SMART GREEN HOUSES DECISION SUPPORT SYSTEMS FOR TOMATO CULTIVATION

GROUP ID: 24-25J-064

SLIIT
FACULTY OF COMPUTING

INTRODUCTION

- Maintaining optimal conditions in tomato greenhouses is maximizing the yield and quality.
- Managing watering, fertilizing, and harvesting efficiently supports sustainable tomato farming.
- Machine learning can predicts watering needs, fertilization, harvest times, disease identification and treatment recommendations.
- Combining IoT and ML, the project aims to make greenhouse management more efficient

OBJECTIVES OF MEMBERS

Optimize Irigation control & Watering Schedules :

- Determine the best timing and frequency for watering to ensure healthy growth and efficient water use.

Optimize Harvesting Schedules :

- Establish the ideal timing for harvesting tomatoes to maximize yield and quality.

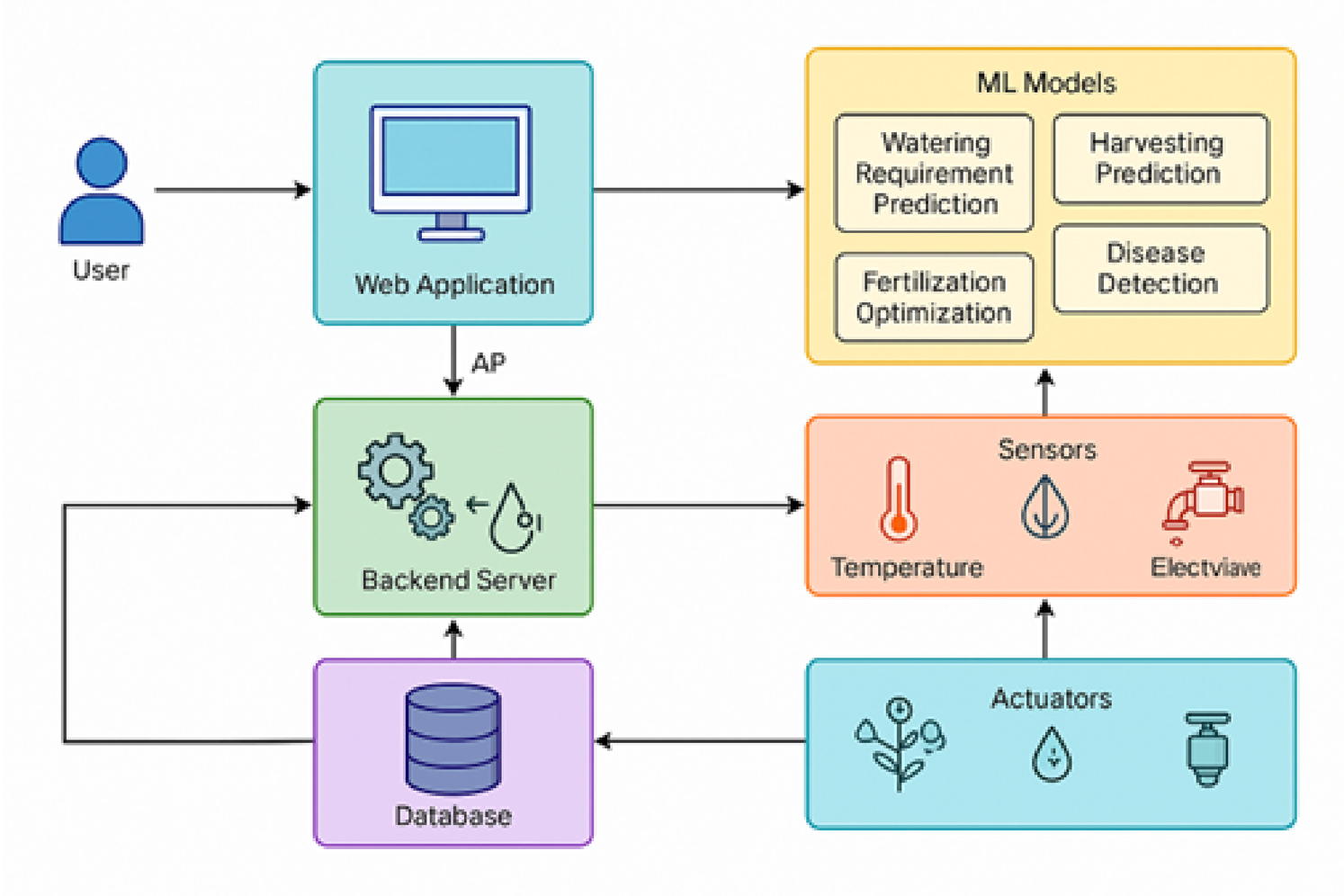
Optimize Fertilizing Schedules and Types :

- Identify the most effective fertilizing routines and products to enhance nutrient availability and promote robust tomato plants.

Disease identification and treatment recommendations

- •Develop ml model to identify, prevent, and treat common tomato diseases and provide recommendations to maintain plant health.

FULL SYSTEM ARCHITECTURE DIAGRAM



GROUP MEMBERS AND SUPERVISORS DETAILES

1. THRIMAVITHANA .V .D - IT21181160
2. ASARDEEN .A - IT21231896
3. NAJAS .M .N .M - IT21186592
4. JAYANETHTHI .I .H .N .S - IT21231414

SUPERVISOR – Mrs. Geethanjali Wimalaratne

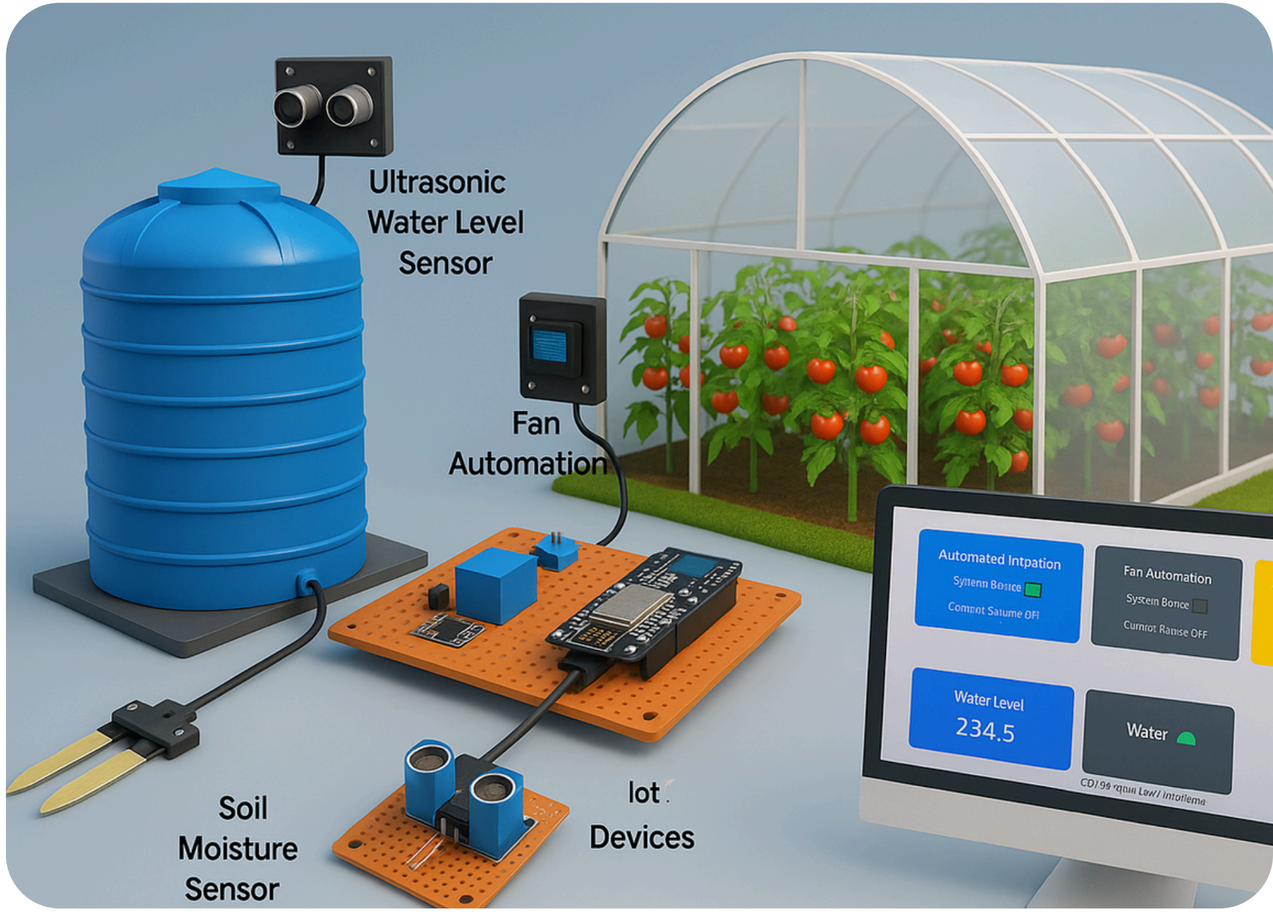
CO-SUPERVISOR – Mr. Samantha Thelijjagoda

IT21231896 | ASARDEEN .A

Information technology

MY COMPONENT

IoT and Machine Learning-Based Water Requirement Prediction and Automation Control System for Greenhouse Irrigation



INTRODUCTION BACKGROUND

- **Water management** is vital for tomato greenhouses.
- Environmental factors like **temperature, humidity, light,** and **soil moisture** directly impact plant water needs.
- **Photosynthesis** efficiency depends on:
  - Adequate **light, CO_2, temperature,** and **water**.
  - Poor irrigation leads: **limited energy production**, affecting growth & yield.
- **IoT + ML models** monitor real-time data to maintain optimal conditions that **enhance photosynthesis, improve crop health,** and **boost resource efficiency.**

21°C - 27°C

Problem Statement:

Manual and static irrigation leads to:

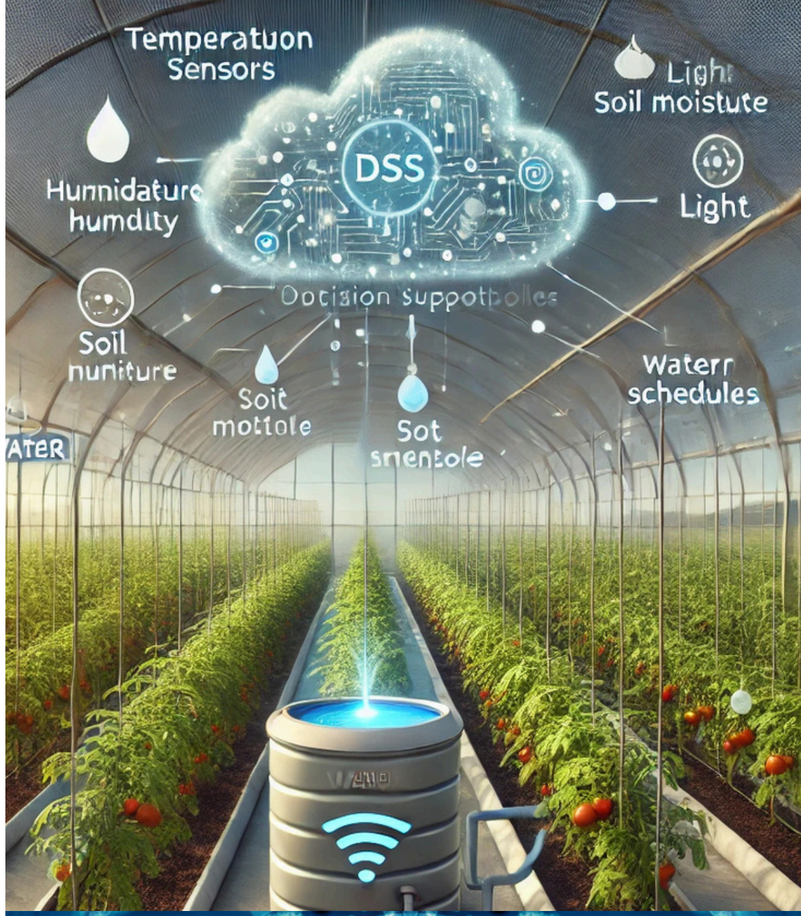
- Over or under watering.
- Poor yield and water wastage.
- High labor costs.

Existing systems lack:

- Real-time prediction.
- Integration of multiple environmental factors.

RESEARCH QUESTIONS

How can we develop a decision support system that accurately predicts optimal water requirements and ensures water availability based on real-time environmental conditions?



RESEARCH GAP

Existing Systems / Research	Environmental Monitoring	Water Availablity Monitoring & Control	machine learning integration	Overflow Detection	Water recruitment prediction	Automation Features & Control	Real time dashboard
IoT-Based Precision Irrigation System	✅	✅	❌	❌	❌	✅	❌
Smart Tomato Cultivation System	✅	✅	❌	❌	❌	✅	✅
Indoor Seed Germination System	✅	❌	❌	❌	❌	❌	❌
LoRaWAN-Based IoT Monitoring	✅	❌	❌	❌	❌	❌	✅
Proposed System (This Study)	✅	✅	✅	✅	✅	✅	✅

MAIN & SPECIFIC OBJECTS

Main objectives

To design a smart, IoT-based decision support system that utilizes machine learning to predict precise watering needs in tomato greenhouses, automate environmental controls, and provide real-time monitoring, control, and historical analysis.

Sub objectives

- 🌱 **Develop ML Models to Predict Irrigation Needs**
  Utilize environmental data and ML models to predict water required per plant.

