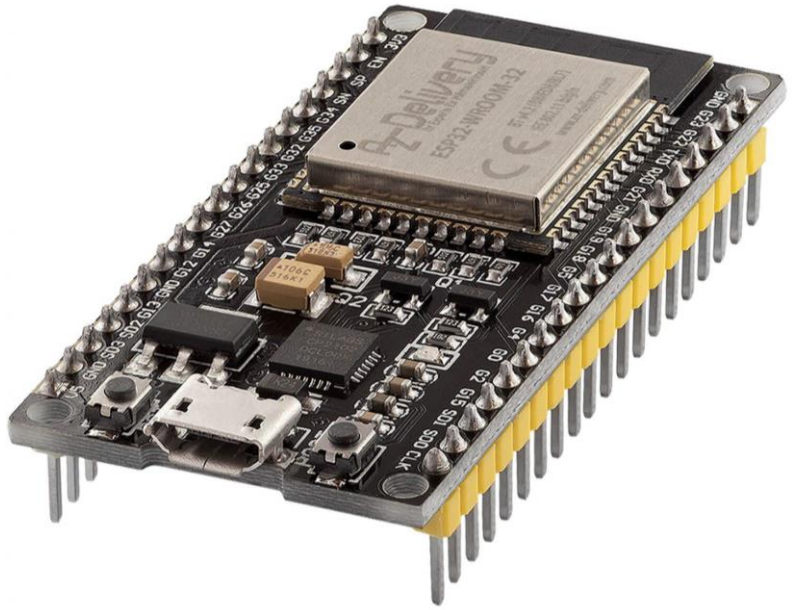
Tools

IOT-Device Circuits Items

Submersible Mini pump :
It give water to soil when the sensor sends the signal that the moisture level in the soil is low.



Channel-5V-Relay-Module :
turn on and off the pump automatically as the required water level is reached



ESP32 Microcontroller:
For data processing and communication.

Model

Predic_Watering.py | Train_watering_schedule.py

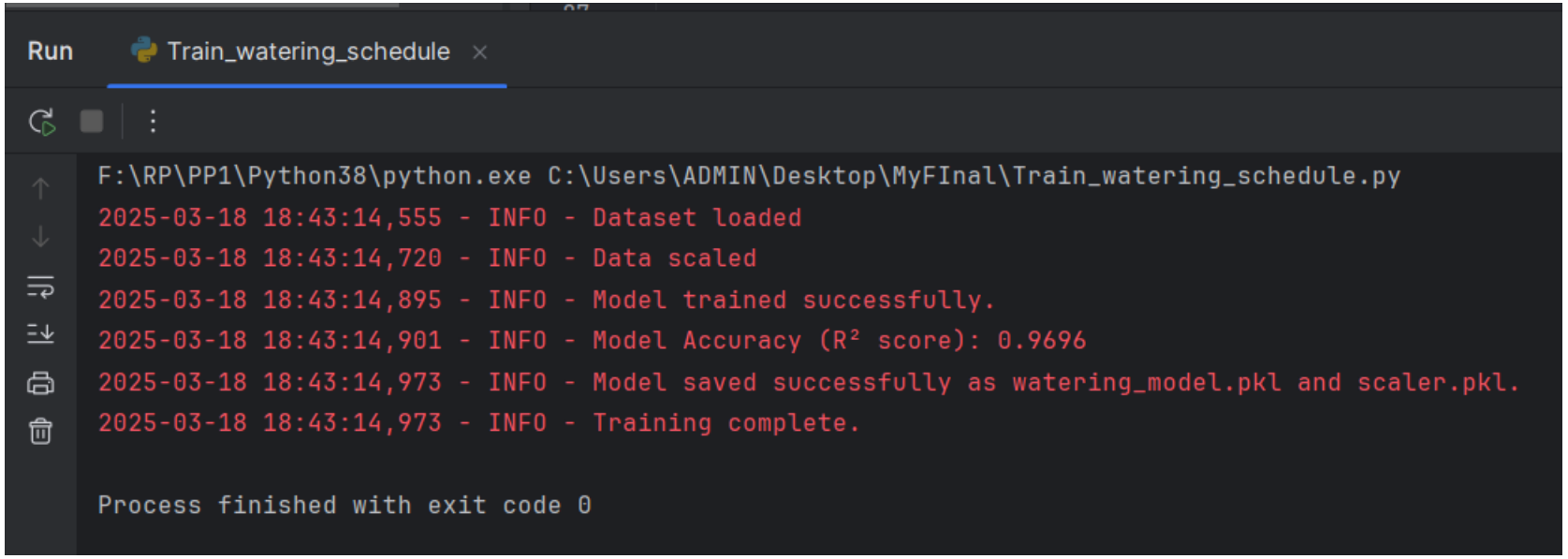
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import joblib
import logging
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')


def load_and_preprocess_data(file_path):  1 usage
    try:
        dataset = pd.read_csv(file_path)
        logging.info("Dataset loaded")
    except Exception as e:
        logging.error(f"Error loading dataset: {e}")
        raise

    # Check for missing values
    if dataset.isnull().sum().any():
        logging.warning("Missing values found in the dataset. Filling missing values with the mean.")
        dataset.fillna(dataset.mean(), inplace=True)

    # Split dataset
    X = dataset[['Precipitation (mm)', 'Ambient Temp (°C)', 'Humidity (%)', 'Soil Temp (°C)', 'Soil Moisture (%)']]
    y = dataset['Watering Duration (minutes)']
    return X, y
```

Predic_Watering.py | Train_watering_schedule.py

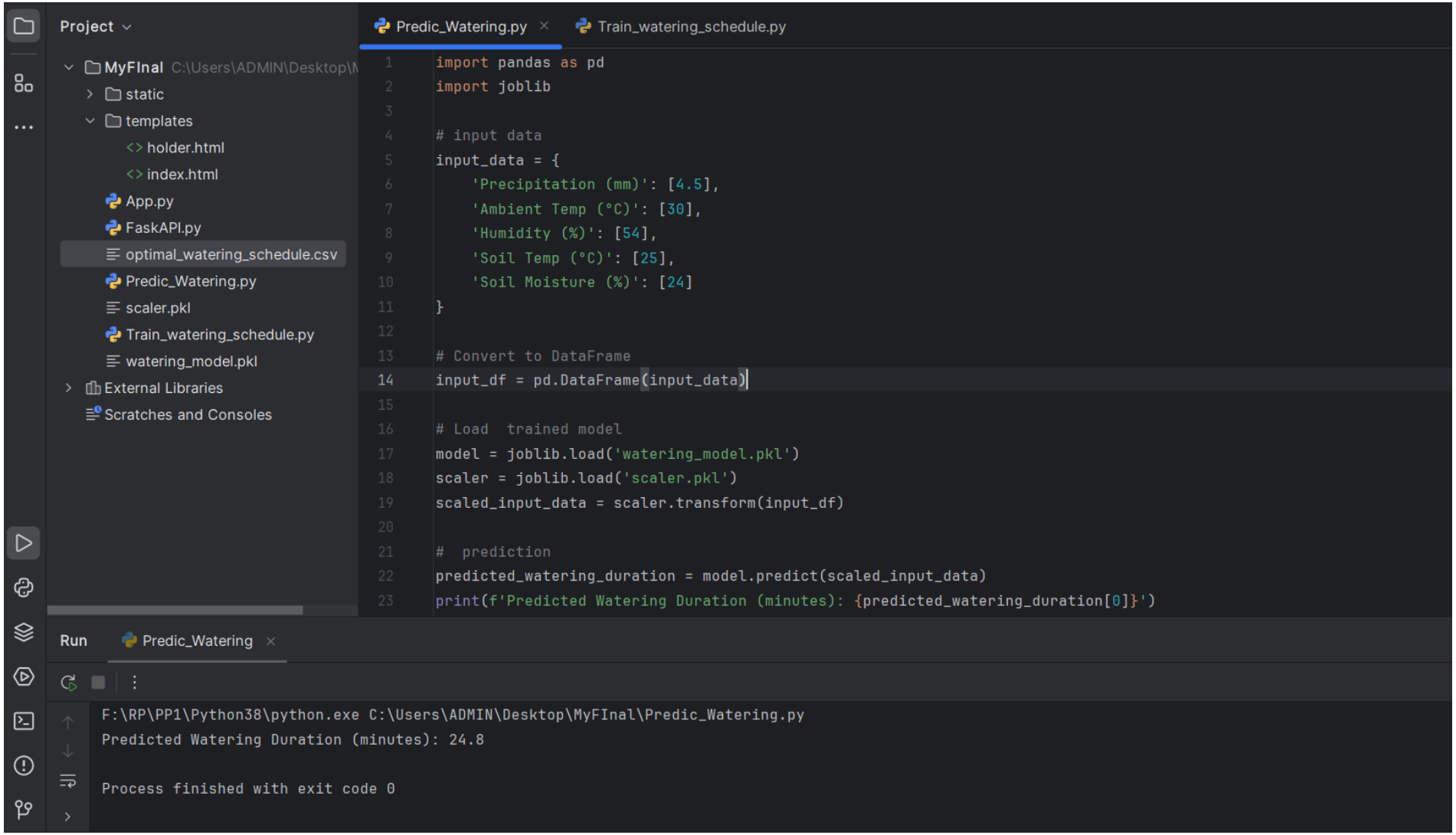
```python
def scale_data(X):  1 usage
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    logging.info("Data scaled")
    return X_scaled, scaler


def train_model(X_train, y_train):  1 usage
    try:
        model = RandomForestRegressor(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)
        logging.info("Model trained successfully.")
    except Exception as e:
        logging.error(f"Error training model: {e}")
        raise
    return model


def evaluate_model(model, X_test, y_test):  1 usage
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    logging.info(f"Model Accuracy (R² score): {r2:.4f}")
    return r2


def save_model(model, scaler, model_file='watering_model.pkl', scaler_file='scaler.pkl'):  1 usage
    try:
        joblib.dump(model, model_file)
        joblib.dump(scaler, scaler_file)
        logging.info(f"Model saved successfully as {model_file} and {scaler_file}.")
    except Exception as e:
        logging.error(f"Error saving model: {e}")
```

```
        raise


def main():
    dataset_file = 'optimal_watering_schedule.csv'
    X, y = load_and_preprocess_data(dataset_file)

    # Split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)

    X_train_scaled, scaler = scale_data(X_train)
    X_test_scaled = scaler.transform(X_test)

    model = train_model(X_train_scaled, y_train)

    evaluate_model(model, X_test_scaled, y_test)

    save_model(model, scaler)

    logging.info("Training complete.")


if __name__ == "__main__":
    main()

```

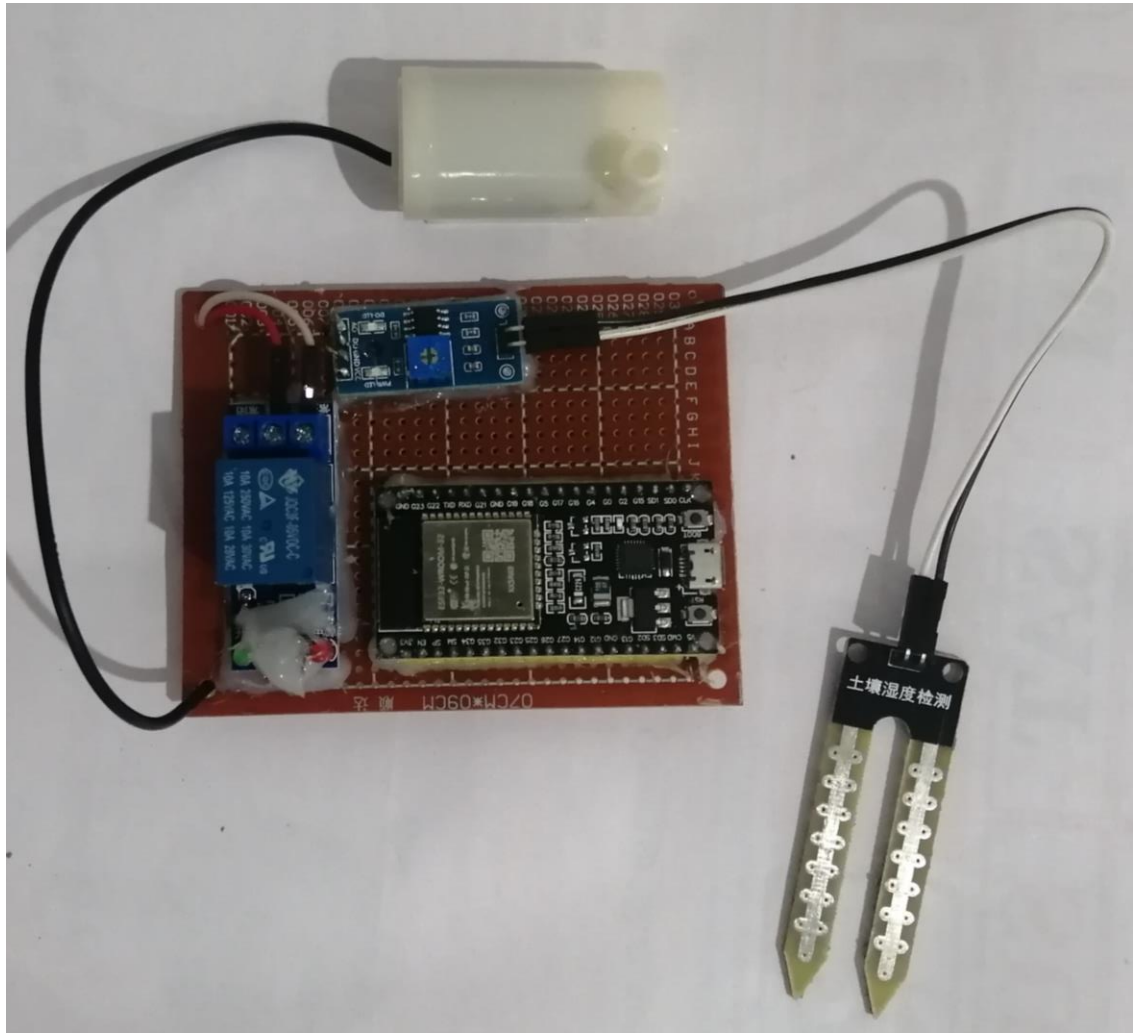
Accuracy


Run
Train_watering_schedule
F:\RP\PP1\Python38\python.exe C:\Users\ADMIN\Desktop\MyFInal\Train_watering_schedule.py
2025-03-18 18:43:14,555 - INFO - Dataset loaded
2025-03-18 18:43:14,720 - INFO - Data scaled
2025-03-18 18:43:14,895 - INFO - Model trained successfully.
2025-03-18 18:43:14,901 - INFO - Model Accuracy (R² score): 0.9696
2025-03-18 18:43:14,973 - INFO - Model saved successfully as watering_model.pkl and scaler.pkl.
2025-03-18 18:43:14,973 - INFO - Training complete.
Process finished with exit code 0

BackEnd


Project
MyFInal C:\Users\ADMIN\Desktop\M
static
templates
holder.html
index.html
App.py
FaskAPI.py
optimal_watering_schedule.csv
Predic_Watering.py
scaler.pkl
Train_watering_schedule.py
watering_model.pkl
External Libraries
Scratches and Consoles
Predic_Watering.py
Train_watering_schedule.py
import pandas as pd
import joblib
# input data
input_data = {
'Precipitation (mm)': [4.5],
'Ambient Temp (°C)': [30],
'Humidity (%)': [54],
'Soil Temp (°C)': [25],
'Soil Moisture (%)': [24]
}
# Convert to DataFrame
input_df = pd.DataFrame(input_data)
# Load trained model
model = joblib.load('watering_model.pkl')
scaler = joblib.load('scaler.pkl')
scaled_input_data = scaler.transform(input_df)
# prediction
predicted_watering_duration = model.predict(scaled_input_data)
print(f'Predicted Watering Duration (minutes): {predicted_watering_duration[0]}')
Run
Predic_Watering
F:\RP\PP1\Python38\python.exe C:\Users\ADMIN\Desktop\MyFInal\Predic_Watering.py
Predicted Watering Duration (minutes): 24.8
Process finished with exit code 0

Prototype



Requirements

Functional

- Analyze collected data from Sensor.
- Data Processing
- Irrigation systems

Non Functional

- Accuracy
- Reliability
- Cost-Effectiveness
- User-Friendliness
- Security

Completed

- Built prototype 80%
- Integrate Tools
- Model Training

To be Complete

- Connect the prototype with Database
- Web UI Design (Frontend)
- Testing

References

[1] https://www.researchgate.net/publication/372523323_Automated_Pest_and_Disease_Identification_in_A griculture_using_Image_Processing

[2] https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1356260/full

[3] https://www.researchgate.net/publication/374505378_An_Automated_System_to_Detect_Plant_Disease_ using_Deep_Learning

[4]

https://ieeexplore.ieee.org/document/9754145

[5]

https://www.sciencedirect.com/science/article/abs/pii/S2214785321042115

[6] Hillel, D. (2004). Introduction to Environmental Soil Physics. Academic Press.

[7] Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop Evapotranspiration: Guidelines for Computing Crop Water Requirements. FAO Irrigation and Drainage Paper 56.

IT21381690

Sharan.u

Information Technology

Disease Identification

Content

BACKGROUND

What are the diseases and how we are going to identify the diseases

What is object Tracking

What is Deep CNN

IT21381690 | Sharan .U | 24-25J-164

Research Problem

How to track or identify what are the disease in the tree

How can we notify about the disease for the farmers

Objectives

01 Early Detection

Detect diseases at an early stage to prevent their spread and minimize damage.

02 Improve Farmer Knowledge and Practices:

Educate and inform farmers about disease prevention, identification, and management techniques, promoting sustainable farming practices.

03 Reduce Economic Losses:

Minimize economic losses for farmers by preventing widespread crop failure and reducing the costs associated with disease management.

Tools and technology

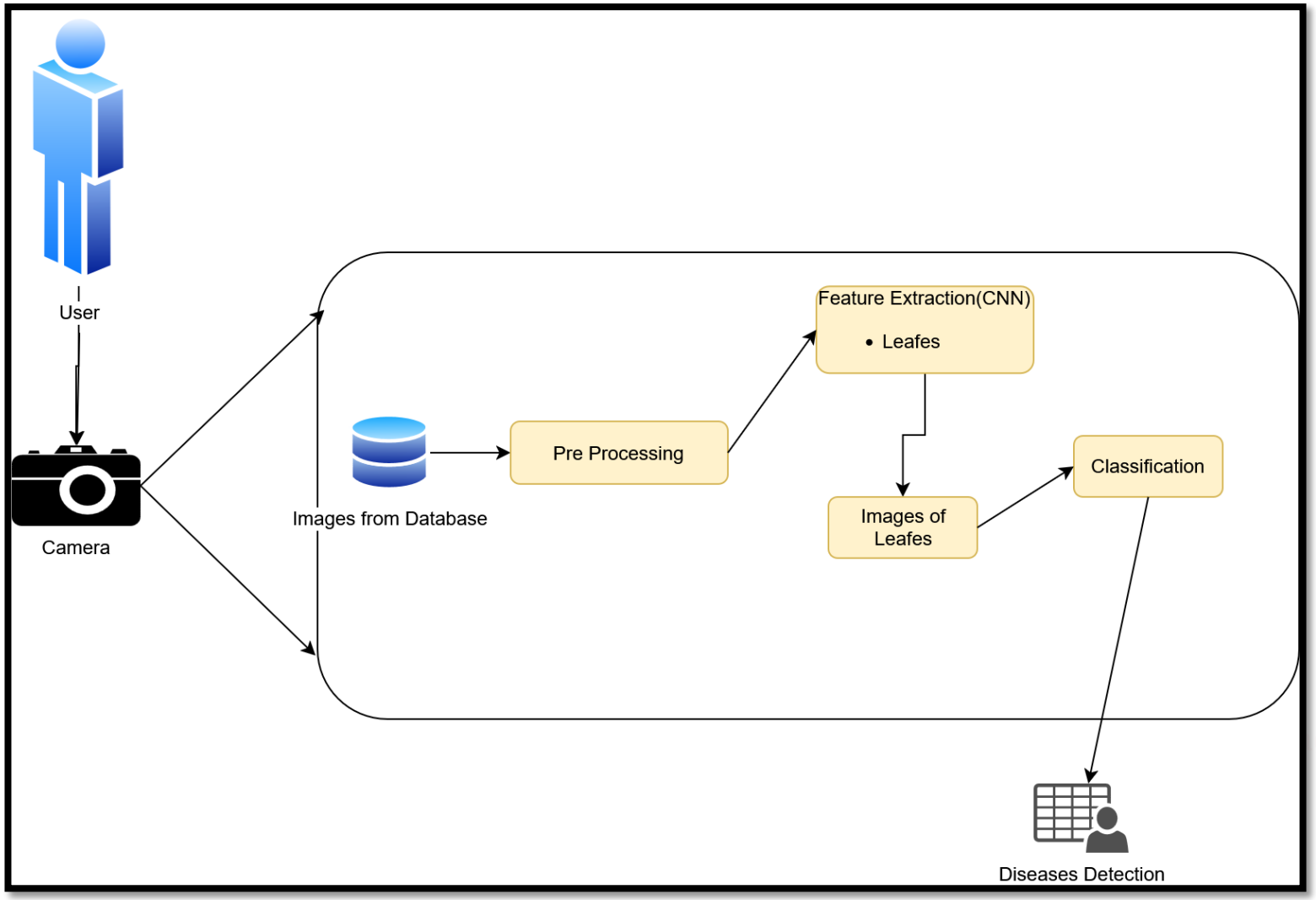
Frame Works
CNN ,
Keras/Tensorflow

Tools
GIT

Project Management
Ms Planner

Programming language
React Js , Python

System Diagram



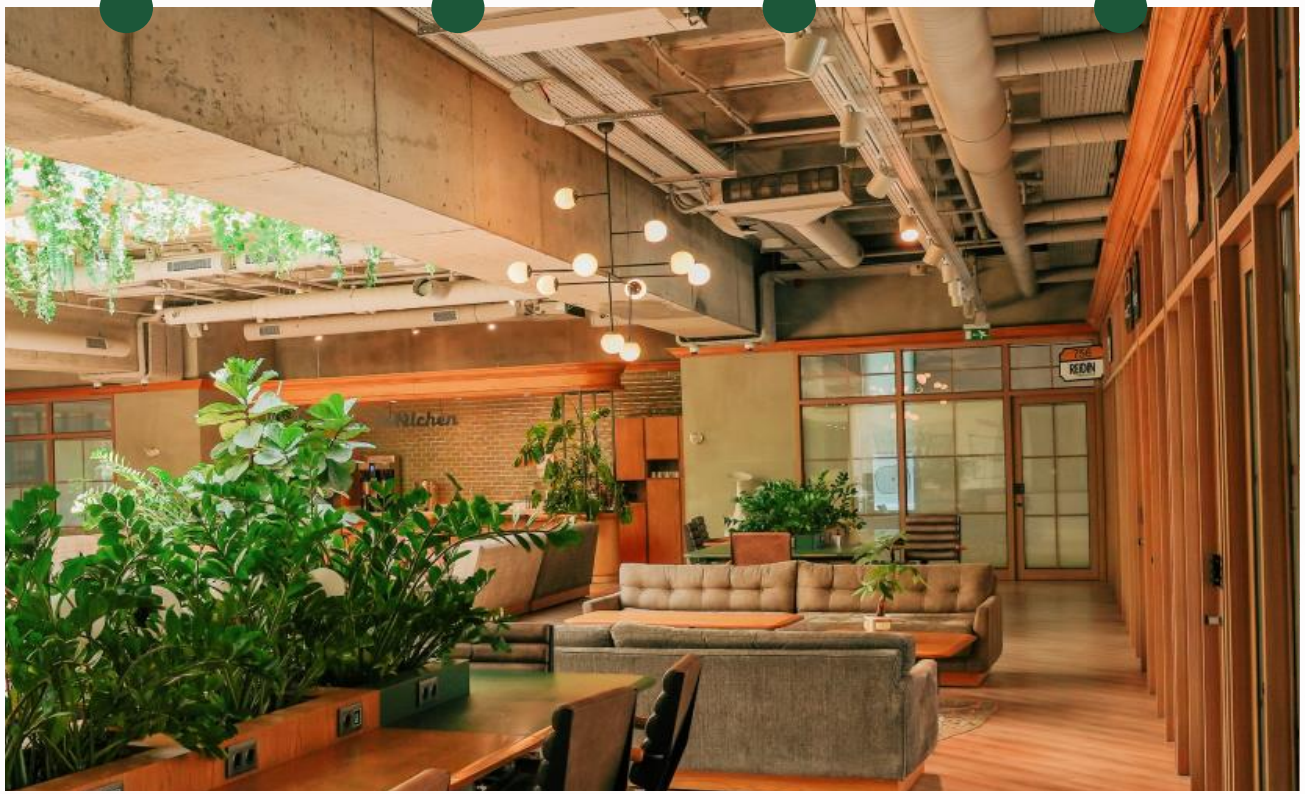
Requirements

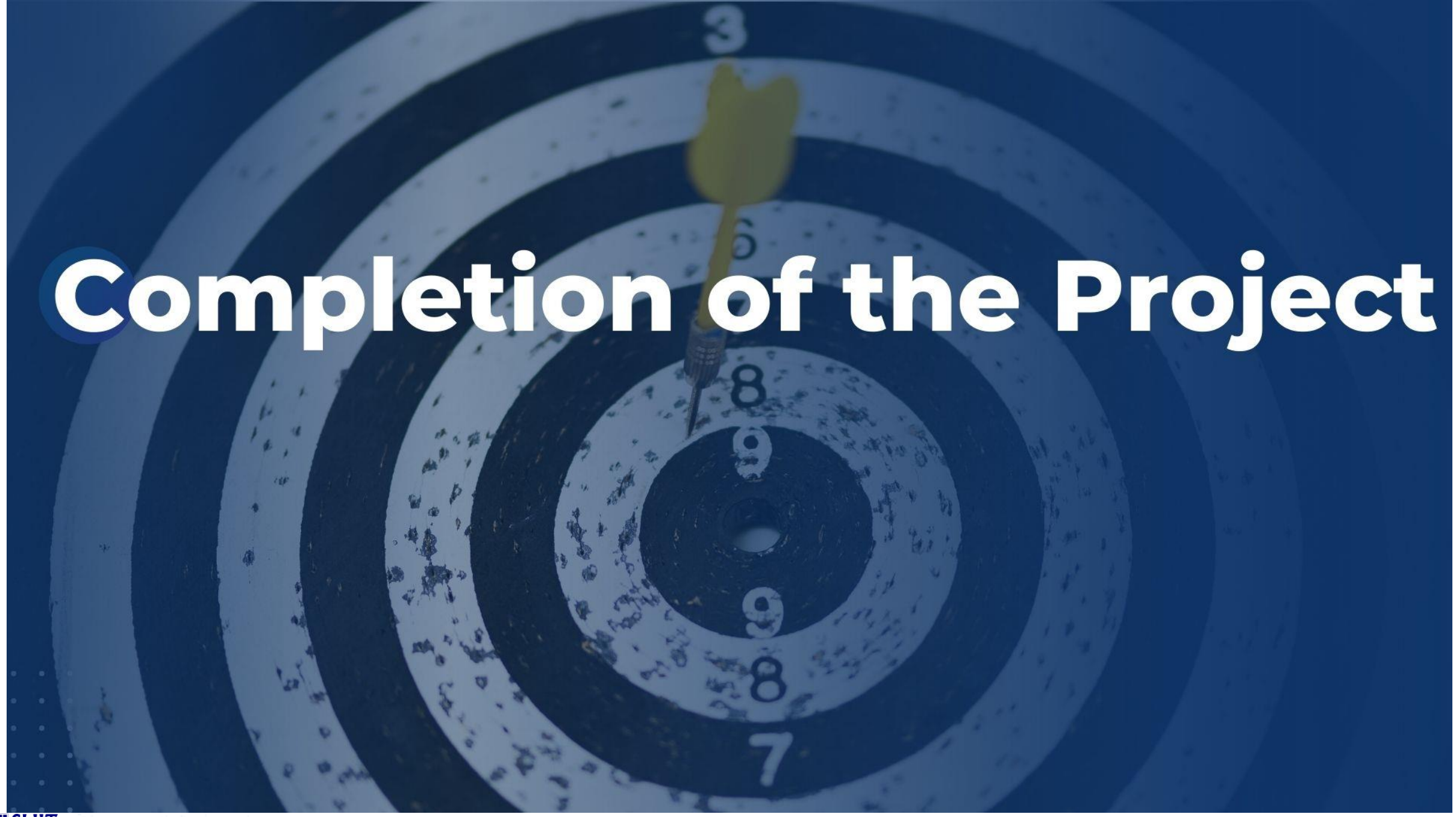
Functional

- Image Capture and Input
- Image Preprocessing
- Disease Detection and Classification
- Notification and Reporting

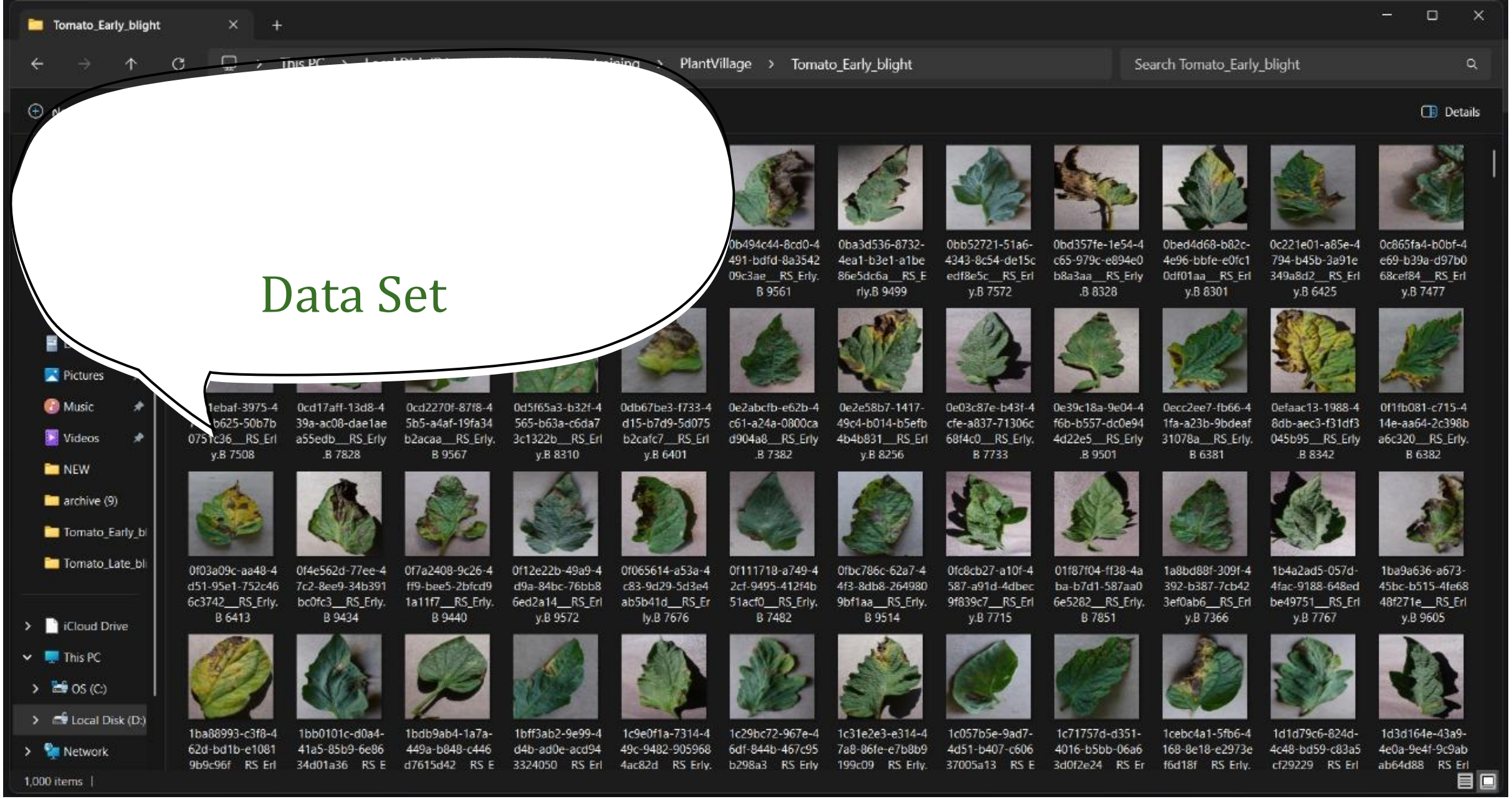
Non Funcational

- Accuracy
- performance
- Availability
- usability




Completion of the Project

Data Set


Tomato_Early_blight
PlantVillage
Tomato_Early_blight
Search Tomato_Early_blight
Details
Pictures
Music
Videos
NEW
archive (9)
Tomato_Early_bl
Tomato_Late_bl
iCloud Drive
This PC
OS (C:)
Local Disk (D:)
Network
1,000 items

Model Training