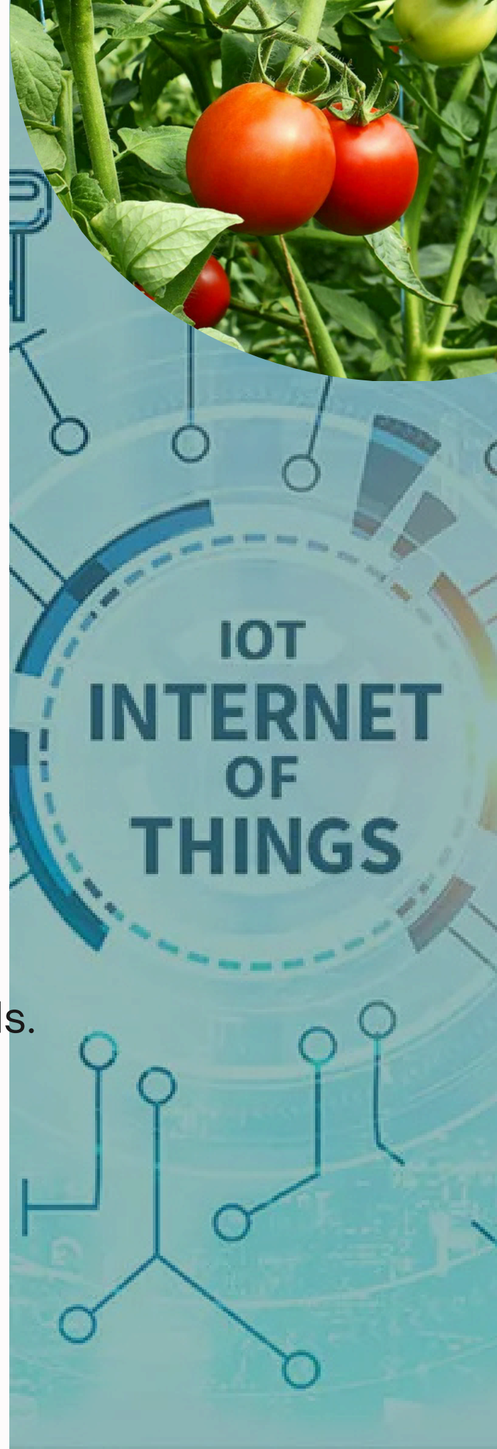
- 📡 **Develop IoT Device with Integrated Real-Time Sensors**
  - ▸Capture live temperature, humidity, light, soil moisture, and water level data via Firebase
  - ▸Monitor real time water availability using an ultrasonic sensor.

.

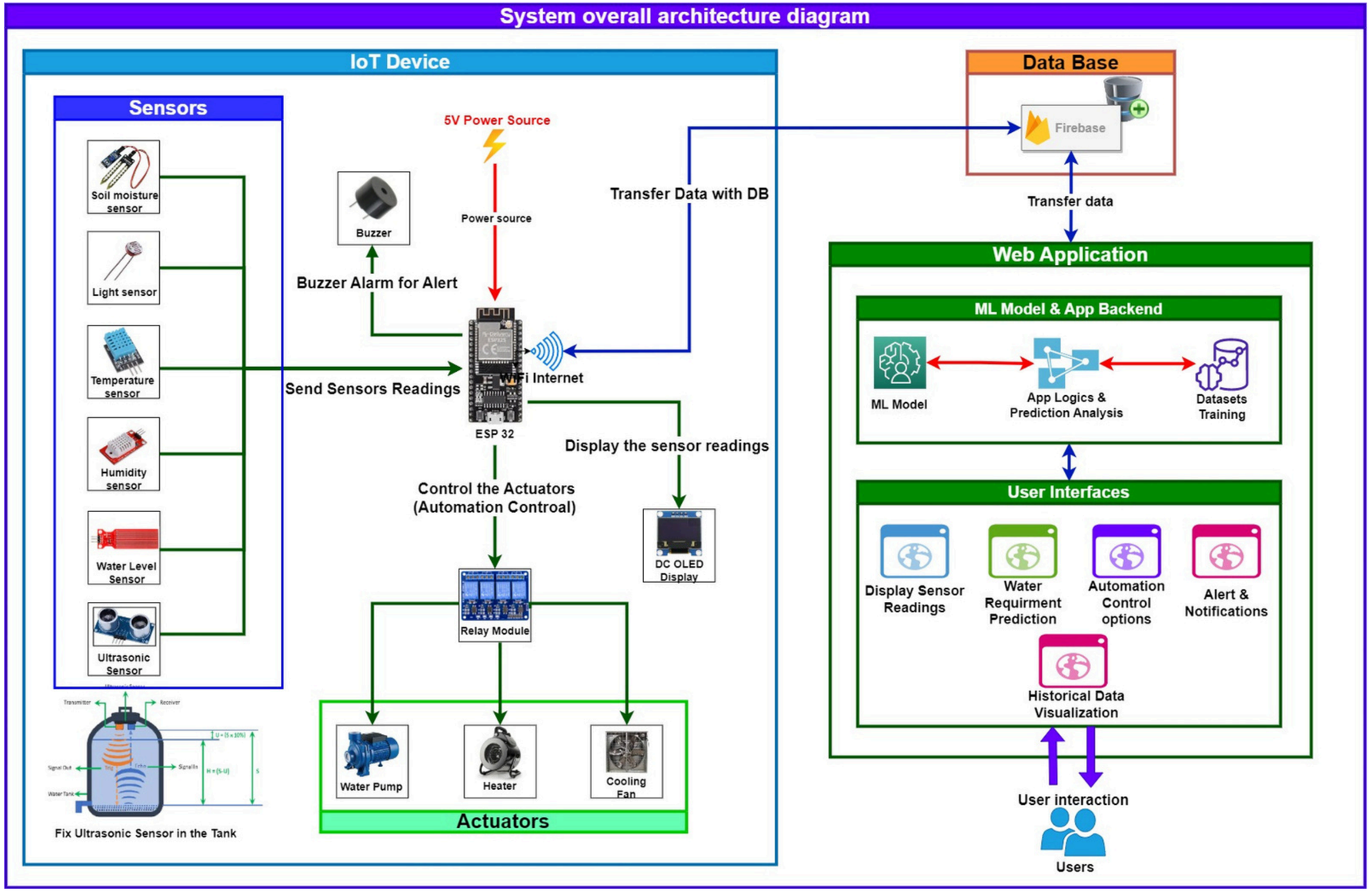
- 🔁 **Enable Automated Controls**
  Control irrigation, fans, Ventilation and alarms, based on live conditions and logic thresholds.

- 📊 **Data Logging & Visualization**
  Maintain and display historical water predictions using charts and tables.

- 💻 **User-Friendly & Responsive Web Dashboard**
  Provide authenticated access for users to view, control, and download system data.



METHODOLOGY - DIAGRAM



Sensors → ESP32 → Firebase → ML model → Web Dashboard

SLIIT FACULTY OF COMPUTING

IT21231896 | ASARDEEN .A | 24-25J-064

100% COMPLETION OF THE PROJECT

🧠 ML Model Development, Trained & Validation

⚙️ Automation Features Implementation

🔗 Complete System Integration & Testing

📡 IoT Device Development & Firebase Integration

🌐 Web Application with Real-Time Dashboard

📈 History Logging & Data Visualization

ABOUT THE IOT DEVELOPMENT

IoT Code

```
// Pin definitions for sensors
#define DHTPIN 15
#define DHTTYPE DHT11
#define TRIG_PIN 27
#define ECHO_PIN 18
#define SIGNAL_PIN 34
#define LIGHT_SENSOR_PIN 33

// OLED display parameters
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define OLED_I2C_ADDRESS 0x3C

// Initialize DHT sensor and OLED
DHT dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 display(SCREEN_WIDTH, SC
  // Read DHT sensor data
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read ultrasonic sensor
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  float distance = (duration * 0.034) / 2;

  // Read soil moisture sensor
  int soilMoisture = analogRead(SIGNAL_PIN);

  // Read light sensor
  int lightLevel = analogRead(LIGHT_SENSOR_PIN);
```

```
// Display data on OLED
display.clearDisplay();
display.setCursor(0, 0);
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.print("Temp: ");
display.print(temperature);
display.println(" C");
display.print("Hum: ");
display.print(humidity);
display.println(" %");
display.print("Dist: ");
display.print(distance);
display.println(" cm");
display.print("Soil: ");
display.print(soilMoisture);
display.println();
display.print("Light: ");
display.print(lightLevel);
display.display();
```

```
// Log data to serial
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println("°C");
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println("%");
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
Serial.print("Soil Moisture: ");
Serial.println(soilMoisture);
Serial.print("Light Level: ");
Serial.println(lightLevel);
```

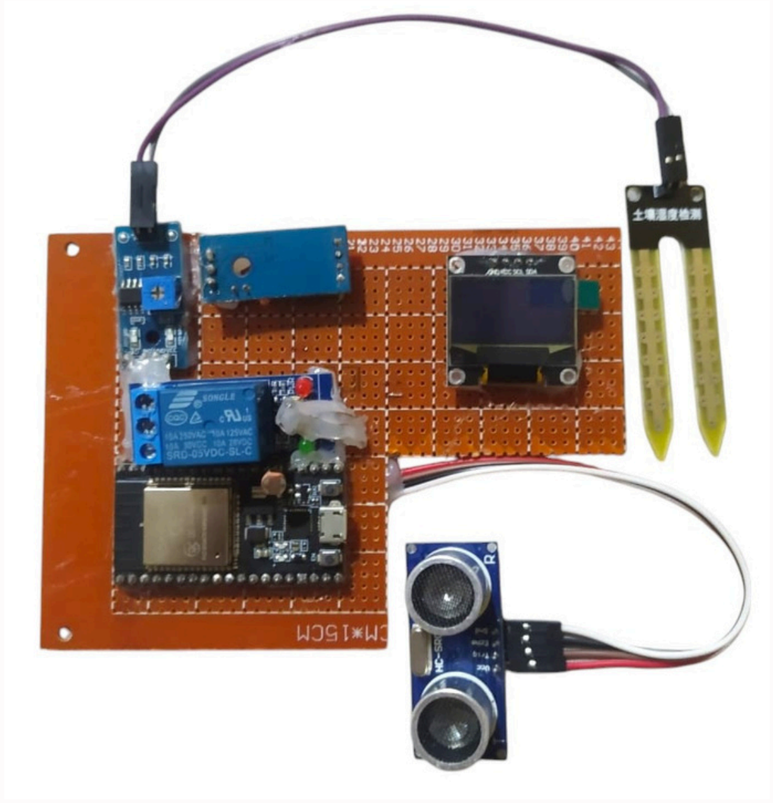
IoT Firebase Configurations

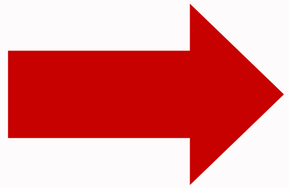
```
// Initialize Firebase
config.api_key = API_KEY;
config.database_url = DATABASE_URL;

if (Firebase.signUp(&config, &auth, "", "")) {
  Serial.println("Firebase authentication succeeded");
  signupOK = true;
} else {
  Serial.printf("Firebase signup failed: %s\n", config.signer.s
}
Firebase.begin(&config, &auth);
```

```
// Upload data to Firebase
if (Firebase.ready() && signupOK && millis() - sendDataPrevMillis > 2000) {
  sendDataPrevMillis = millis();

  Firebase.RTDB.setFloat(&fbdo, "/sensors/temperature", temperature);
  Firebase.RTDB.setFloat(&fbdo, "/sensors/humidity", humidity);
  Firebase.RTDB.setFloat(&fbdo, "/sensors/distance", distance);
  Firebase.RTDB.setInt(&fbdo, "/sensors/soilMoisture", soilMoisture);
  Firebase.RTDB.setInt(&fbdo, "/sensors/lightLevel", lightLevel);

  if (fbdo.httpCode() != 200) {
    Serial.print("Firebase failed: ");
    Serial.println(fbdo.errorReason());
  } else {
    Serial.println("Data uploaded to Firebase");
  }
}
```

IOT DEVICE

Previous version of the IoT device

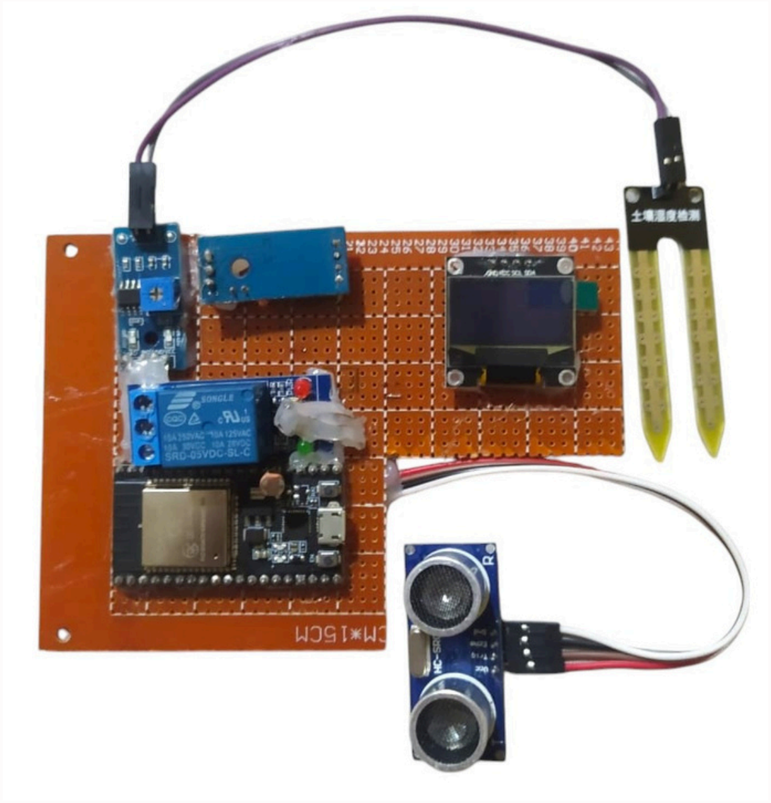


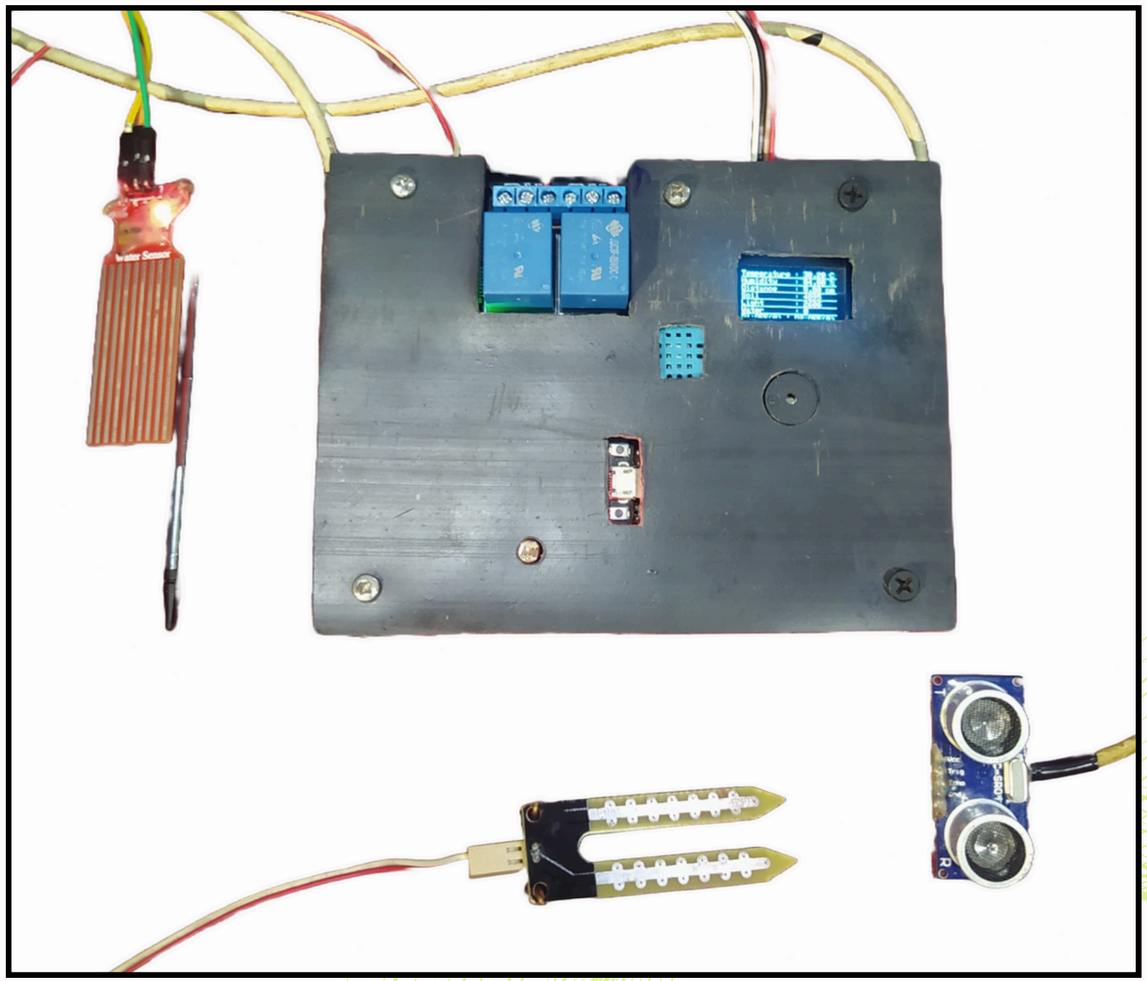


IOT DEVICE

Updated Version of The IoT

IoT device being successfully developed and covered with a casing for safety.





ABOUT THE DATASET

The dataset captures real-time environmental factors affecting tomato cultivation, including Timestamp, Temperature (°C), Humidity (%), Soil Moisture (%), Light Level (lux), and Water Needed (Liters), to optimize watering schedules in the Smart Greenhouse Decision Support System.

Timestamp	Temperature (Â°C)	Humidity (%)	Soil Moisture (%)	Light Level (lux)	Water Level Needed (Liters)
11/1/2024 8:00	27.5	69.6	10.5	903	4.8
11/1/2024 9:00	39	69.5	36.5	1696	11.2
11/1/2024 10:00	34.6	34.6	17.1	966	6.9
11/1/2024 11:00	32	54.7	48.4	1183	12.9
11/1/2024 11:00	23.1	32.9	15.9	517	5.5
11/1/2024 13:00	23.1	57.5	26.6	609	7.6
11/1/2024 14:00	21.2	52.1	13.4	1089	4.8
11/1/2024 15:00	37.3	74.4	49.9	1220	13.7
11/1/2024 16:00	32	47.5	30.1	1400	9.2
11/1/2024 17:00	34.2	35.9	33.8	937	10.2
11/1/2024 18:00	20.4	37.1	12.7	1542	4.6
11/1/2024 19:00	39.4	68.1	40	1790	11.9
11/1/2024 20:00	36.6	60.9	18.4	1670	7.3
11/1/2024 21:00	24.2	35.1	45.9	559	11.6
11/1/2024 22:00	23.6	34.2	18.2	1221	6
11/1/2024 23:00	23.7	65	17.6	657	5.9
11/2/2024 0:00	26.1	33.6	11.5	863	4.9
11/2/2024 1:00	30.5	71.1	28.9	1980	8.8

DATA PREPROCESSING & FEATURE ENGINEERING

```
# Preprocess the data
data['Timestamp'] = pd.to_datetime(data['Timestamp'])

# Feature Engineering
data['Hour'] = data['Timestamp'].dt.hour
data['DayOfWeek'] = data['Timestamp'].dt.dayofweek
data['Month'] = data['Timestamp'].dt.month
data['Year'] = data['Timestamp'].dt.year
```

```
# Load dataset
file_path = "irrigation_dataset.csv"
data = pd.read_csv(file_path)

# Preprocess the data
data['Timestamp'] = pd.to_datetime(data['Timestamp'])

# Feature Engineering
data['Hour'] = data['Timestamp'].dt.hour
data['DayOfWeek'] = data['Timestamp'].dt.dayofweek
data['Month'] = data['Timestamp'].dt.month
data['Year'] = data['Timestamp'].dt.year

# Handle missing values
imputer = SimpleImputer(strategy='median')
data[['Temperature (°C)', 'Humidity (%)', 'Soil Moisture (%)', 'Light Level (lux)']] = imputer.fit_transform(
    data[['Temperature (°C)', 'Humidity (%)', 'Soil Moisture (%)', 'Light Level (lux)']]
```

```
# Feature Scaling and Polynomial Expansion
# Feature Scaling
scaler = StandardScaler()
features = ['Temperature (°C)', 'Humidity (%)', 'Soil Moisture (%)', 'Light Lev
data[features] = scaler.fit_transform(data[features])

# Polynomial Feature Expansion
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(data[features])
```

```
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Timestamp                   420 non-null    datetime64[ns]
 1   Temperature (°C)            420 non-null    float64
 2   Humidity (%)                420 non-null    float64
 3   Soil Moisture (%)           420 non-null    float64
 4   Light Level (lux)           420 non-null    float64
 5   Water Level Needed (Liters) 420 non-null    float64
 6   Hour                        420 non-null    int64
 7   DayOfWeek                   420 non-null    int64
 8   Month                       420 non-null    int64
 9   Year                        420 non-null    int64
dtypes: datetime64[ns](1), float64(5), int64(4)
memory usage: 32.9 KB
```

MODEL TRAINING

```python
# Split data into training
X = X_poly
y = data['Water Level Needed (Liters)']
X_train, X_temp, y_train, y_temp = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split( *arrays: X_temp, y_temp, test_size=0.5, random_state=42)

# Hyperparameter Tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1,
                           verbose=2,
                           scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
```

```python
# Best parameters
print("Best Hyperparameters found:", grid_search.best_params_)

# Train the model
best_model = grid_search.best_estimator_
y_val_pred = best_model.predict(X_val)

# performance metrics
val_rmse = mean_squared_error(y_val, y_val_pred, squared=False)
val_r2 = r2_score(y_val, y_val_pred)
val_mae = mean_absolute_error(y_val, y_val_pred)

print(f"Validation RMSE: {val_rmse:.2f}")
print(f"Validation R2 Score: {val_r2:.2f}")
print(f"Validation MAE: {val_mae:.2f}")
```

```python
# Save Models
joblib.dump(best_model, filename: "irrigation_model.pkl")
joblib.dump(scaler, filename: "scaler.pkl")
joblib.dump(poly, filename: "poly.pkl")
print("Best model, scaler, and poly transformer saved successfully.")
```

PERFORMANSE MATRIC ACCURECY :

```
Validation RMSE: 0.19
Validation R2 Score: 0.99
Validation MAE: 0.15
```

MODEL SELECTION & ACCURACY COMPARISON

🧪 **Models Tested:**

1. ❌ Linear Regression
   - ▸ Very simple, but low accuracy
   - ▸ R^2 Score: ~0.61
2. ❌ Decision Tree Regression
   - ▸ Showed overfitting on training data
   - ▸ Poor generalization

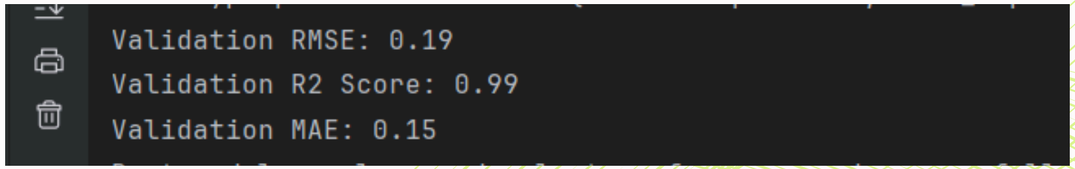
✅ **Final Model Chosen: Polynomial Regression**

- ✔ Captures non-linear patterns between environment and water needs
- ✔ Well-suited for agriculture sensor data
- ✔ Best accuracy and real-time performance

📈 **Final Validation Results:**

- 🔹 R^2 Score: 0.99
- 🔹 RMSE: 0.19
- 🔹 MAE: 0.15

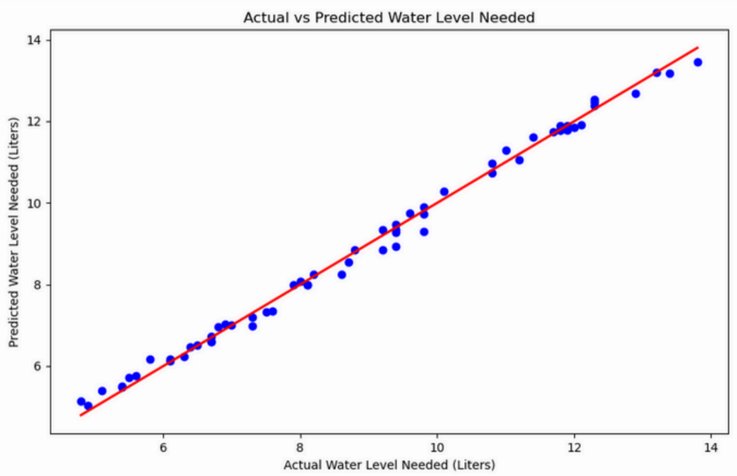
PERFORMANSE MATRIC ACCURECY :

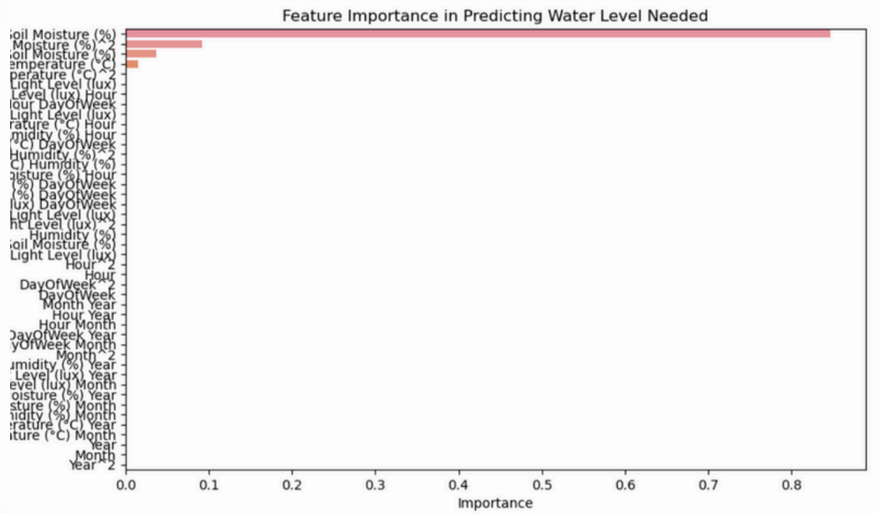


- **Polynomial Regression offered the perfect balance between accuracy, speed, and simplicity for real-time irrigation prediction.**