```
112/112 [==============================] - 232s 2s/step - loss: 0.7848 - accuracy: 0.6086 - val_loss: 0.6873 - val_accuracy: 0.7232
Epoch 2/50
112/112 [==============================] - 206s 2s/step - loss: 0.4766 - accuracy: 0.7984 - val_loss: 0.3200 - val_accuracy: 0.8549
Epoch 3/50
112/112 [==============================] - 194s 2s/step - loss: 0.3934 - accuracy: 0.8357 - val_loss: 0.5766 - val_accuracy: 0.7879
Epoch 4/50
112/112 [==============================] - 195s 2s/step - loss: 0.3598 - accuracy: 0.8578 - val_loss: 0.6928 - val_accuracy: 0.7165
Epoch 5/50
112/112 [==============================] - 191s 2s/step - loss: 0.3421 - accuracy: 0.8609 - val_loss: 0.2980 - val_accuracy: 0.8683
Epoch 6/50
112/112 [==============================] - 192s 2s/step - loss: 0.2760 - accuracy: 0.8835 - val_loss: 0.2895 - val_accuracy: 0.8973
Epoch 7/50
112/112 [==============================] - 193s 2s/step - loss: 0.3122 - accuracy: 0.8718 - val_loss: 0.2510 - val_accuracy: 0.9018
Epoch 8/50
112/112 [==============================] - 196s 2s/step - loss: 0.2323 - accuracy: 0.9099 - val_loss: 0.2089 - val_accuracy: 0.9241
Epoch 9/50
112/112 [==============================] - 195s 2s/step - loss: 0.2402 - accuracy: 0.9023 - val_loss: 0.2551 - val_accuracy: 0.8884
Epoch 10/50
112/112 [==============================] - 198s 2s/step - loss: 0.2254 - accuracy: 0.9115 - val_loss: 0.2434 - val_accuracy: 0.8951
Epoch 11/50
112/112 [==============================] - 195s 2s/step - loss: 0.1837 - accuracy: 0.9303 - val_loss: 0.3928 - val_accuracy: 0.8616
Epoch 12/50
112/112 [==============================] - 292s 3s/step - loss: 0.1876 - accuracy: 0.9219 - val_loss: 0.4521 - val_accuracy: 0.8103
Epoch 13/50
112/112 [==============================] - 351s 3s/step - loss: 0.2197 - accuracy: 0.9113 - val_loss: 0.3225 - val_accuracy: 0.8728
Epoch 14/50
112/112 [==============================] - 359s 3s/step - loss: 0.1689 - accuracy: 0.9351 - val_loss: 0.2297 - val_accuracy: 0.8973
Epoch 15/50
112/112 [==============================] - 338s 3s/step - loss: 0.1404 - accuracy: 0.9490 - val_loss: 0.2506 - val_accuracy: 0.9085
Epoch 16/50
112/112 [==============================] - 346s 3s/step - loss: 0.1326 - accuracy: 0.9507 - val_loss: 0.2217 - val_accuracy: 0.9062
Epoch 17/50
112/112 [==============================] - 352s 3s/step - loss: 0.1352 - accuracy: 0.9527 - val_loss: 0.1301 - val_accuracy: 0.9509
Epoch 18/50
112/112 [==============================] - 342s 3s/step - loss: 0.1323 - accuracy: 0.9541 - val_loss: 0.1370 - val_accuracy: 0.9509
Epoch 19/50
112/112 [==============================] - 350s 3s/step - loss: 0.1291 - accuracy: 0.9524 - val_loss: 0.1514 - val_accuracy: 0.9397
Epoch 20/50
112/112 [==============================] - 348s 3s/step - loss: 0.1681 - accuracy: 0.9378 - val_loss: 0.2712 - val_accuracy: 0.8929
Epoch 21/50
112/112 [==============================] - 345s 3s/step - loss: 0.1082 - accuracy: 0.9569 - val_loss: 0.2783 - val_accuracy: 0.9062
Epoch 22/50
112/112 [==============================] - 345s 3s/step - loss: 0.1052 - accuracy: 0.9600 - val_loss: 0.1062 - val_accuracy: 0.9710
Epoch 23/50
112/112 [==============================] - 349s 3s/step - loss: 0.1240 - accuracy: 0.9538 - val_loss: 0.2360 - val_accuracy: 0.9420
Epoch 24/50
112/112 [==============================] - 358s 3s/step - loss: 0.1032 - accuracy: 0.9616 - val_loss: 0.1834 - val_accuracy: 0.9330
Epoch 25/50
```

Model accuracy

Old

```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test accuracy: {test_accuracy:.4f}")
```

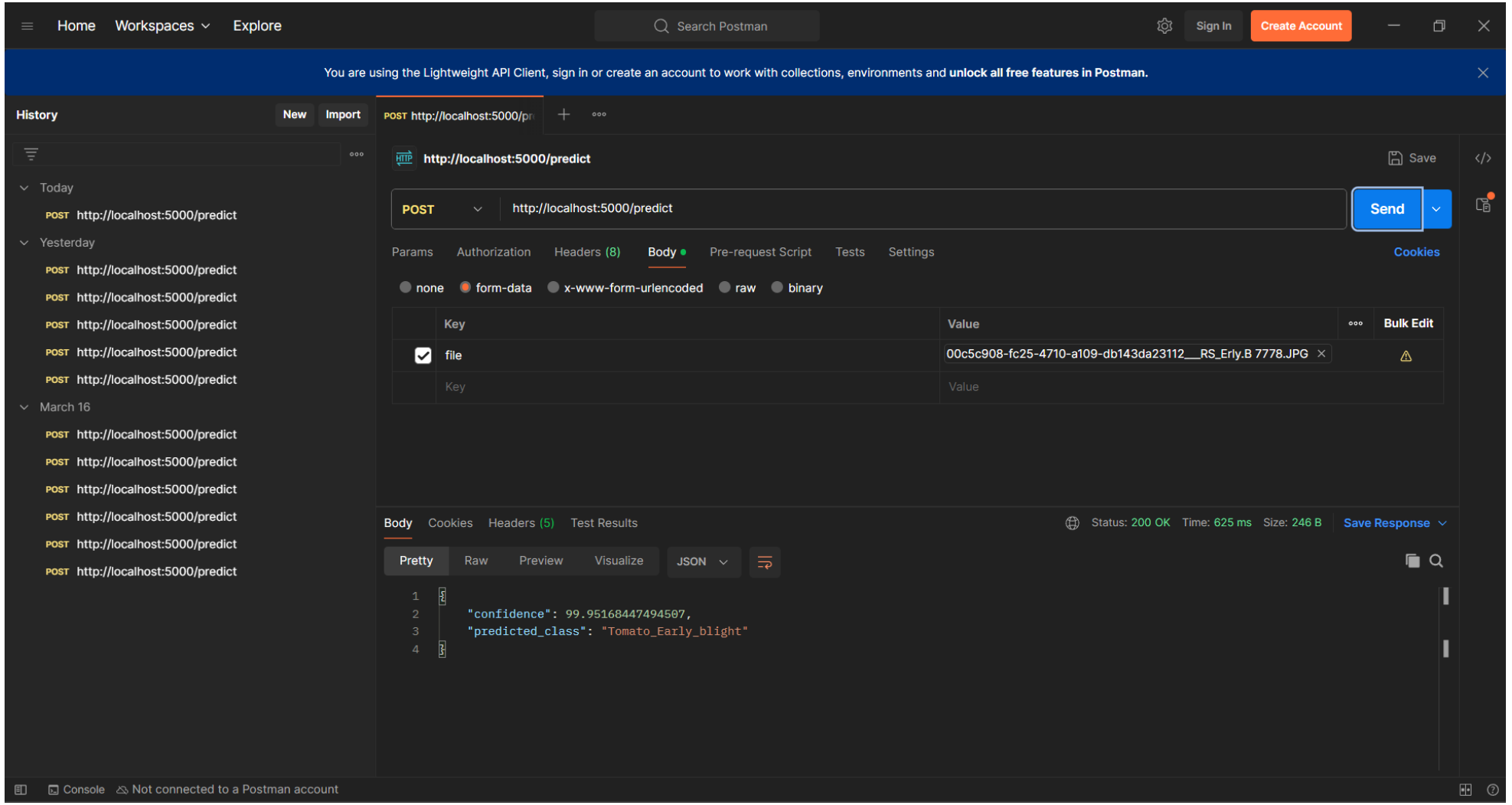
```
56/56 ━━━━━━━━━━━━━━━━━━━━ 120s 2s/step - accuracy: 0.8432 - loss: 0.4751
Test accuracy: 0.8481
```

New