MODEL PERFORMANCE EVALUATION

```python
#plot the model Performance
plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_val_pred, color='blue')
plt.plot( *args: [min(y_val), max(y_val)], [min(y_val), max(y_val)], co
plt.title('Actual vs Predicted Water Level Needed')
plt.xlabel('Actual Water Level Needed (Liters)')
plt.ylabel('Predicted Water Level Needed (Liters)')
plt.show()

# Find the Important feature(x data) for water need prediction
feature_importances = best_model.feature_importances_
features_list = poly.get_feature_names_out(features)
importance_df = pd.DataFrame({
    'Feature': features_list,
    'Importance': feature_importances
})
importance_df = importance_df.sort_values(by='Importance', ascending=F

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importance in Predicting Water Level Needed')
plt.show()
```





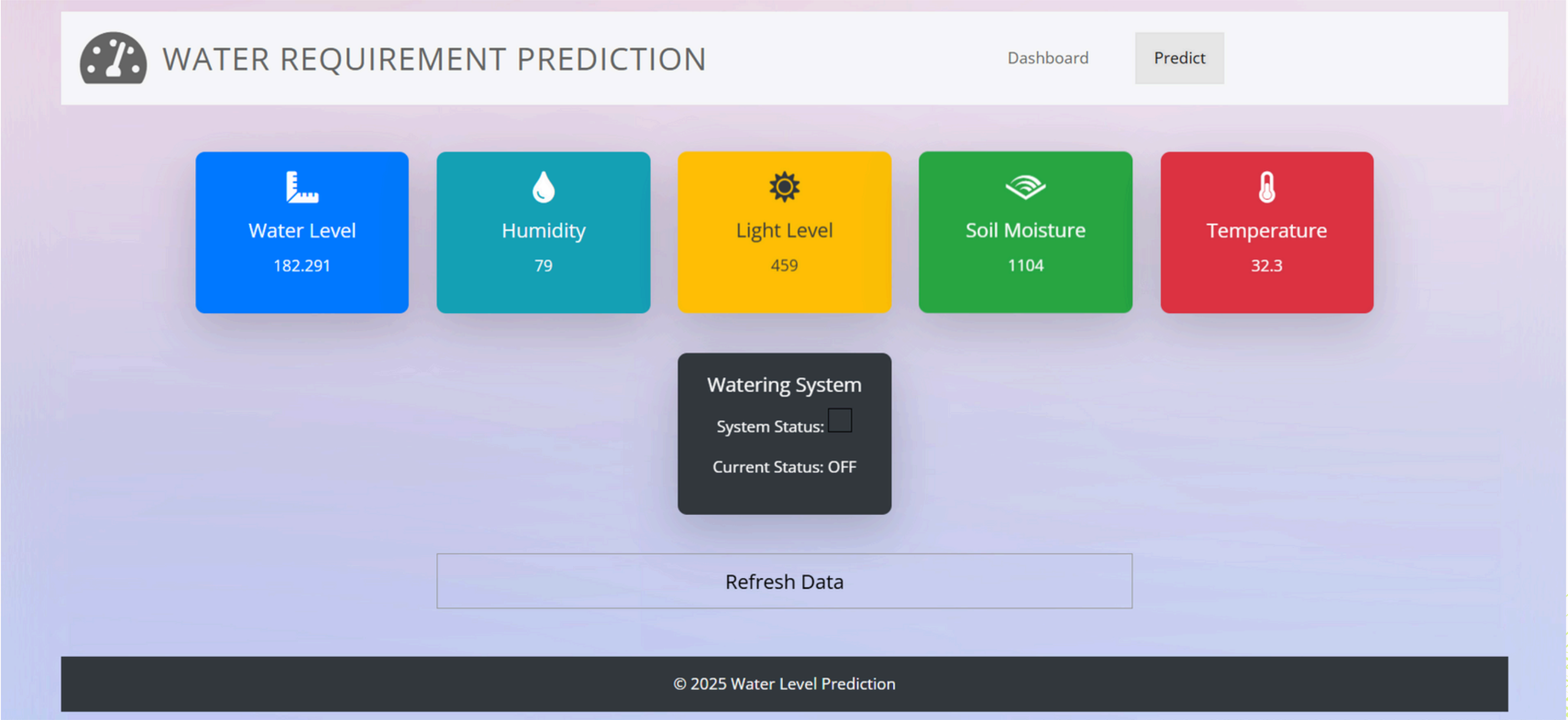

```python
#  predict watering needs
def predict_watering(temperature, humidity, soil_moisture, light_level, timestamp):  1 usage
    timestamp = pd.to_datetime(timestamp)
    hour = timestamp.hour
    day_of_week = timestamp.dayofweek
    month = timestamp.month
    year = timestamp.year

    # DataFrame input
    input_data = pd.DataFrame( data: [[temperature, humidity, soil_moisture, light_level, hour, day_of_week,
                               columns=['Temperature (°C)', 'Humidity (%)', 'Soil Moisture (%)', 'Light Leve

    # Normalize the input
    input_data_scaled = scaler.transform(input_data)

    # Apply polynomial
    input_data_poly = poly.transform(input_data_scaled)

    # prediction
    predicted_water_level = model.predict(input_data_poly)

    return predicted_water_level[0]
```

```python
    # prediction
    predicted_water_level = model.predict(input_data_poly)

    return predicted_water_level[0]

# data live
temperature = 25.5
humidity = 65.0
soil_moisture = 25.3
light_level = 1500
timestamp = "2024-11-01 10:00:00"


predicted_water_level = predict_watering(temperature, humidity, soil_moisture, ligl
print(f"Predicted Water Level Needed: {predicted_water_level:.2f} Liters")
```

```
Run  Preidct_watering
C:\Users\moham\Downloads\Compressed\Python38_01
C:\Users\moham\Downloads\Compressed\Python38_01
  warnings.warn(
Predicted Water Level Needed: 7.62 Liters

Process finished with exit code 0
```

REAL-TIME MONITORING DASHBOARD

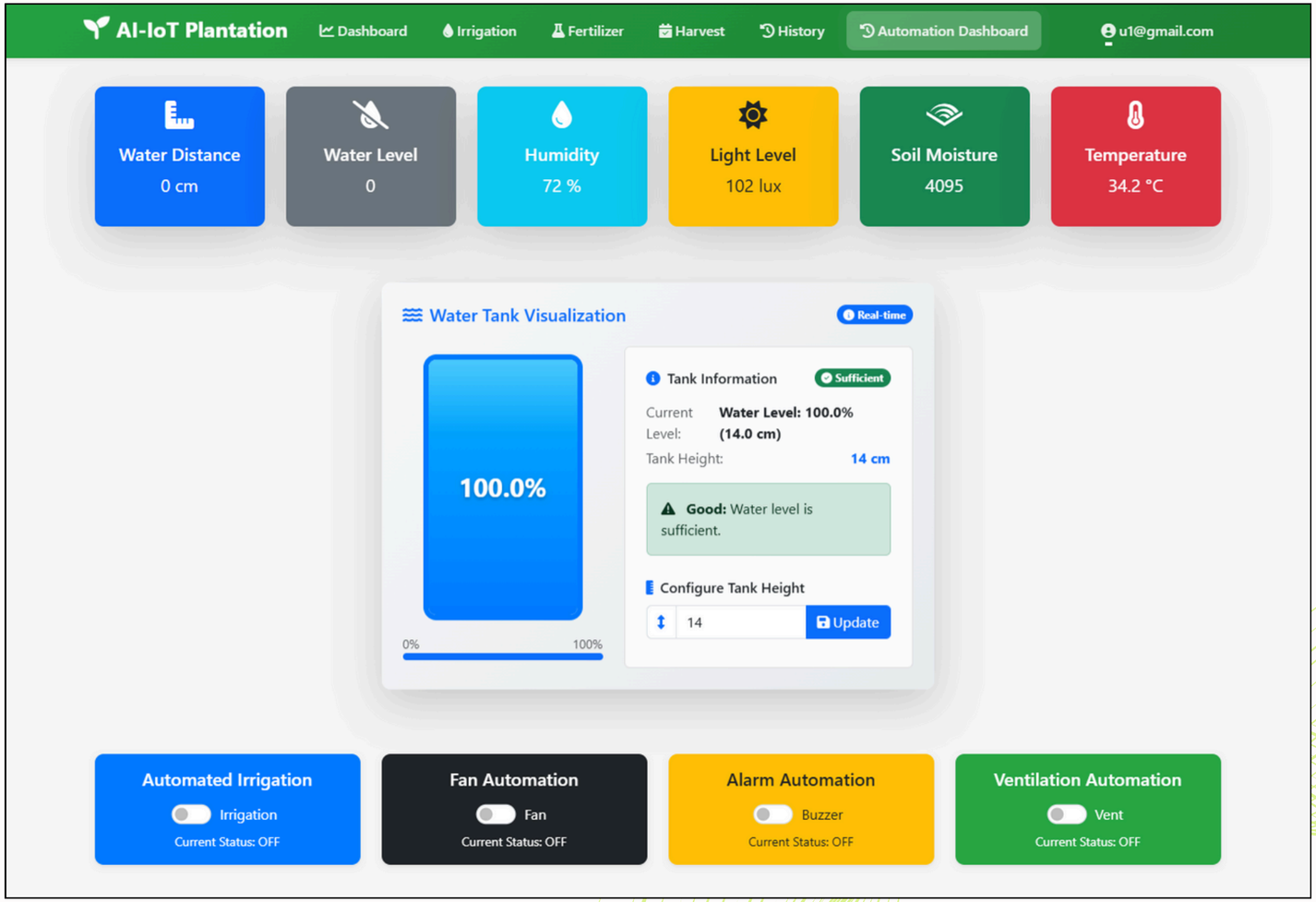
My previus version of the reading UI



REAL-TIME MONITORING & CONTROL DASHBOARD

- **Water Tank Visualization - configurable height** and level **alerts**
- Automation Controls for **irrigation, fan, vendilation** and **alarm** systems
- Live Alerts based on threshold conditions (e.g. low water, high heat)
- Real-Time Updates via Firebase

📈 **This UI Helps users monitor, control greenhouse conditions effectively.**



REAL-TIME AUTOMATION CONTROLS

Automated Irrigation Control

Problem in Existing Systems:

- Traditional irrigation systems rely only on **soil moisture** or **time-based triggers**, often leading to **over-irrigation** and **water wastage**.

Proposed Enhancement :

- The system automatically stops irrigation when **overflow is detected**, preventing **water wastage and root diseases** in tomato plants.

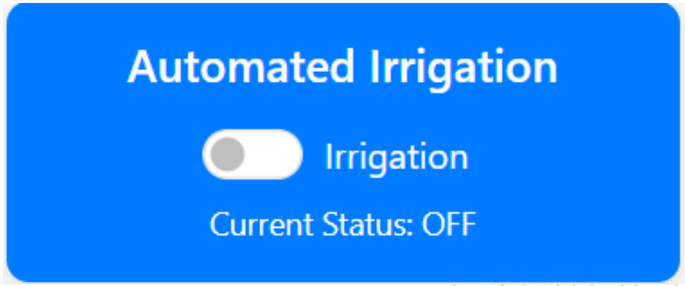
Trigger Conditions:

- When SoilMoisture > 2500 → **Soil moisture is dry**
- Tank > 10% → **Tank water availability**
- WaterLevel ≤ 1000 → **Water overflow detected**

Soil Moisture Sensor Value Range:
0 (wet) to ~4095 (dry)

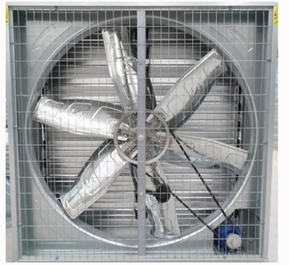
System Benefits:

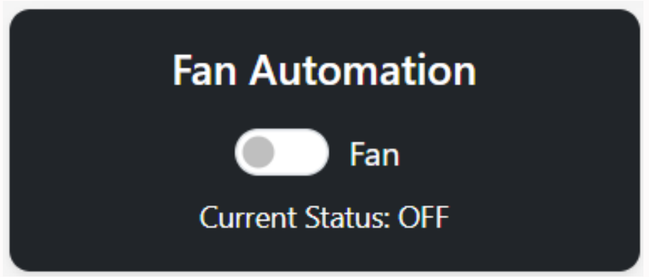
✓ Prevents overwatering & diseases
✓ Saves water & electricity



REAL-TIME AUTOMATION CONTROLS

Smart Climate Ventilation Control





Trigger Conditions:

- Humidity > 70 or
- Temperature > 35°C

Benefits:

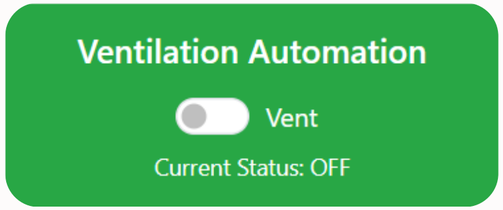
- Prevents plant diseases caused by Humidity (Ex: **fungal risk** - Botrytis, Powdery Mildew)
- Controal temperature for optimal plant health
- Prevents entry of **pests, dust, and airborne diseases**