```
[60]: scores
```

```
[60]: [0.1117370650172336, 0.96875]
```

Working component



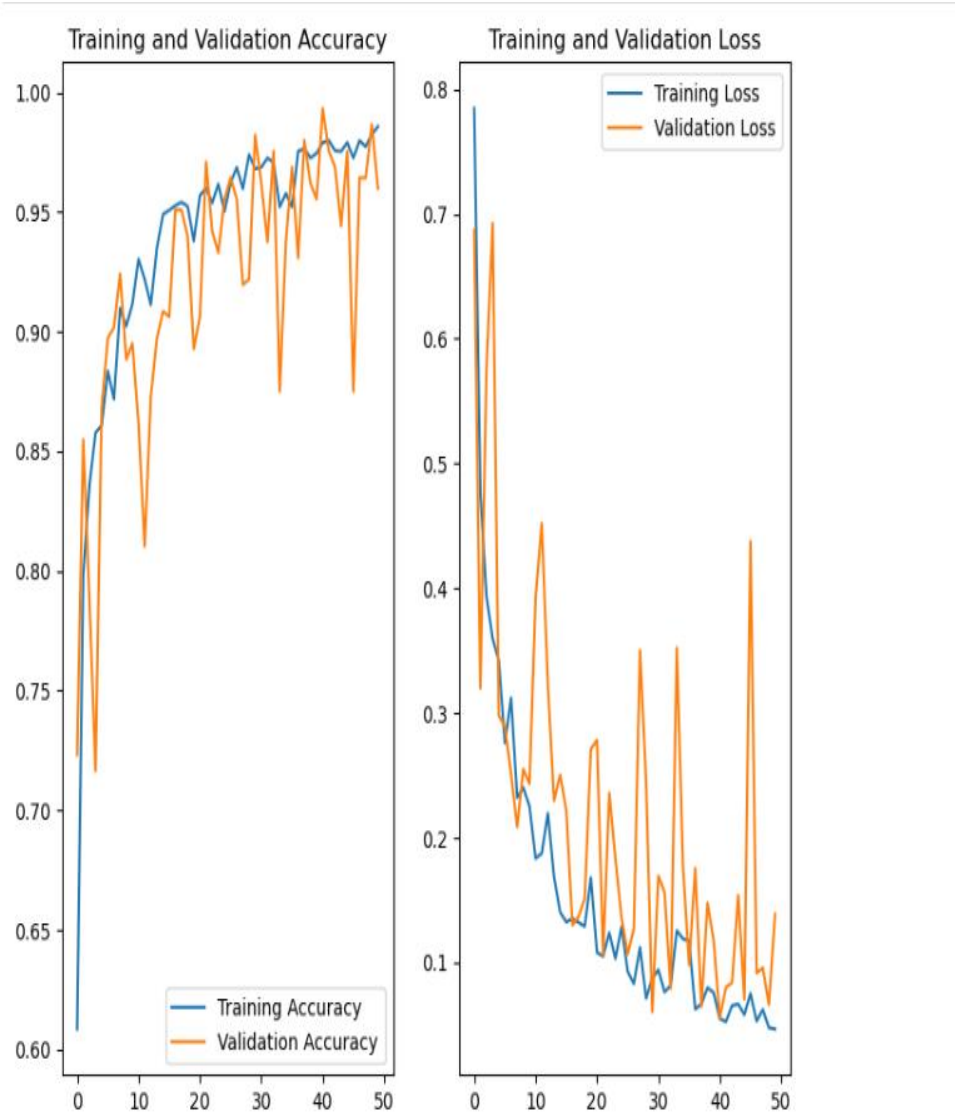
BackEnd

```python
import os
import tensorflow as tf

try:
    from tensorflow.keras import layers, models
except ImportError:
    print("Failed to import tensorflow.keras. Using tf.keras instead.")
    from tensorflow import keras
    layers = keras.layers
    models = keras.models
from flask import Flask, request, jsonify
from werkzeug.utils import secure_filename
import numpy as np


app = Flask(__name__)


UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

os.makedirs(UPLOAD_FOLDER, exist_ok=True)


MODEL_PATH = "D:/archive (9)/tomato.h5"
```

```python
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part in the request'}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No file selected'}), 400

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        try:
            img_array = preprocess_image(file_path)
            predictions = model.predict(img_array)
            predicted_class_idx = np.argmax(predictions[0])
            confidence = float(predictions[0][predicted_class_idx]) * 100

            class_names = ['Tomato_Early_blight', 'Tomato_Late_blight', 'Tomato_healthy']
            predicted_label = class_names[predicted_class_idx]

            os.remove(file_path)
            return jsonify({
                'predicted_class': predicted_label,
```

```python
            os.remove(file_path)
            return jsonify({
                'predicted_class': predicted_label,
                'confidence': confidence
            }), 200

        except Exception as e:
            if os.path.exists(file_path):
                os.remove(file_path)
            return jsonify({'error': f'Prediction failed: {str(e)}'}), 500

    return jsonify({'error': 'Invalid file format'}), 400


@app.route('/health', methods=['GET'])
def health():
    return jsonify({'status': 'healthy', 'model_loaded': model is not None}), 200

if __name__ == '__main__':
    os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
    print(f"Starting Flask app with TensorFlow {tf.__version__}")
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Accuracy Matix



IT21228162 - Sankeethan. Y

Information Technology

Integrated Weather Forecasting and Plant Growth Monitoring System Based on Soil Moisture Analysis

Introduction

01 Background

02 Research Problem

03 Objectives

Background

This component integrates weather forecasting and soil moisture analysis to monitor plant growth and predict future growth patterns. The goal is to optimize farming practices by aligning them with environmental conditions and crop needs.

Research Problem

Farmers need accurate predictions of plant growth to make informed decisions about fertilization and other interventions.

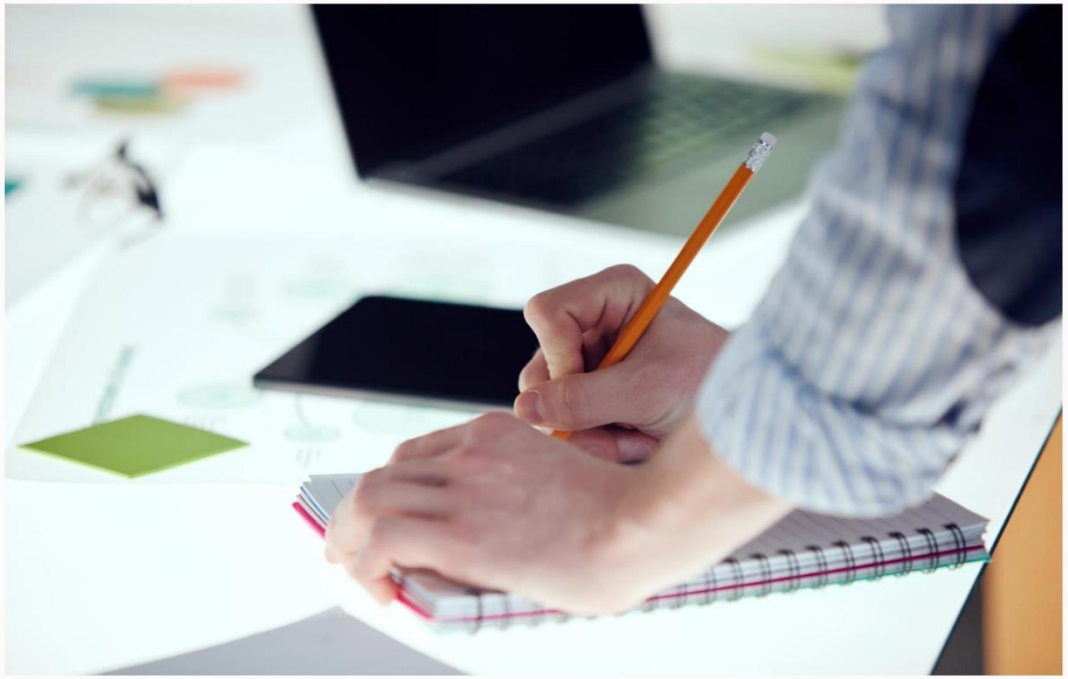
OBJECTIVES

Weather Forecast Integration

Develop a system for monitoring plant growth.

Predict growth outcomes based on customized fertilization

Methodology

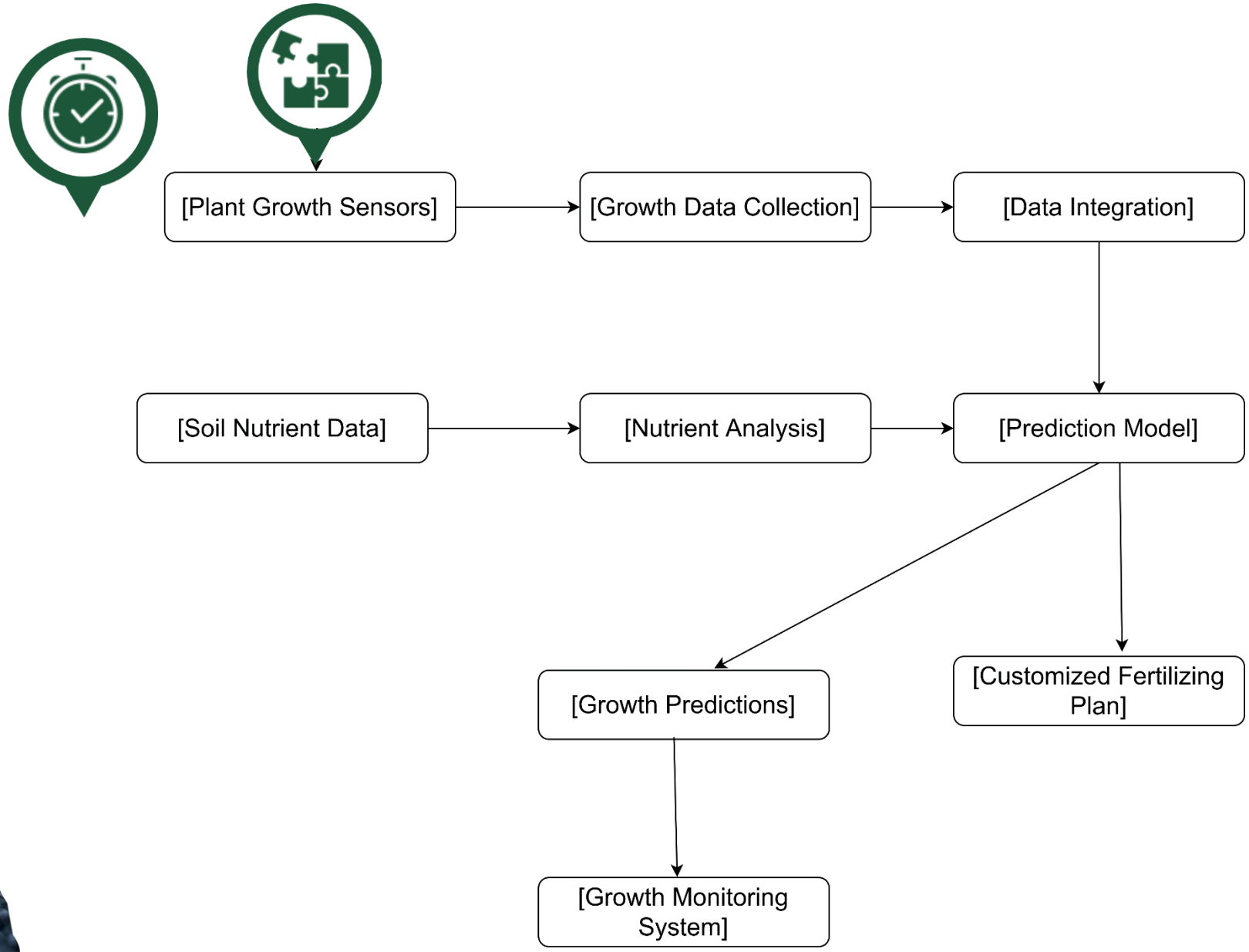


01 System Diagram

02 Technologies

03 Requirements

System Diagram



Technologies

PROGRAMMING LANGUAGES

- Python

FRAMEWORK

- Flask

OTHER APPLICATION

- Arduino IDE
- PyCharm

DATABASE

- Firebase

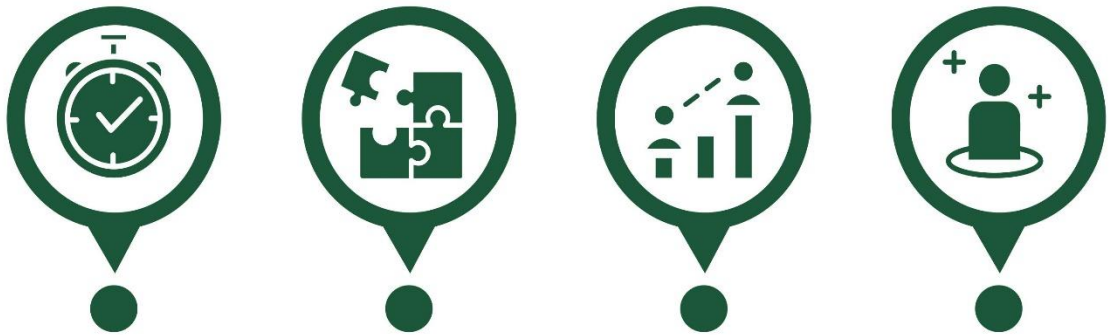
Requirements

Functional

- Growth monitoring software
- Machine learning models for prediction
- Analytical software for growth predictions

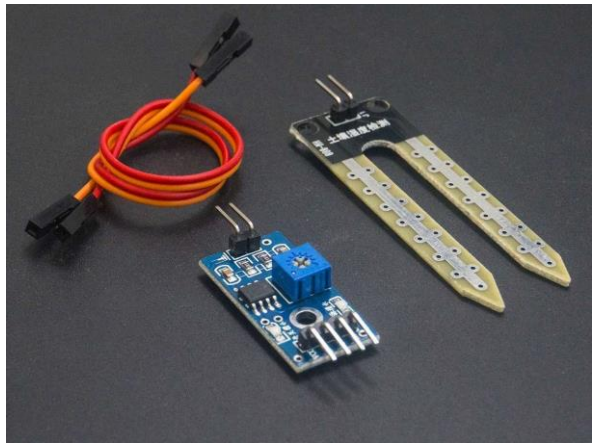
Non Funcational

- Accuracy
- Performance
- Availability
- Usability

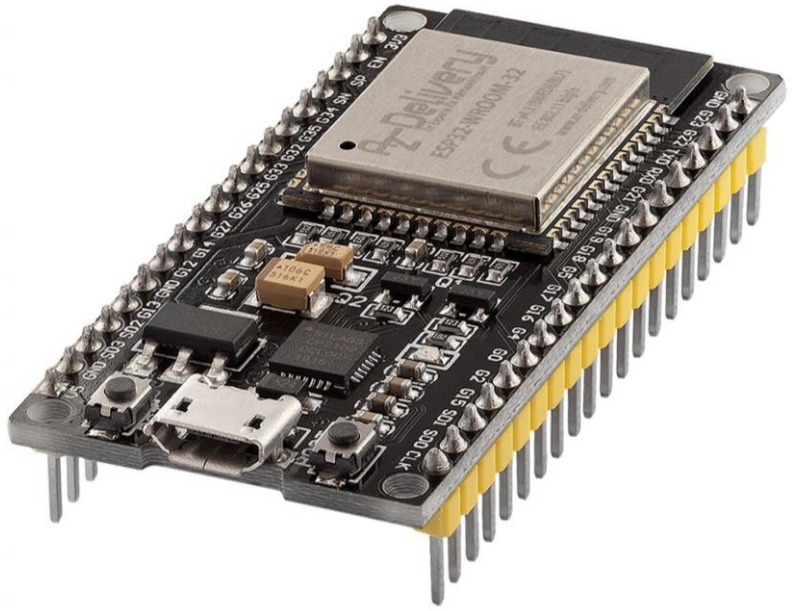


Tools

IOT-Device Circuits Items



Soil Moisture Sensor Module:
measure or estimate the amount of water in the soil.



ESP32 Microcontroller:
For data processing and communication.

Model

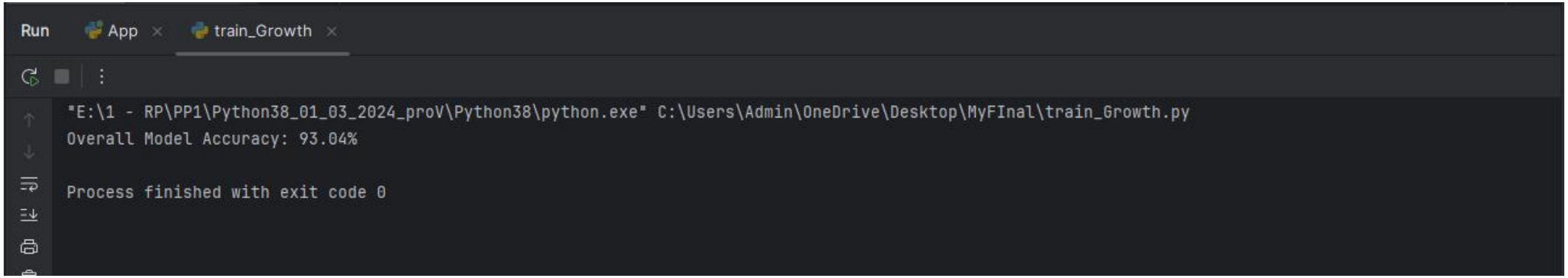
train_Growth.py

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error
import joblib

# Load dataset
df = pd.read_csv('plant_growth_data.csv')

# Prepare Features and Target Variables
X = df[['Soil Moisture (%)', 'Precipitation (mm)', 'Fertilizer Amount (kg)']]

# Targets
y_height = df['Plant Height (cm)']
y_leaf_area = df['Leaf Area (cm²)']

# Split the Data
X_train, X_test, y_height_train, y_height_test = train_test_split( *arrays: X, y_height, test_size=0.3, random_state=42)
_, _, y_leaf_area_train, y_leaf_area_test = train_test_split( *arrays: X, y_leaf_area, test_size=0.3, random_state=42)

# Train Models
height_model = RandomForestRegressor(n_estimators=100, random_state=42)
height_model.fit(X_train, y_height_train)

leaf_area_model = RandomForestRegressor(n_estimators=100, random_state=42)
leaf_area_model.fit(X_train, y_leaf_area_train)
```

train_Growth.py

```python
leaf_area_model = RandomForestRegressor(n_estimators=100, random_state=42)
leaf_area_model.fit(X_train, y_leaf_area_train)

# Evaluate Models
y_height_pred = height_model.predict(X_test)
height_mape = mean_absolute_percentage_error(y_height_test, y_height_pred) * 100
height_accuracy = 100 - height_mape

y_leaf_area_pred = leaf_area_model.predict(X_test)
leaf_area_mape = mean_absolute_percentage_error(y_leaf_area_test, y_leaf_area_pred) * 100
leaf_area_accuracy = 100 - leaf_area_mape

# Compute Overall Accuracy
overall_accuracy = (height_accuracy + leaf_area_accuracy) / 2


print(f"Overall Model Accuracy: {overall_accuracy:.2f}%")

# Save Models
joblib.dump(height_model, filename: 'height_model.pkl')
joblib.dump(leaf_area_model, filename: 'leaf_area_model.pkl')
```

Accuracy

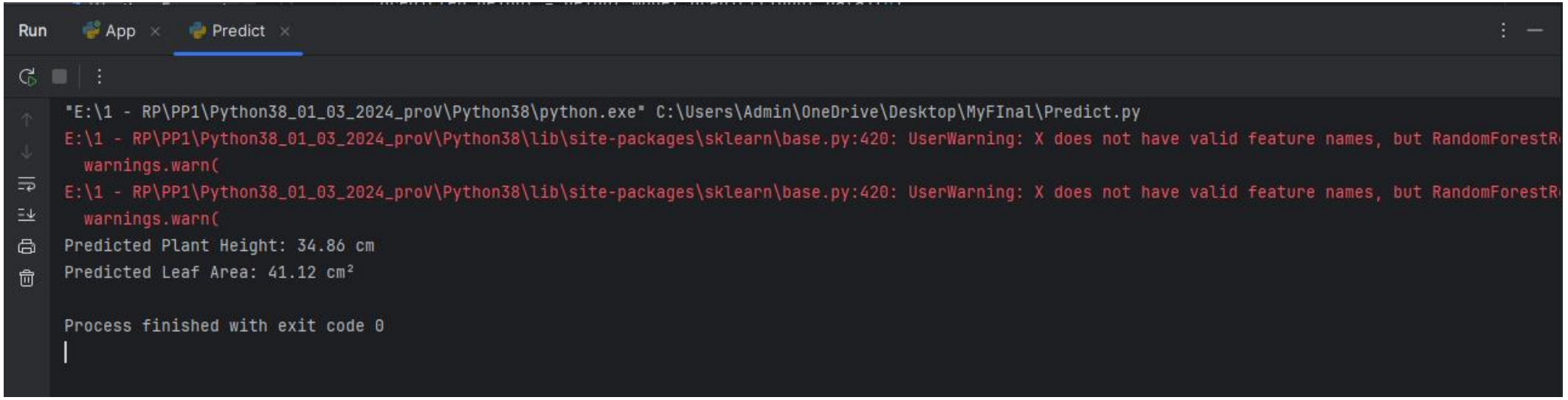

Run
App
train_Growth
"E:\1 - RP\PP1\Python38_01_03_2024_proV\Python38\python.exe" C:\Users\Admin\OneDrive\Desktop\MyFInal\train_Growth.py
Overall Model Accuracy: 93.04%
Process finished with exit code 0

BackEnd

Predict.py

```python
import joblib
import numpy as np

# rained models
height_model = joblib.load('height_model.pkl')
leaf_area_model = joblib.load('leaf_area_model.pkl')

# prediction
def predict_growth(soil_moisture, precipitation, fertilizer_amount):  1 usage

    input_data = np.array([[soil_moisture, precipitation, fertilizer_amount]])

    predicted_height = height_model.predict(input_data)[0]
    predicted_leaf_area = leaf_area_model.predict(input_data)[0]

    return predicted_height, predicted_leaf_area

# input
soil_moisture = 45
precipitation = 3
fertilizer_amount = 2.2

#  prediction
predicted_height, predicted_leaf_area = predict_growth(soil_moisture, precipitation, fertilizer_amount)
print(f"Predicted Plant Height: {predicted_height:.2f} cm")
print(f"Predicted Leaf Area: {predicted_leaf_area:.2f} cm²")

```

BackEnd


Run
App
Predict
"E:\1 - RP\PP1\Python38_01_03_2024_proV\Python38\python.exe" C:\Users\Admin\OneDrive\Desktop\MyFInal\Predict.py
E:\1 - RP\PP1\Python38_01_03_2024_proV\Python38\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but RandomForestR
warnings.warn(
E:\1 - RP\PP1\Python38_01_03_2024_proV\Python38\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but RandomForestR
warnings.warn(
Predicted Plant Height: 34.86 cm
Predicted Leaf Area: 41.12 cm²
Process finished with exit code 0

BackEnd

Weather_Forecast.py

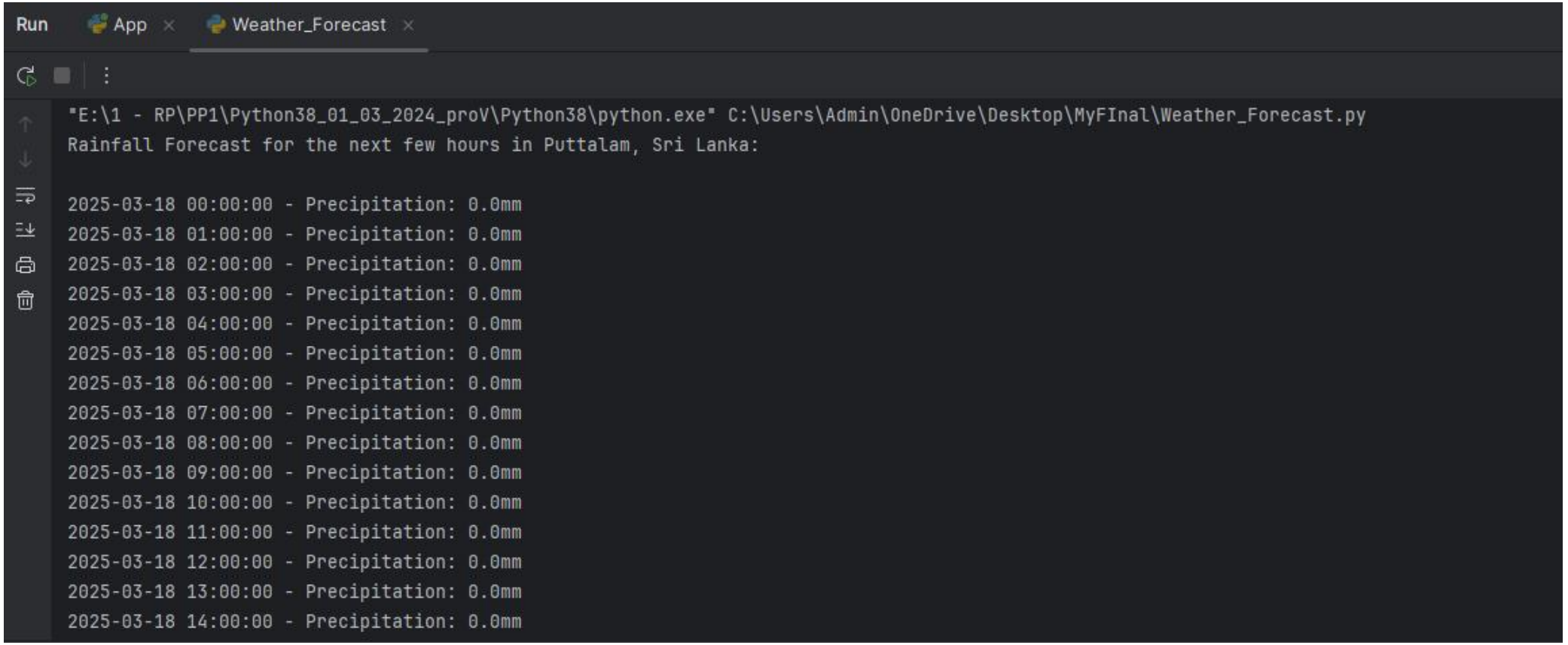
```python
import requests
import datetime
from urllib.parse import quote_plus