REAL-TIME AUTOMATION CONTROLS

Smart Servo-Based Ventilation Control

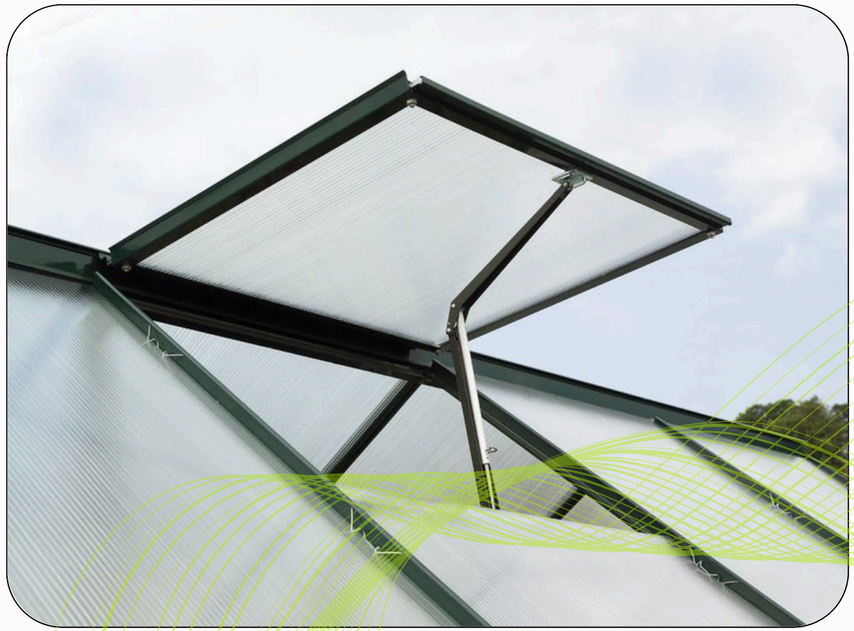
🔧 Trigger Condition:

- 🌡️ Temperature > 35°C or Humidity > 70 → **Ventilation windows automatically open**



🌱 **Benefits:**

- Reduces **heat stress** by enabling natural air circulation.
- Maintains ideal temperature for **photosynthesis** and growth.
- **Low energy** use – Minimizing energy use compared to fans.
- Best when **external conditions are safe** (**pest, birds, airborne diseases like fungal**)
- Improves plant **CO_2 exchange**



REAL-TIME AUTOMATION CONTROLS

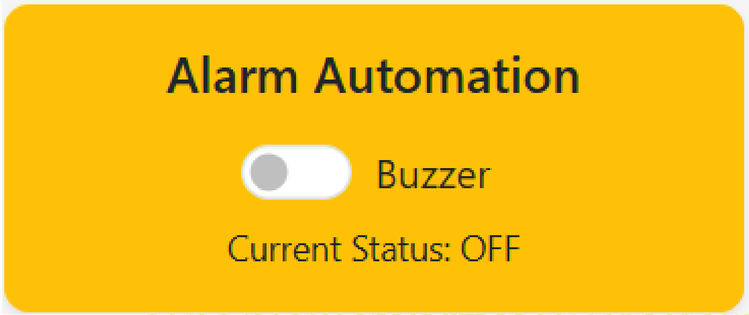
Critical Alert System – Buzzer Automation

Triggers (any one):

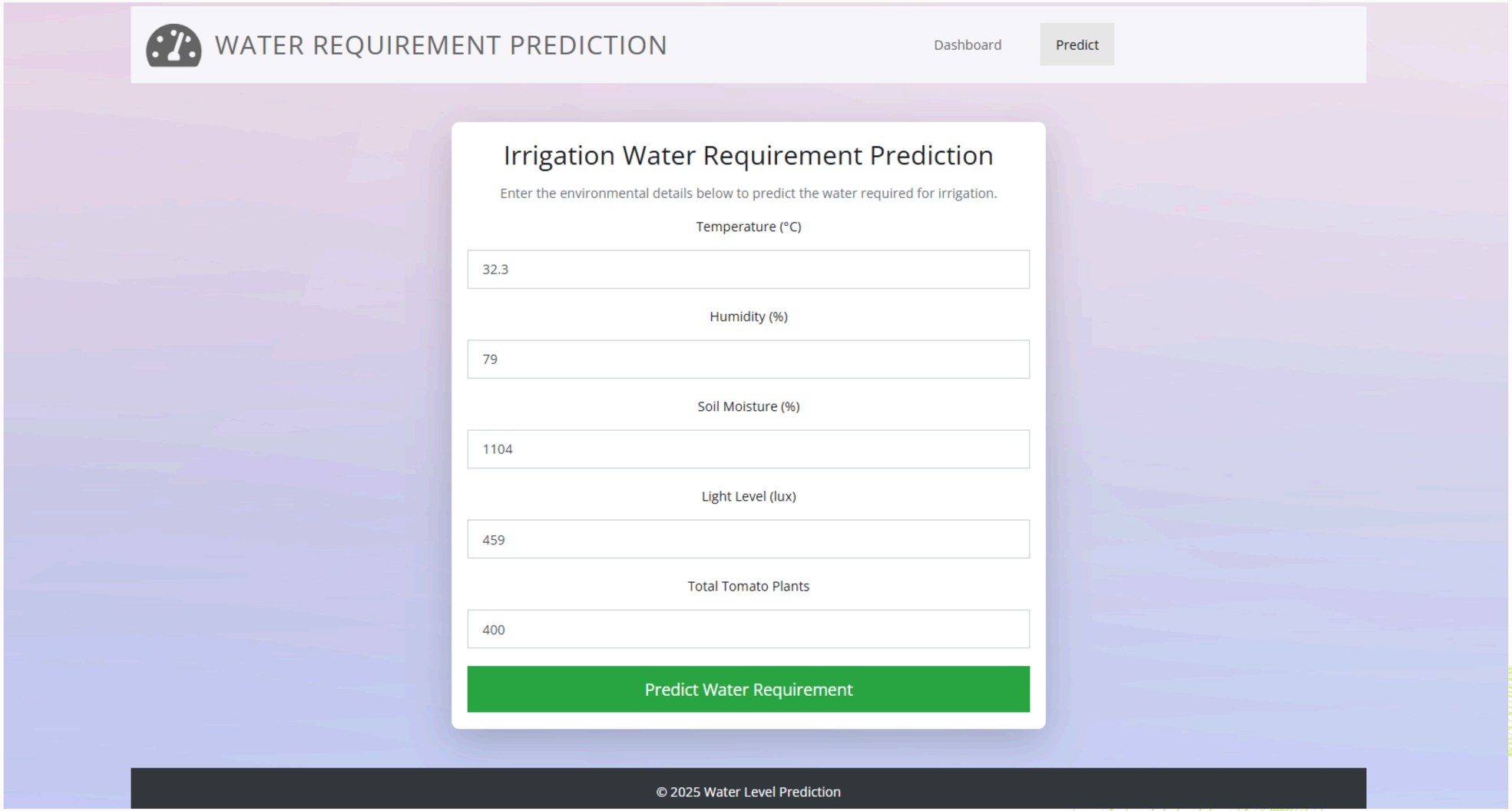
- Water Level > 1000 → **Water Overflow**
- Tank Level < 10% → **Water Availability is low**
- Temperature > 40°C → **Over Temperature**
- Soil Moisture > 2500 → **When the Soil Moisture dry**

Benefits of this Automation System:

- Immediate **physical alert** to critical risks
- Reduces reliance on dashboard monitoring
- **Failsafe safety Feature** if web access or internet is down
- Fully automated



WATER REQUIREMENT PREDICTION UI



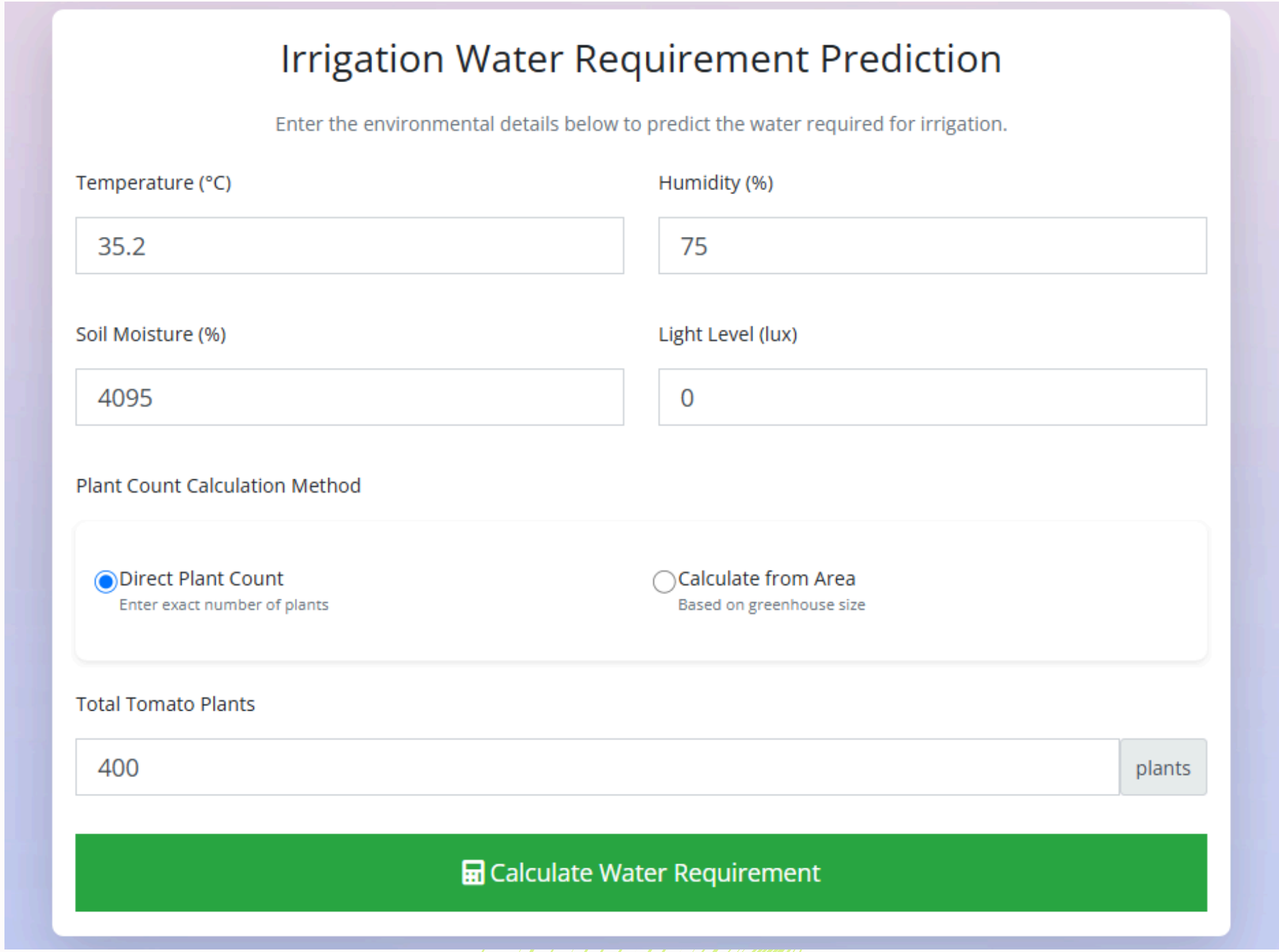
WATER REQUIREMENT PREDICTION UI

🧾 User Inputs:

- Temperature (°C), Humidity (%), Soil Moisture (%), Light Level (lux)
- Plant Count: Can input in two methods
  - Direct input the Plant Counts
  - Auto-calculate from greenhouse area & spacing

📈 Outputs:

- ✅ Per Plant Water Need (L)
- ✅ Total Plant Count (if 2nd method selected)
- ✅ Total Water Requirement (L)
- 🧠 Prediction Reason (e.g., high temperature, dry soil)