#  coordinates
latitude = 8.0
longitude = 79.8

# timezone
timezone = "Asia/Colombo"

# API URL
url = f"https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&hourly=precipitation&timezone={

# request to the Open-Meteo API
response = requests.get(url)

# Check if the response
if response.status_code == 200:
    data = response.json()

    # forecast for the next few hours
    print("Rainfall Forecast for the next few hours in Puttalam, Sri Lanka:\n")
    for i in range(len(data['hourly']['precipitation'])):
        timestamp = data['hourly']['time'][i]
        precipitation = data['hourly']['precipitation'][i]
```

BackEnd

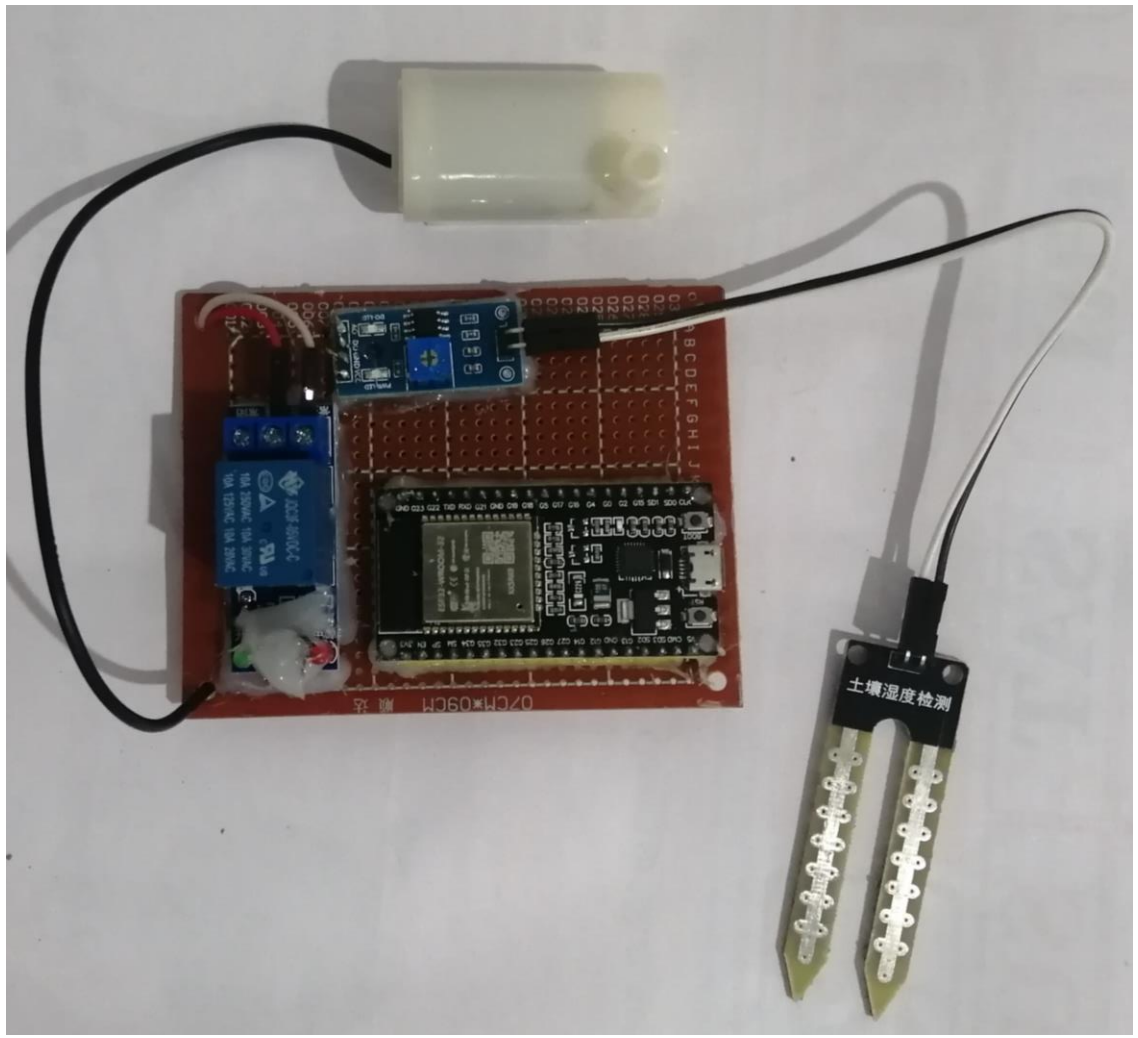
Weather_Forecast.py

```python
# request to the Open-Meteo API
response = requests.get(url)

# Check if the response
if response.status_code == 200:
    data = response.json()

    # forecast for the next few hours
    print("Rainfall Forecast for the next few hours in Puttalam, Sri Lanka:\n")
    for i in range(len(data['hourly']['precipitation'])):
        timestamp = data['hourly']['time'][i]
        precipitation = data['hourly']['precipitation'][i]

        date_time = datetime.datetime.strptime(timestamp, format: "%Y-%m-%dT%H:%M")
        formatted_time = date_time.strftime('%Y-%m-%d %H:%M:%S')

        print(f"{formatted_time} - Precipitation: {precipitation}mm")
else:
    print(f"Error fetching weather data. Status code: {response.status_code}")
    print(f"Response Text: {response.text}")
```

BackEnd


Run
App
Weather_Forecast
"E:\1 - RP\PP1\Python38_01_03_2024_proV\Python38\python.exe" C:\Users\Admin\OneDrive\Desktop\MyFInal\Weather_Forecast.py
Rainfall Forecast for the next few hours in Puttalam, Sri Lanka:
2025-03-18 00:00:00 - Precipitation: 0.0mm
2025-03-18 01:00:00 - Precipitation: 0.0mm
2025-03-18 02:00:00 - Precipitation: 0.0mm
2025-03-18 03:00:00 - Precipitation: 0.0mm
2025-03-18 04:00:00 - Precipitation: 0.0mm
2025-03-18 05:00:00 - Precipitation: 0.0mm
2025-03-18 06:00:00 - Precipitation: 0.0mm
2025-03-18 07:00:00 - Precipitation: 0.0mm
2025-03-18 08:00:00 - Precipitation: 0.0mm
2025-03-18 09:00:00 - Precipitation: 0.0mm
2025-03-18 10:00:00 - Precipitation: 0.0mm
2025-03-18 11:00:00 - Precipitation: 0.0mm
2025-03-18 12:00:00 - Precipitation: 0.0mm
2025-03-18 13:00:00 - Precipitation: 0.0mm
2025-03-18 14:00:00 - Precipitation: 0.0mm

Prototype



Completed

- Built prototype 80%
- Integrate Tools
- Model Training

To be Complete

- Connect the prototype with Database
- Web UI Design (Frontend)
- Testing

References

[1]https://ieeexplore.ieee.org/document/9754145

[2]https://www.sciencedirect.com/science/article/abs/pii/S2214785321042115

[3] Pettorelli, N. (2013). The Normalization of Vegetation Indices. Springer.

[4] Rouse, J. W., Haas, R. H., Schell, J. A., & Deering, D. W. (1974). Monitoring Vegetation Systems in the Great Plains with ERTS. NASA.

IT21290206 - Mayootharan. P

Information Technology

Predictive Analytics for Crop Yield, Harvest Readiness, and Fertilizer Optimization

Introduction

01 Background

02 Research Problem

03 Objectives

Background

Advances in precision agriculture are transforming traditional farming methods by leveraging IoT-based technologies and data analytics. This research focuses on developing a system that uses real-time data from sensors and agricultural department datasets to predict crop yield, harvest readiness, and fertilizer requirements for optimized resource use and productivity enhancement.

Research Problem

Modern agriculture faces challenges such as unpredictable yields, inefficient resource utilization, and inconsistent harvest readiness. Manual methods of monitoring soil and environmental conditions are time-consuming and prone to errors. Additionally, over- or under-application of fertilizers contributes to soil degradation and economic losses. This research addresses these issues by using IoT-based solutions to improve efficiency, accuracy, and sustainability in farming.

OBJECTIVES

To develop a model for efficient yield prediction, harvest readiness, and fertilizer optimization using sensor data and agricultural datasets.

SUB - OBJECTIVES

- Monitor soil and environmental conditions using sensors.
- Implement predictive models for yield estimation and harvest timing.
- Develop a model to predict the fertilizer plan.
- Create a real-time notification system for farmers regarding harvest readiness.
- Test the system under various environmental conditions to ensure accuracy and reliability.

Methodology

01 Data Collection

02 Data Processing

Clean and preprocess sensor and historical data.
Store data in a centralized system using ESP32 for wireless communication.

03 Model Development

Develop machine learning models for yield prediction, harvest readiness, and fertilizer optimization.
Train models using historical and real-time data.

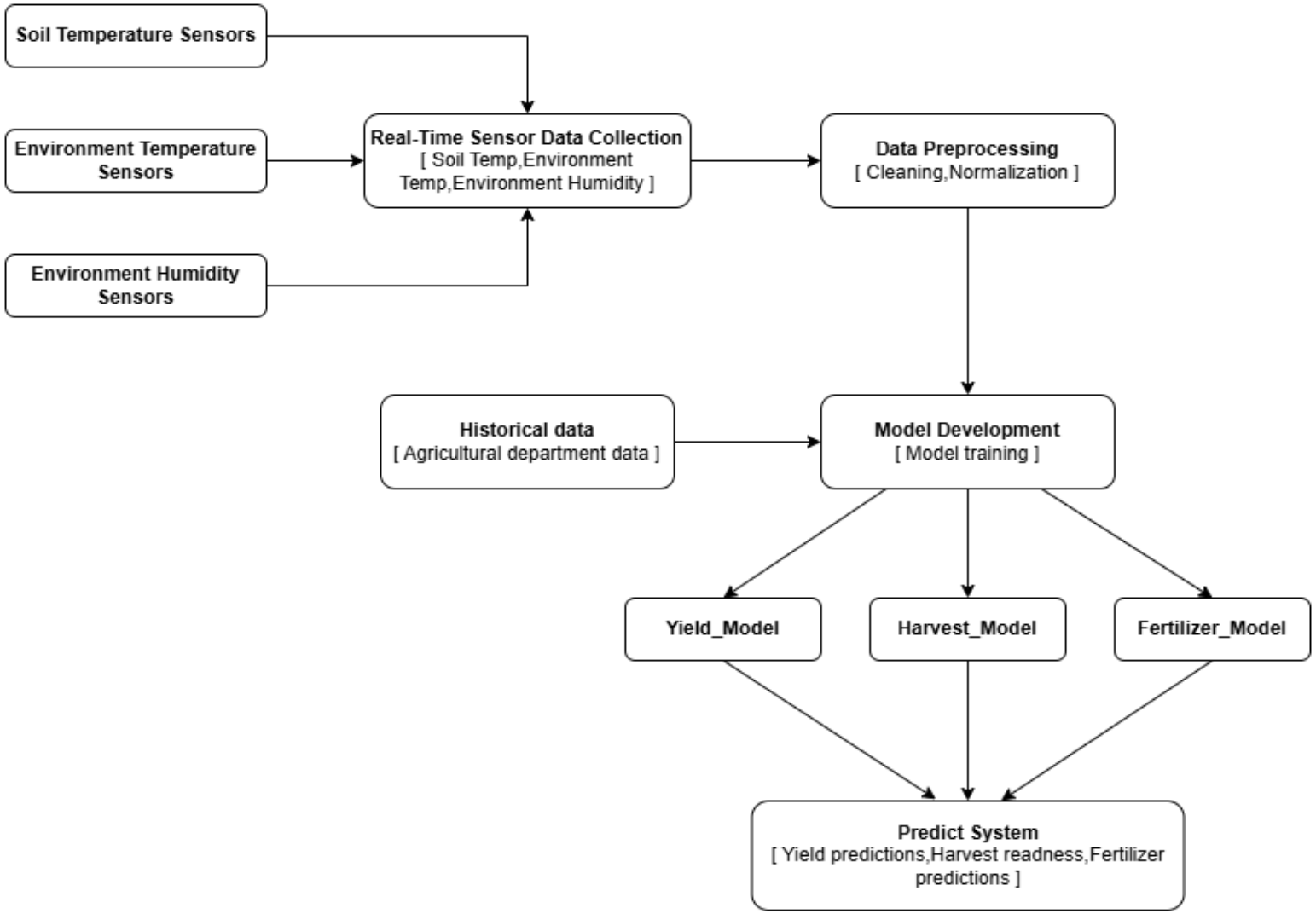
04 System Implementation

Integrate predictions with a user interface
Set up notification systems for alerts and recommendations.

05 Validation and Optimization

Refine models to improve accuracy.

System Diagram



Technologies

- Python

- Flask

- Firebase

- Arduino IDE
- PyCharm

Requirements

Functional

- Monitor soil temperature, environmental temperature, and humidity in real time.
- Analyze collected data to predict crop yield and determine harvest readiness.
- Data processing

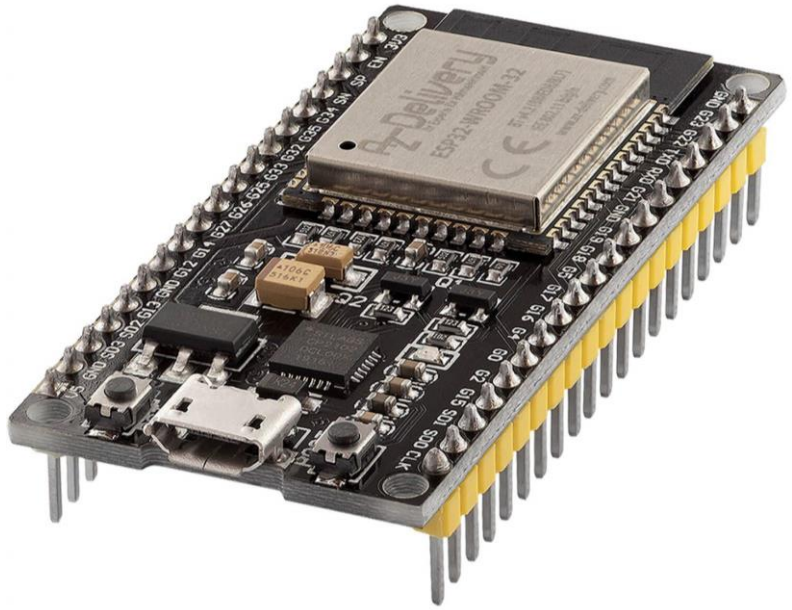
Non Funcational

- **Accuracy**: The system must provide precise predictions and recommendations.
- **Reliability**: Ensure consistent data collection and processing.
- **Cost-Effectiveness**: Use affordable components and minimize operational costs.
- **User-Friendliness**: Interfaces should be simple for farmers with minimal technical expertise.
- **Security**: Safeguard sensitive data and communications.

Tools

IOT-Device Circuits Items

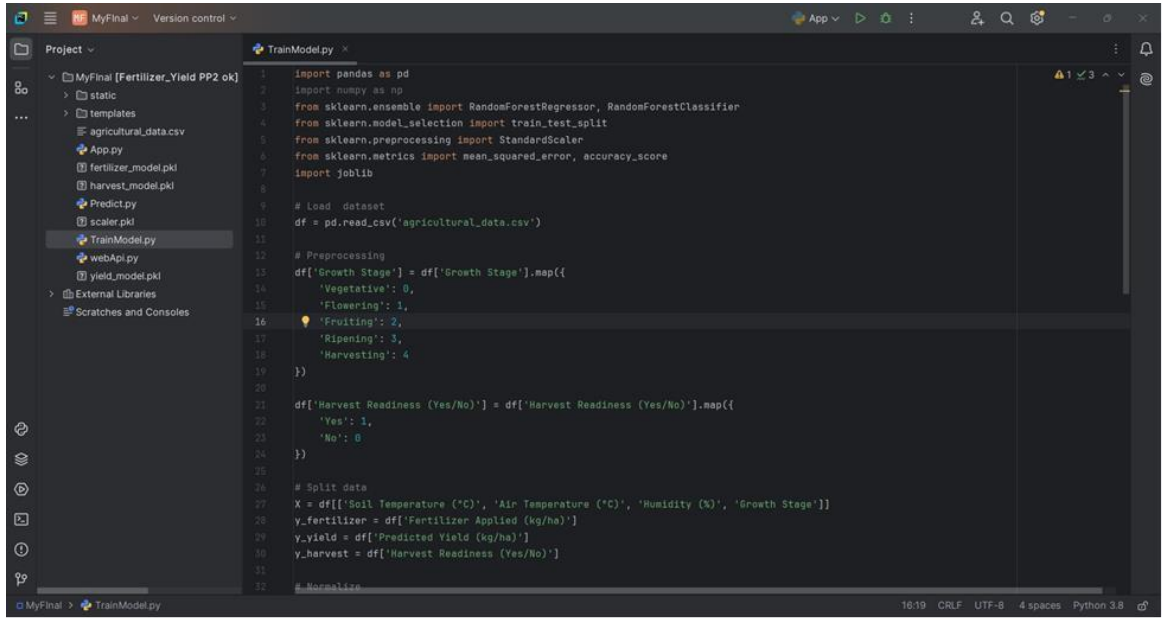
Soil Temperature Sensor :
DS18B20 or similar



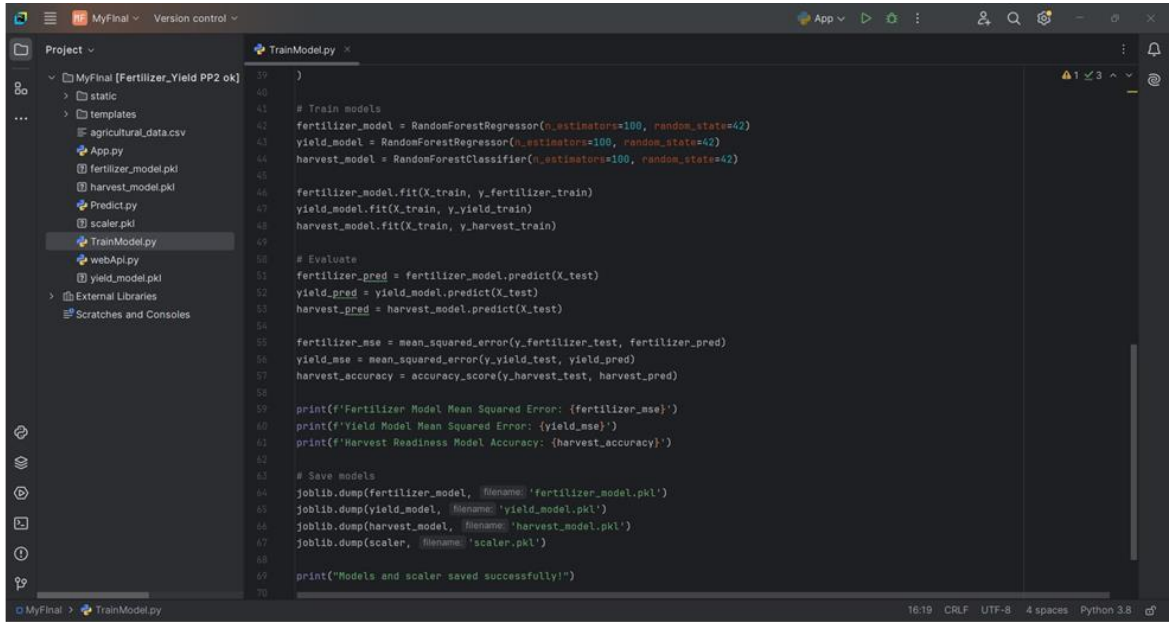
Environment Temperature sensor ,Environment humidity sensor :
DHT22/DHT11

ESP32 Microcontroller:
For data processing and communication.

Model

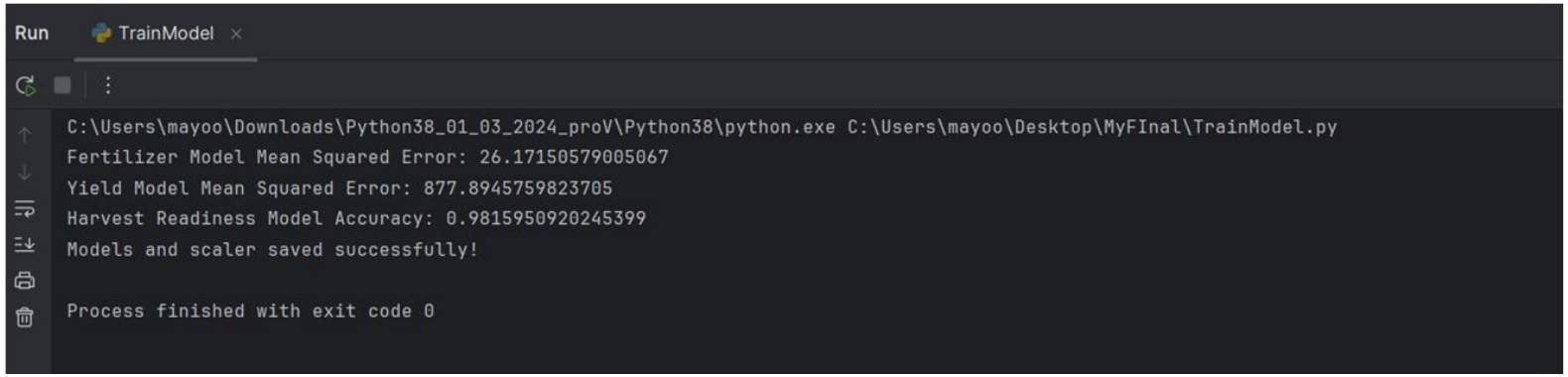