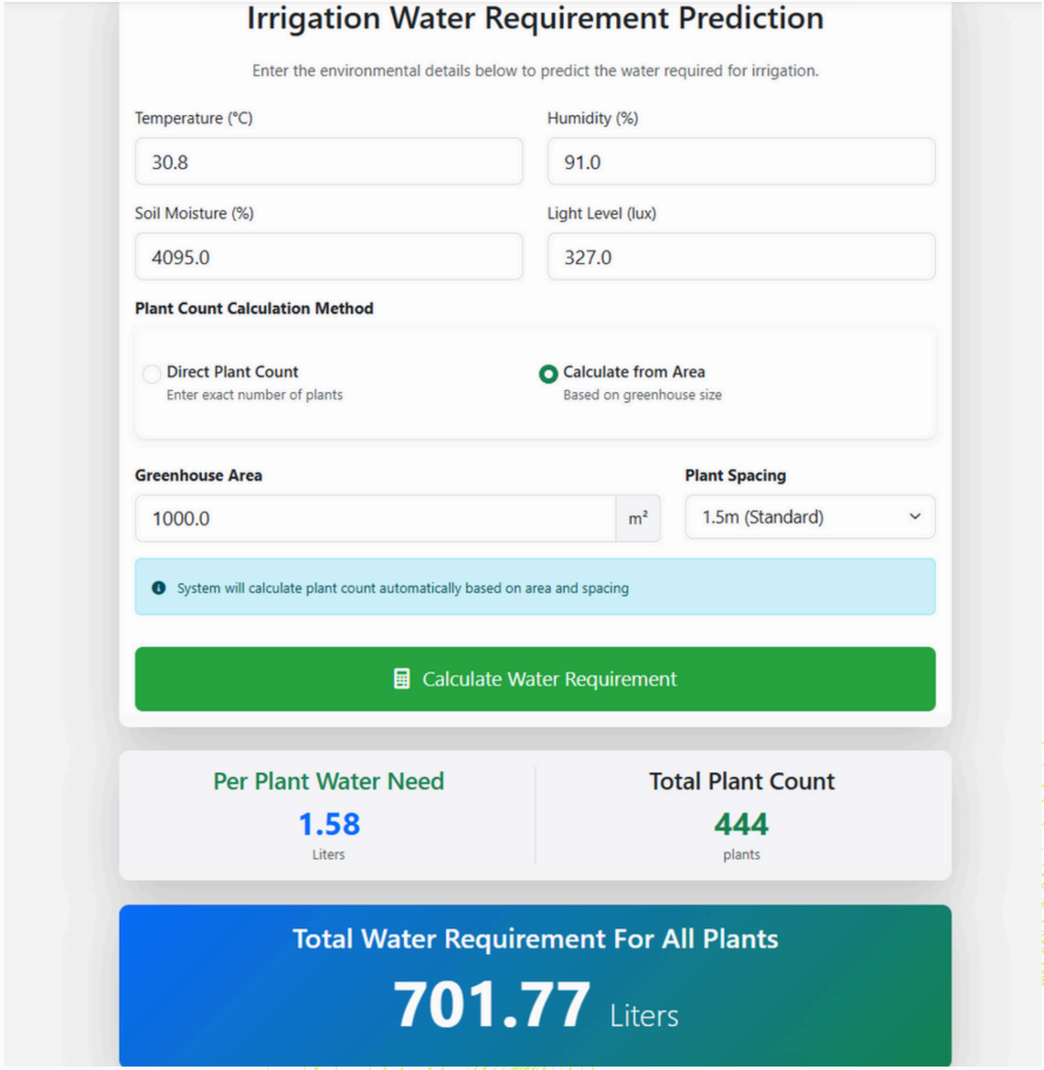


WATER REQUIREMENT PREDICTION UI

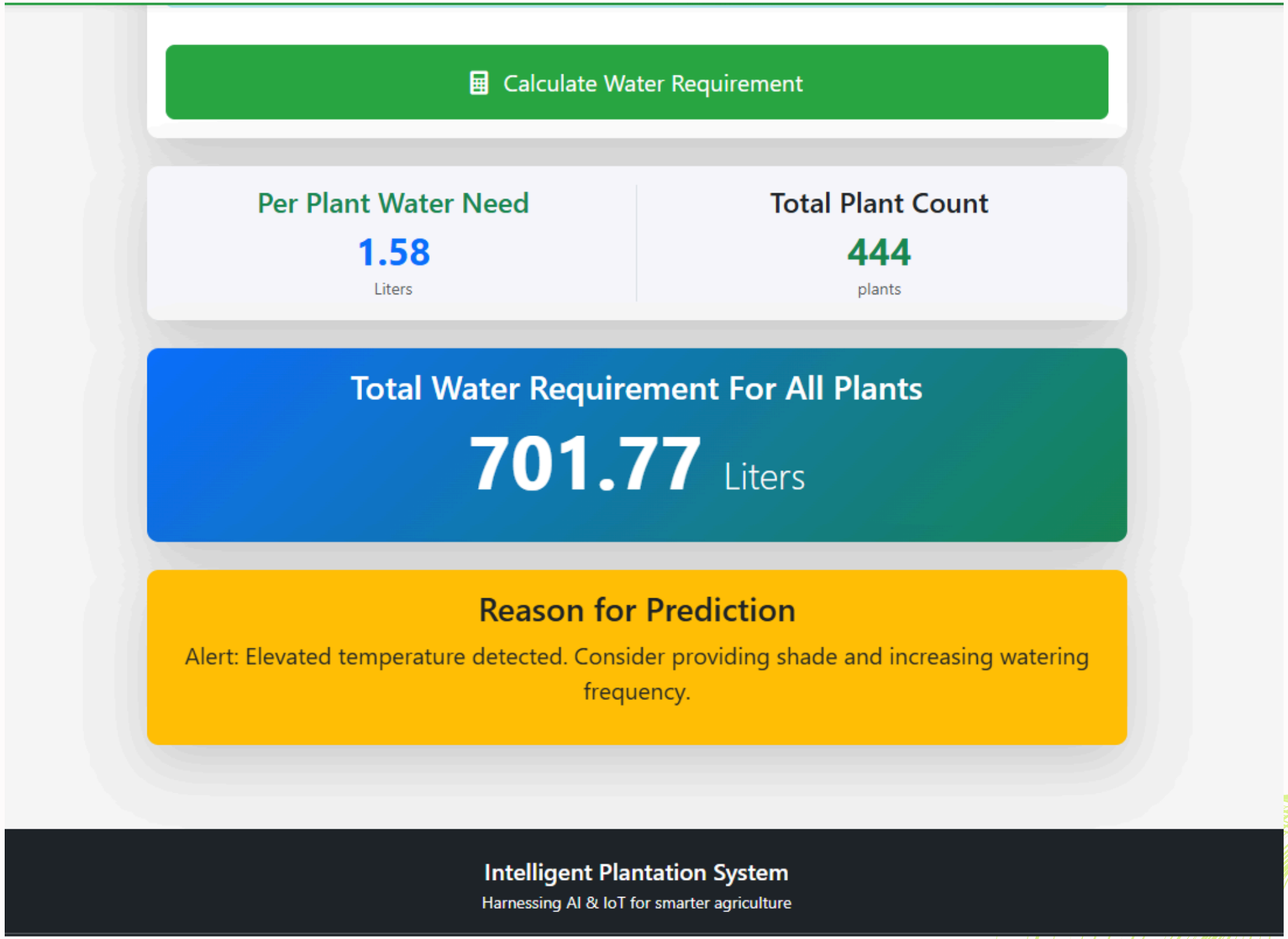
Introduced a new method called ‘Calculation From Area’ for total plant count calculation based on the greenhouse’s area (square feet) and plant gap.

sample output:

if area=1000 and the Gap=1.5 → plant counts=444



WATER REQUIREMENT PREDICTION OUTPUT



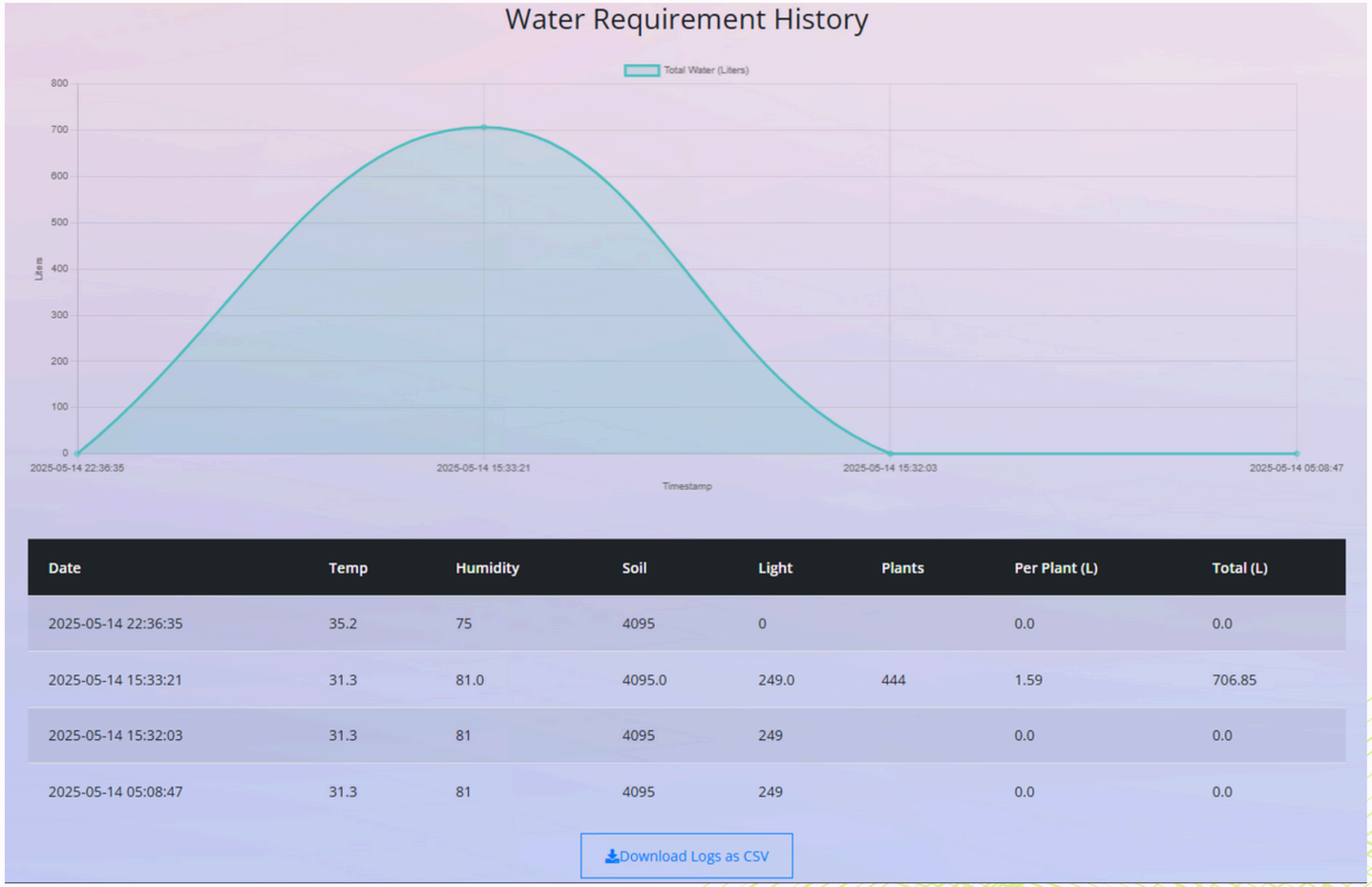
WATER REQUIREMENT HISTORY DASHBOARD UI

💧 **Water Requirement History UI**

- 📅 Logs each prediction with environmental inputs() & water output
- 📈 Line chart shows total water usage trends over time
- 📋 Table view for detailed record inspection
- 📥 **Download as CSV** for offline tracking

✅ **Benefits:**

- **Tracks irrigation performance**
- **Supports data-driven decisions**
- **Aids in optimizing water usage**



REFERENCES

[1] Tao, W., Zhao, L., Wang, G. & Liang, R. (2021). Review of the internet of things communicationtechnologies in smartagriculture and challenges, Computers and Electronics in Agriculture, Vol. 189, 2021, 106352, ISSN0168-1699, https://doi.org/10.1016/j.compag.2021.106352.

[2] Food and AgricultureOrganization of theUnited Nations, Global agriculture towards2050, https://www.fao.org/fileadmin/templates/wsfs/docs/Issues_papers/HLEF2050_Global_Agriculture.pdf.

[3] Saggi, M.K. & Jain, S. A. (2022). Survey Towards Decision Support System on SmartIrrigation Scheduling UsingMachine Learning approaches. Archives of Computational Methods inEngineering, 29, 4455–4478. https://doi.org/10.1007/s11831-022-09746-3.

[4] Salim, O., Fouad, K. & Hassan, B. (2022). Dual-Level Sensor Selection with Adaptive Sensor Recovery to Extend WSNs' Lifetime, Human-centric Computing and Information Sciences, Vol. 12, Article number: 18, https://doi.org/10.22967/HCIS.2022.12.018.

[5] Goap, A.,Sharma, D., Shukla,A.K. & Krishna,R. (2018). An IoT based smart irrigation management system using Machine learning and open source technologies, Computers and Electronics in Agriculture, Vol. 155,PP: 41-49, ISSN0168-1699, https://doi.org/10.1016/j.compag.2018.09.040.

[6] Fouad, K. & Elbably, D. (2020).Intelligent approachfor large-scale data mining.Int. J. Computer Applications inTechnology, Vol. 63, Nos. 1/2, PP: 93-112.

[7] Fouad, K., Hassan, B. & Salim, O. (2022). Hybrid Sensor Selection Technique for LifetimeExtension of WirelessSensor Networks. Computers, Materials& Continua 2022,70(3), 4965-4985. https://doi.org/10.32604/cmc.2022.020926.

REFERENCES

[1] Tao, W., Zhao, L., Wang, G. & Liang, R. (2021). Review of the internet of things communicationtechnologies in smartagriculture and challenges, Computers and Electronics in Agriculture, Vol. 189, 2021, 106352, ISSN0168-1699, https://doi.org/10.1016/j.compag.2021.106352.

[2] Food and AgricultureOrganization of theUnited Nations, Global agriculture towards2050, https://www.fao.org/fileadmin/templates/wsfs/docs/Issues_papers/HLEF2050_Global_Agriculture.pdf.

[3] Saggi, M.K. & Jain, S. A. (2022). Survey Towards Decision Support System on SmartIrrigation Scheduling UsingMachine Learning approaches. Archives of Computational Methods inEngineering, 29, 4455–4478. https://doi.org/10.1007/s11831-022-09746-3.

[4] Salim, O., Fouad, K. & Hassan, B. (2022). Dual-Level Sensor Selection with Adaptive Sensor Recovery to Extend WSNs' Lifetime, Human-centric Computing and Information Sciences, Vol. 12, Article number: 18, https://doi.org/10.22967/HCIS.2022.12.018.

[5] Goap, A.,Sharma, D., Shukla,A.K. & Krishna,R. (2018). An IoT based smart irrigation management system using Machine learning and open source technologies, Computers and Electronics in Agriculture, Vol. 155,PP: 41-49, ISSN0168-1699, https://doi.org/10.1016/j.compag.2018.09.040.

[6] Fouad, K. & Elbably, D. (2020).Intelligent approachfor large-scale data mining.Int. J. Computer Applications inTechnology, Vol. 63, Nos. 1/2, PP: 93-112.

[7] Fouad, K., Hassan, B. & Salim, O. (2022). Hybrid Sensor Selection Technique for LifetimeExtension of WirelessSensor Networks. Computers, Materials& Continua 2022,70(3), 4965-4985. https://doi.org/10.32604/cmc.2022.020926.

REFERENCES

[1] Tao, W., Zhao, L., Wang, G. & Liang, R. (2021). Review of the internet of things communicationtechnologies in smartagriculture and challenges, Computers and Electronics in Agriculture, Vol. 189, 2021, 106352, ISSN0168-1699, https://doi.org/10.1016/j.compag.2021.106352.

[2] Food and AgricultureOrganization of theUnited Nations, Global agriculture towards2050, https://www.fao.org/fileadmin/templates/wsfs/docs/Issues_papers/HLEF2050_Global_Agriculture.pdf.

[3] Saggi, M.K. & Jain, S. A. (2022). Survey Towards Decision Support System on SmartIrrigation Scheduling UsingMachine Learning approaches. Archives of Computational Methods inEngineering, 29, 4455–4478. https://doi.org/10.1007/s11831-022-09746-3.

[4] Salim, O., Fouad, K. & Hassan, B. (2022). Dual-Level Sensor Selection with Adaptive Sensor Recovery to Extend WSNs' Lifetime, Human-centric Computing and Information Sciences, Vol. 12, Article number: 18, https://doi.org/10.22967/HCIS.2022.12.018.

[5] Goap, A.,Sharma, D., Shukla,A.K. & Krishna,R. (2018). An IoT based smart irrigation management system using Machine learning and open source technologies, Computers and Electronics in Agriculture, Vol. 155,PP: 41-49, ISSN0168-1699, https://doi.org/10.1016/j.compag.2018.09.040.

[6] Fouad, K. & Elbably, D. (2020).Intelligent approachfor large-scale data mining.Int. J. Computer Applications inTechnology, Vol. 63, Nos. 1/2, PP: 93-112.

[7] Fouad, K., Hassan, B. & Salim, O. (2022). Hybrid Sensor Selection Technique for LifetimeExtension of WirelessSensor Networks. Computers, Materials& Continua 2022,70(3), 4965-4985. https://doi.org/10.32604/cmc.2022.020926.

IT21181160 | Thrimavithana V. D.

Information technology

INTRODUCTION TO PREDICTING FERTILIZING SCHEDULES

Tomatoes are a vital crop, but their cultivation is challenged by climate change, which affects soil nutrients. Proper fertilization is essential to ensure healthy growth, but mismanagement can harm plants and the environment.

KEY CHALLENGES

- **Over-Fertilization:** Causes nutrient leaching, pollution, and plant toxicity.
- **Under-Fertilization:** Leads to poor growth and low yields.

ENVIRONMENTAL FACTORS

- **High temperatures** increase nutrient demand but risk imbalances.
- **Low temperatures** slow nutrient uptake, risking deficiencies.

RESEARCH QUESTION

How can we develop a decision support system for tomato greenhouses that optimizes fertilizing schedules and types that ensures nutrient balance based on real-time plant and environmental conditions?

RESEARCH GAP

Existing Products / Research	Features						
	Automated Fertilizer Application	Real-Time Nutrient Monitoring	Precision Nutrient Delivery	Environmentally Sustainable Practices	Integration with IoT and Sensors	Impact on Fruit Quality	Predictive Analytics for Nutrient Needs
IoT-Based Precision Fertilization System	✓	✗	✓	✗	✓	✗	✗
Smart Fertigation Management System	✓	✓	✗	✗	✗	✓	✗
Integrated Hydroponic Nutrient Solution	✗	✓	✓	✓	✗	✗	✓
Proposed System	✓	✓	✓	✓	✓	✓	✓

METHODOLOGY

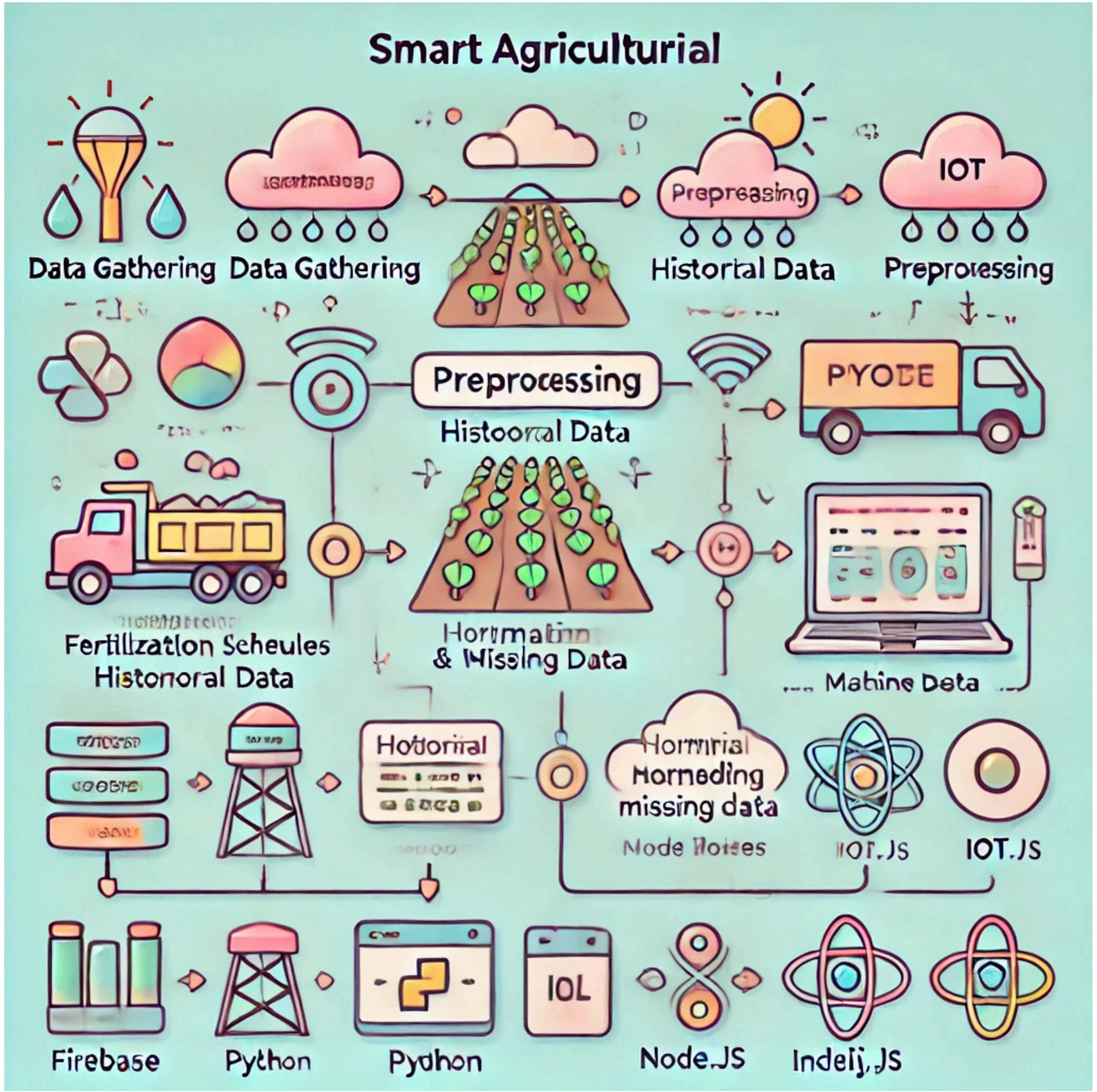
1.Gather soil, nutrient data and environmental conditions.

2.Collect historical fertilization schedules and crop yield data.

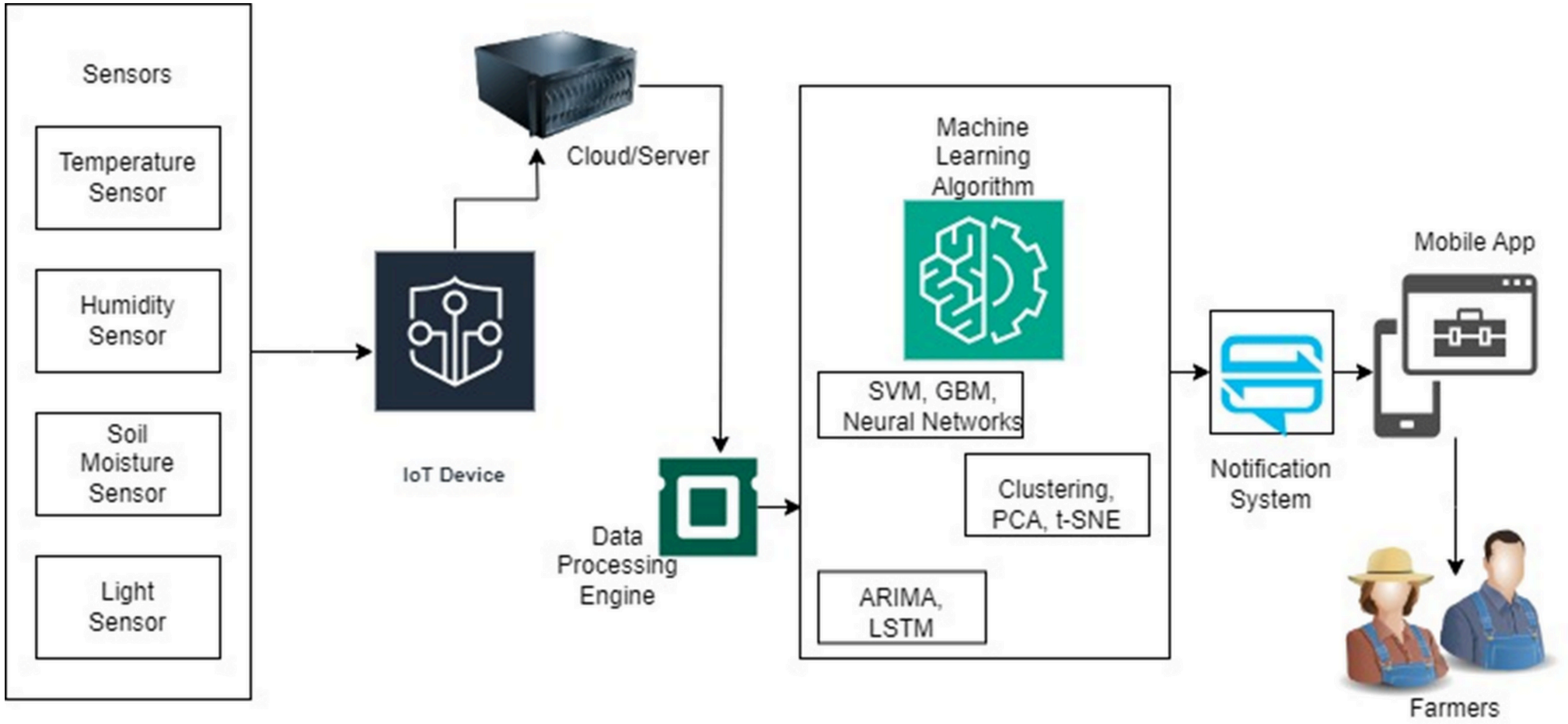
3.Preprocess the data to handle missing values and normalize it.

4.Choose appropriate hybrid ML models.

5.Train the models using the preprocessed data to optimize fertilization schedules.



SYSTEM DIAGRAM



DATASET

	A	B	C	D	E	F	G	H
1	Organic Matter	Humidity (%)	Temperature (ツ-C)	Fertilizer Type	Fertilizer Quantity (kg/ha)	Fertilizer Timing (Week)	Crop Yield (tons/ha)	Historical Yield (tons/ha)
2	2.749080238	75.83158087	20.13094457	Potash	180.4636872	8	4.675482342	7.215243009
3	3.901428613	75.79236286	26.63537372	Ammonium Nitrate	176.9410612	7	8.222115865	4.788313252
4	3.463987884	61.82412206	21.78035967	DAP	218.7758391	11	7.24419086	3.632189534
5	3.197316968	69.88840609	29.61070317	Potash	97.97520052	6	4.579206821	7.69094318
6	2.312037281	61.1511752	21.48662728	Urea	94.57535091	7	5.45136297	6.864895329
7	2.311989041	70.99057765	24.14624124	Potash	181.4074344	5	4.074424218	3.289116074
8	2.116167224	68.83061003	20.85349668	Ammonium Nitrate	160.9741684	5	6.812607793	5.160302795
9	3.732352292	77.75408366	29.96874252	Ammonium Nitrate	118.3824933	6	5.807591565	8.740798893
10	3.202230023	67.01830025	25.0219501	DAP	91.11871166	4	7.590144906	6.002406874
11	3.416145156	62.34134033	25.95385017	Potash	91.99215494	8	5.783426846	5.596374816
12	2.041168989	62.85983364	20.67076477	Urea	205.1867225	7	5.799514669	5.746197631
13	3.939819704	75.23021263	27.4996047	Urea	106.8614255	5	8.459159192	4.253296344
14	3.664885282	72.36436127	22.09905593	NPK	125.2720187	4	4.288857016	5.212243025
15	2.424678221	62.02245352	28.98054289	Urea	111.7318963	6	9.417369971	5.218879536
16	2.363649934	61.68213612	22.0513964	DAP	129.6994828	10	9.113583246	3.314130121
17	2.36680902	74.01938263	21.90687721	DAP	89.71933798	9	8.006828099	7.605405771
18	2.608484486	61.45526013	20.36549668	Urea	152.6683707	10	7.559329421	5.499026591
19	3.049512863	76.43720119	24.72066945	Urea	89.4657589	10	9.353814989	7.933079321
20	2.863890037	74.12484454	25.64841133	NPK	192.0499111	5	5.111981077	8.102088007
21	2.58245828	61.62697561	20.65708639	Urea	112.7196915	11	4.473814973	4.271964328
22	3.223705789	61.69675428	27.75527617	NPK	155.601668	11	5.437061019	6.944120893
23	2.278987721	79.73279157	24.53288835	NPK	203.2110723	6	8.767469672	5.833690825
24	2.584289297	67.48541592	25.24390269	Urea	171.1228317	7	4.2080216	8.280940428
25	2.732723687	67.41284294	24.40762747	Ammonium Nitrate	154.6140901	11	7.496841011	4.294444075
26	2.912139968	76.25599135	24.00763061	Ammonium Nitrate	125.4067254	10	9.972625097	7.066879202
27	3.570351923	78.94497155	25.59640331	NPK	126.6202678	11	9.134176563	6.646512734
28	2.399347564	79.72002128	21.55240246	Urea	173.7281737	10	7.128674134	4.771808865