```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, accuracy_score
import joblib

# Load dataset
df = pd.read_csv('agricultural_data.csv')

# Preprocessing
df['Growth Stage'] = df['Growth Stage'].map({
    'Vegetative': 0,
    'Flowering': 1,
    'Fruiting': 2,
    'Ripening': 3,
    'Harvesting': 4
})

df['Harvest Readiness (Yes/No)'] = df['Harvest Readiness (Yes/No)'].map({
    'Yes': 1,
    'No': 0
})

# Split data
X = df[['Soil Temperature (°C)', 'Air Temperature (°C)', 'Humidity (%)', 'Growth Stage']]
y_fertilizer = df['Fertilizer Applied (kg/ha)']
y_yield = df['Predicted Yield (kg/ha)']
y_harvest = df['Harvest Readiness (Yes/No)']

# Normalize
```

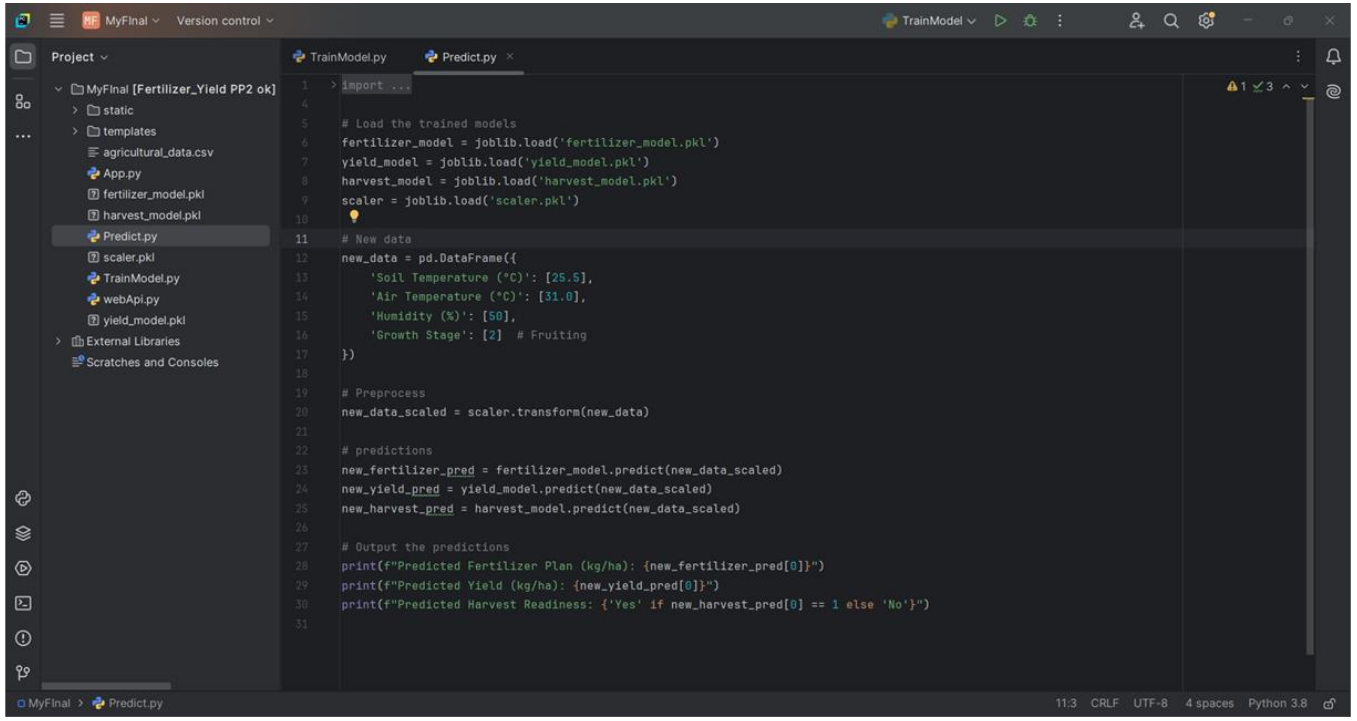


```python
)

# Train models
fertilizer_model = RandomForestRegressor(n_estimators=100, random_state=42)
yield_model = RandomForestRegressor(n_estimators=100, random_state=42)
harvest_model = RandomForestClassifier(n_estimators=100, random_state=42)

fertilizer_model.fit(X_train, y_fertilizer_train)
yield_model.fit(X_train, y_yield_train)
harvest_model.fit(X_train, y_harvest_train)

# Evaluate
fertilizer_pred = fertilizer_model.predict(X_test)
yield_pred = yield_model.predict(X_test)
harvest_pred = harvest_model.predict(X_test)

fertilizer_mse = mean_squared_error(y_fertilizer_test, fertilizer_pred)
yield_mse = mean_squared_error(y_yield_test, yield_pred)
harvest_accuracy = accuracy_score(y_harvest_test, harvest_pred)

print(f'Fertilizer Model Mean Squared Error: {fertilizer_mse}')
print(f'Yield Model Mean Squared Error: {yield_mse}')
print(f'Harvest Readiness Model Accuracy: {harvest_accuracy}')

# Save models
joblib.dump(fertilizer_model, filename: 'fertilizer_model.pkl')
joblib.dump(yield_model, filename: 'yield_model.pkl')
joblib.dump(harvest_model, filename: 'harvest_model.pkl')
joblib.dump(scaler, filename: 'scaler.pkl')

print("Models and scaler saved successfully!")
```

Accuracy

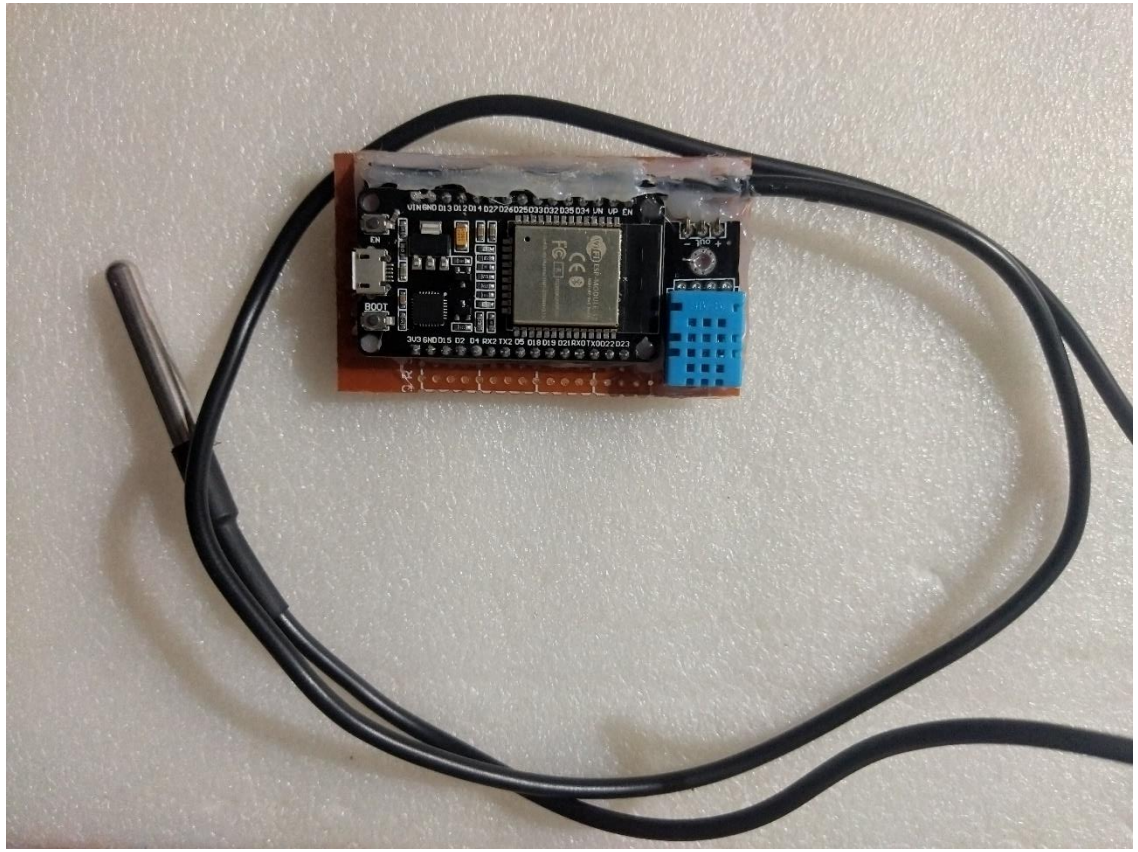

Run
TrainModel
C:\Users\mayoo\Downloads\Python38_01_03_2024_proV\Python38\python.exe C:\Users\mayoo\Desktop\MyFInal\TrainModel.py
Fertilizer Model Mean Squared Error: 26.17150579005067
Yield Model Mean Squared Error: 877.8945759823705
Harvest Readiness Model Accuracy: 0.9815950920245399
Models and scaler saved successfully!
Process finished with exit code 0

Backend

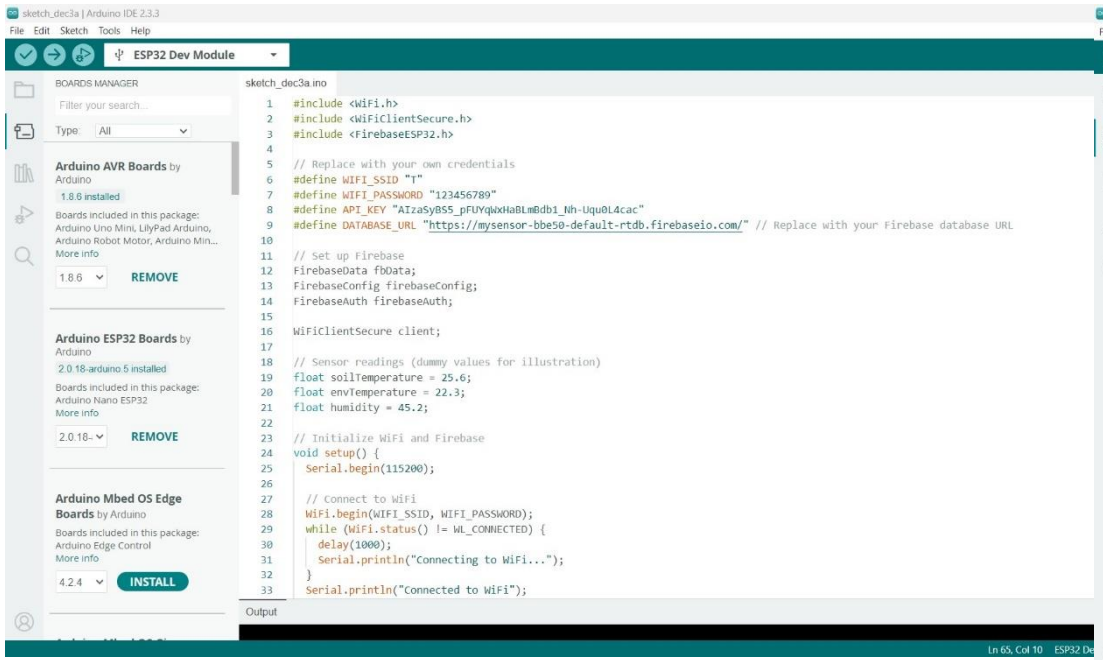


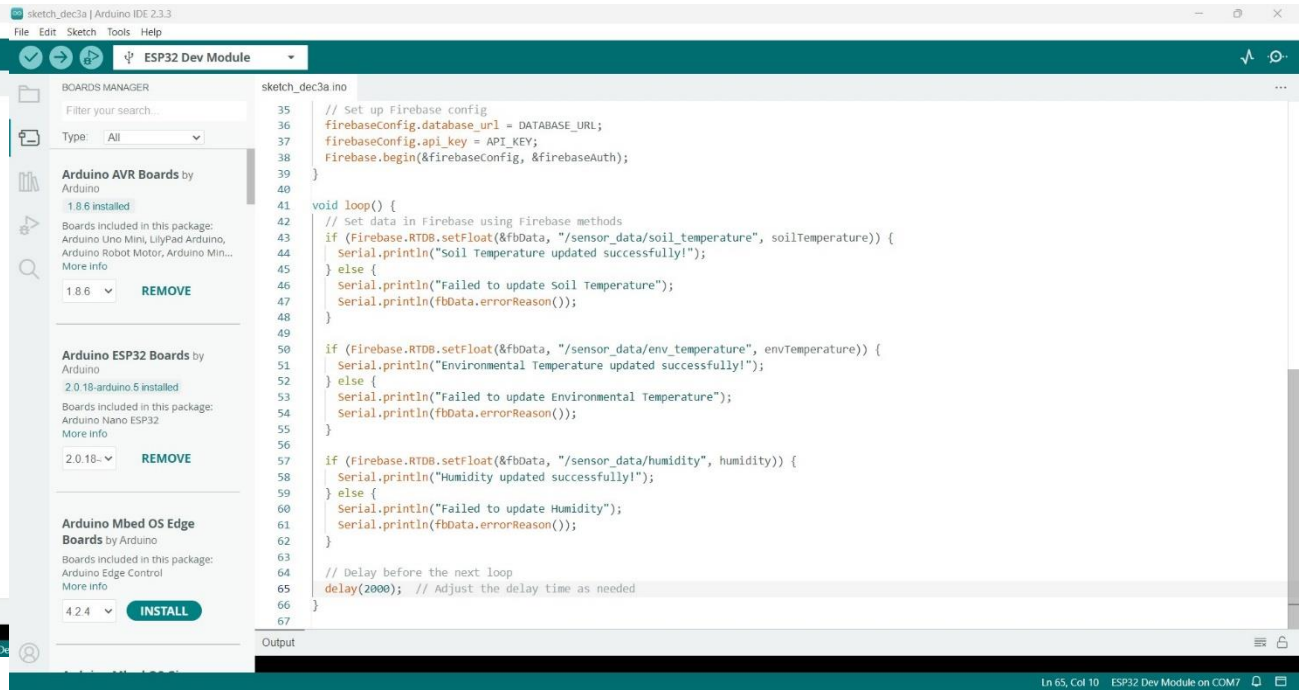

```
C:\Users\mayoo\Downloads\Python38_01_03_2024_proV\Python38\python.exe C:\Users\mayoo\Desktop\MyFInal\Predict.py
Predicted Fertilizer Plan (kg): 103.7718076828001
Predicted Yield (kg): 848.9043078939992
Predicted Harvest Readiness: Yes

Process finished with exit code 0
```

Prototype



Prototype Code





Prototype Result

Completed

- Built prototype 80%
- Integrate Sensors & Tools
- Model Development
- Backend

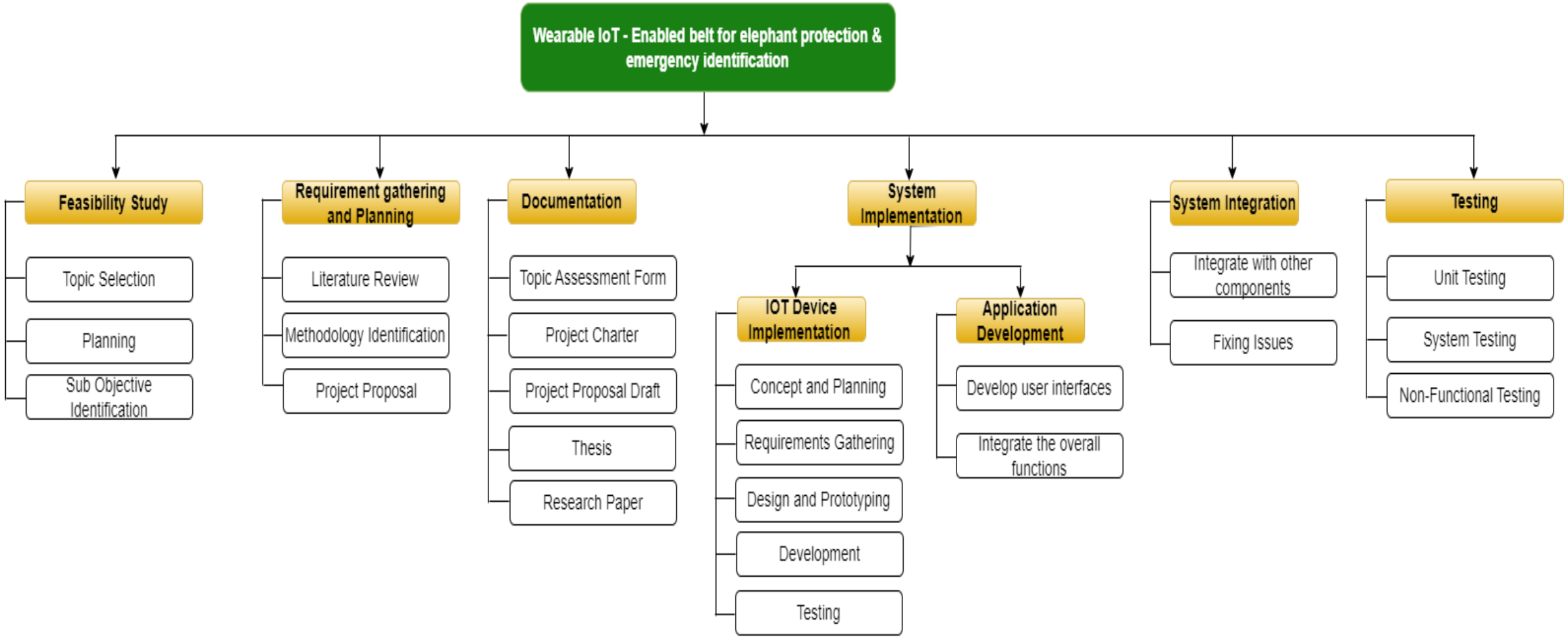
To be Complete

- Connect the prototype with Database
- Web UI Design (Frontend)
- Testing

References

[1]
https://ieeexplore.ieee.org/document/9754145
[2]
https://www.sciencedirect.com/science/article/abs/pii/S2214785321042115
[3] Tsaftaris, S. A., & Blomberg, S. (2018). Machine Learning for Crop Yield Prediction. Springer.
[4] Yang, X., & Li, H. (2018). Remote Sensing for Precision Agriculture: Yield Prediction and Harvest Management. Wiley.

Work Breakdown Structure

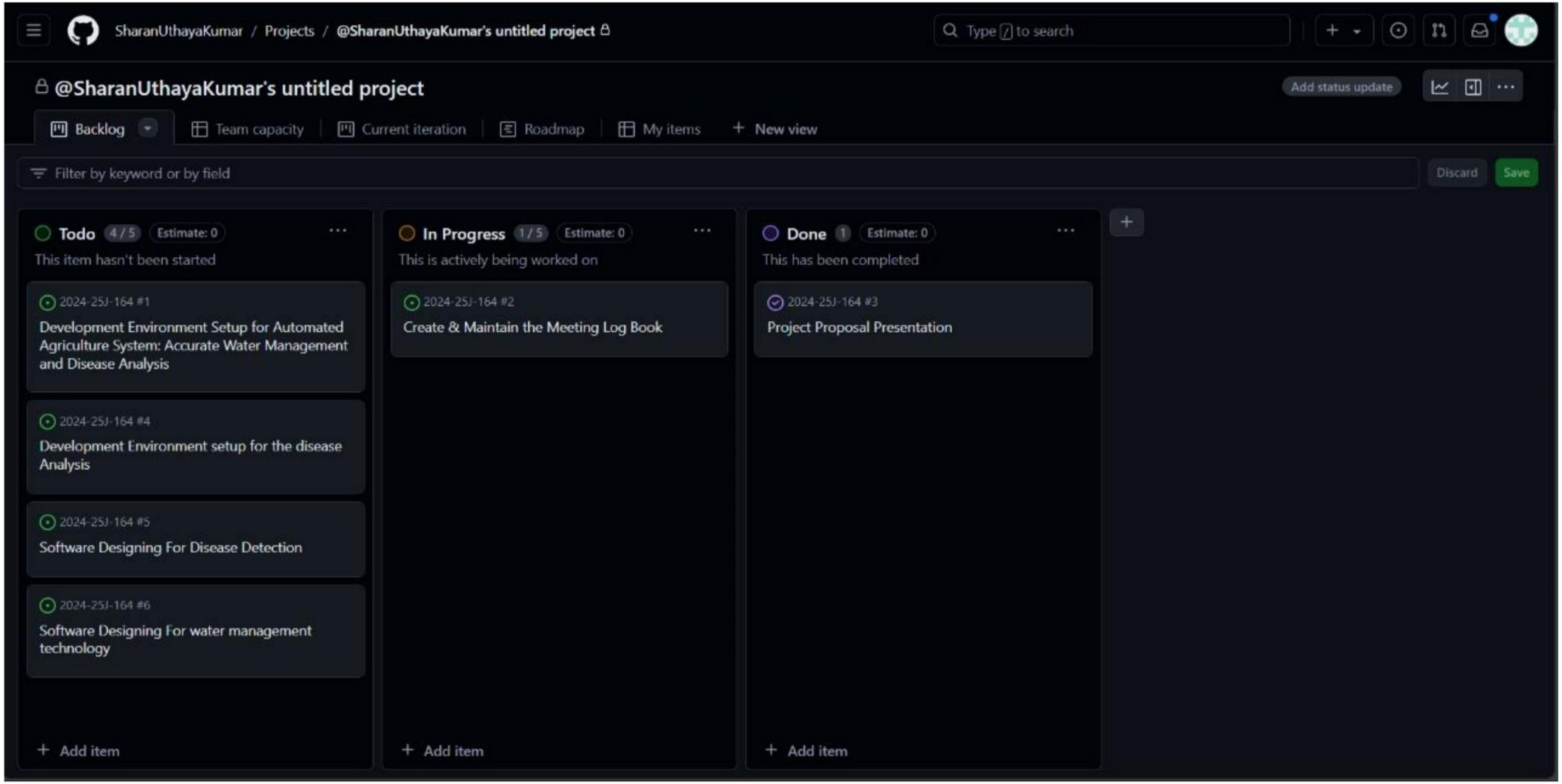


Cost Management Plan

Budget

Description	Cost Rs
IOT – Device Budget	40,000
Other Expenses Budget	25,000
Total	**65,000**

Project Management Plan




Q
A

Thanks