fertilization_data

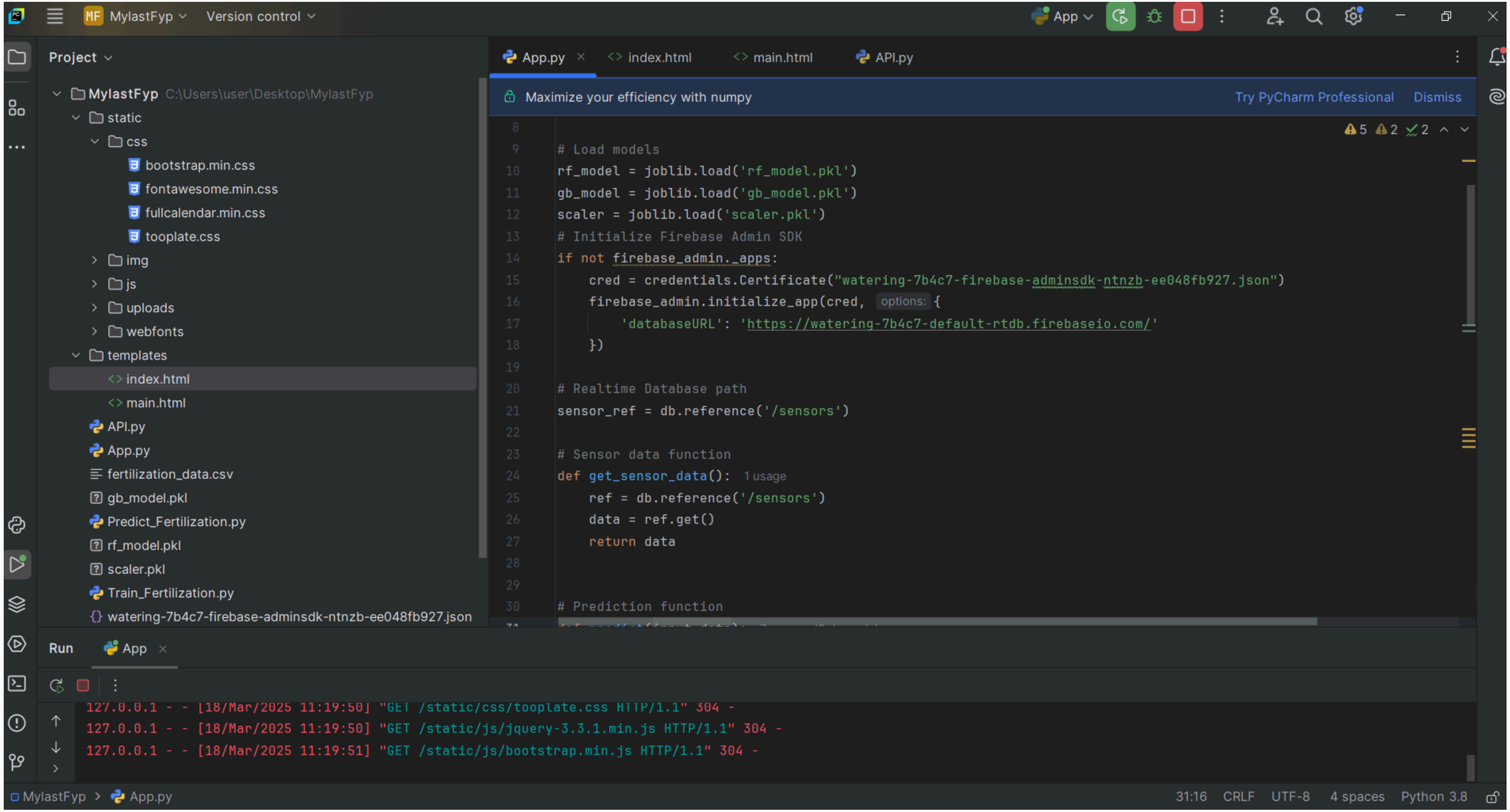
DATA PREPROCESSING

```
# Load the dataset
file_path = "fertilization_data.csv"
df = pd.read_csv(file_path)

# Preprocess
df = pd.get_dummies(df, columns=['Fertilizer Type'], drop_first=True)

# Split data
X = df.drop(['Fertilizer Quantity (kg/ha)', 'Fertilizer Timing (Week)'], axis=1)  # Features
y = df[['Fertilizer Quantity (kg/ha)', 'Fertilizer Timing (Week)']]  # Targets

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

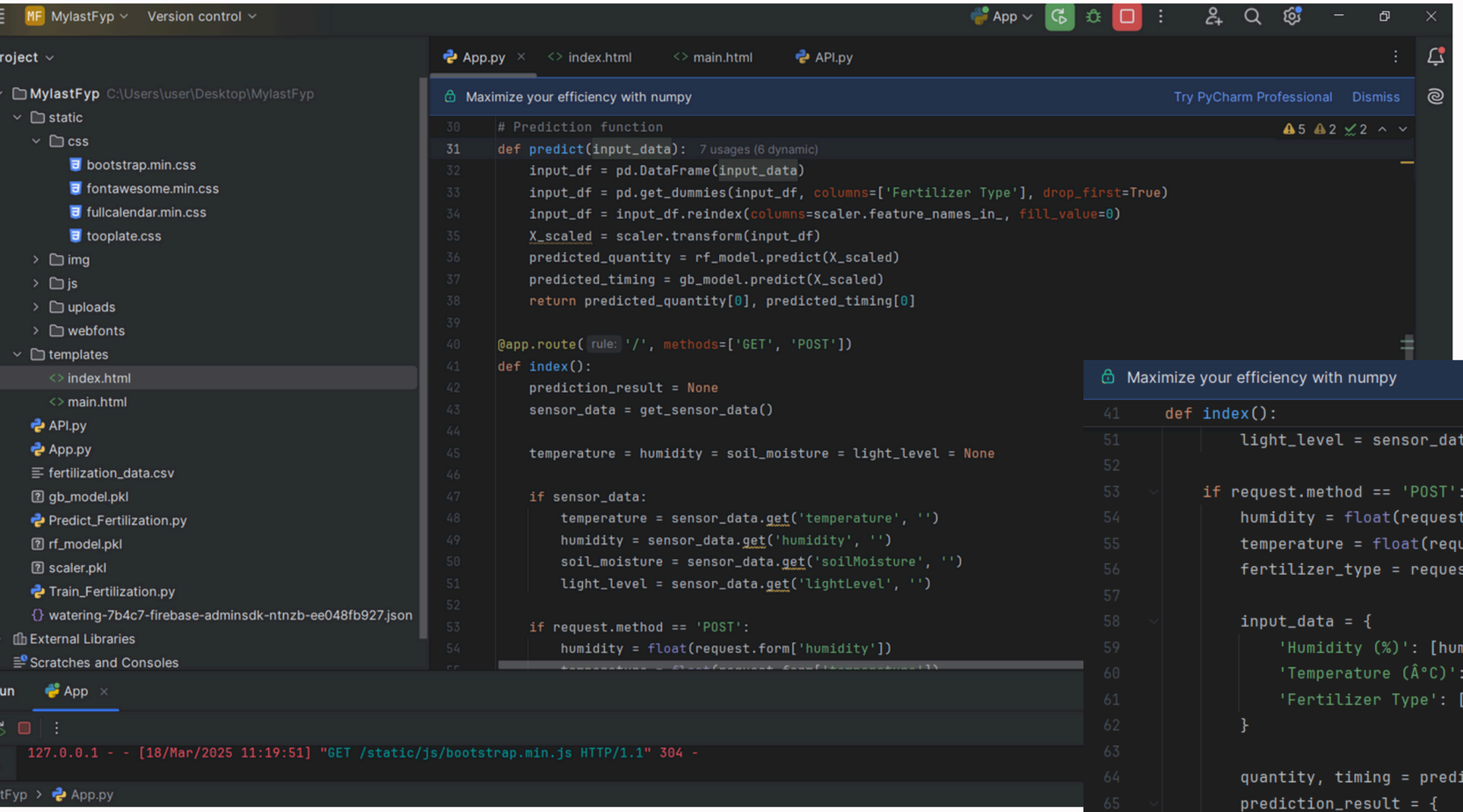
MODEL TRAINING

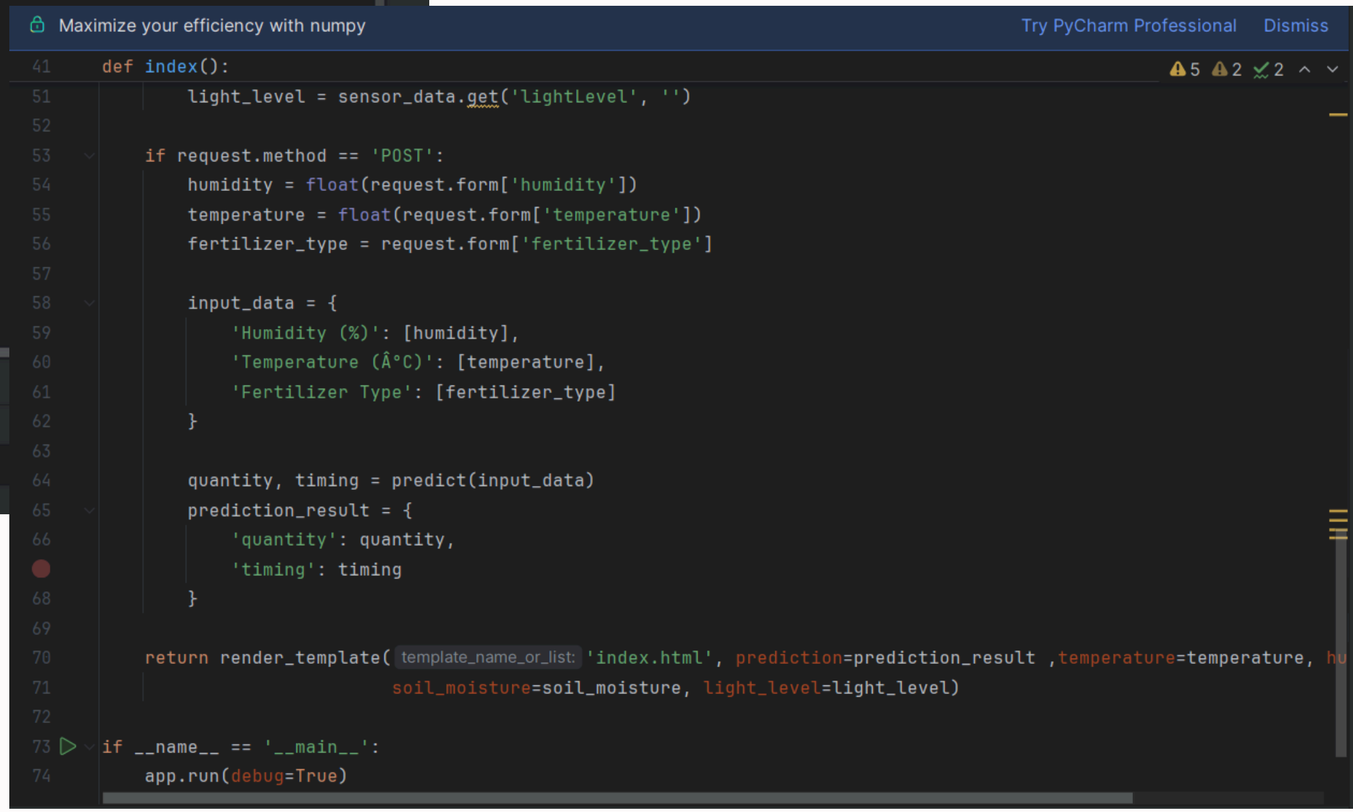
```
# Train the models
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train['Fertilizer Quantity (kg/ha)'])
gb_model.fit(X_train_scaled, y_train['Fertilizer Timing (Week)'])
rf_predictions = rf_model.predict(X_test_scaled)
gb_predictions = gb_model.predict(X_test_scaled)

# Evaluate
rf_mae = mean_absolute_error(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)
rf_mse = mean_squared_error(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)
rf_r2 = r2_score(y_test['Fertilizer Quantity (kg/ha)'], rf_predictions)

gb_mae = mean_absolute_error(y_test['Fertilizer Timing (Week)'], gb_predictions)
gb_mse = mean_squared_error(y_test['Fertilizer Timing (Week)'], gb_predictions)
gb_r2 = r2_score(y_test['Fertilizer Timing (Week)'], gb_predictions)
rf_cv_scores = cross_val_score(rf_model, X_train_scaled, y_train['Fertilizer Quantity (kg/ha)'], cv=5, scoring='neg_mean_absolute_
gb_cv_scores = cross_val_score(gb_model, X_train_scaled, y_train['Fertilizer Timing (Week)'], cv=5, scoring='neg_mean_absolute_err

```

SENSOR DATA - FUNCTION

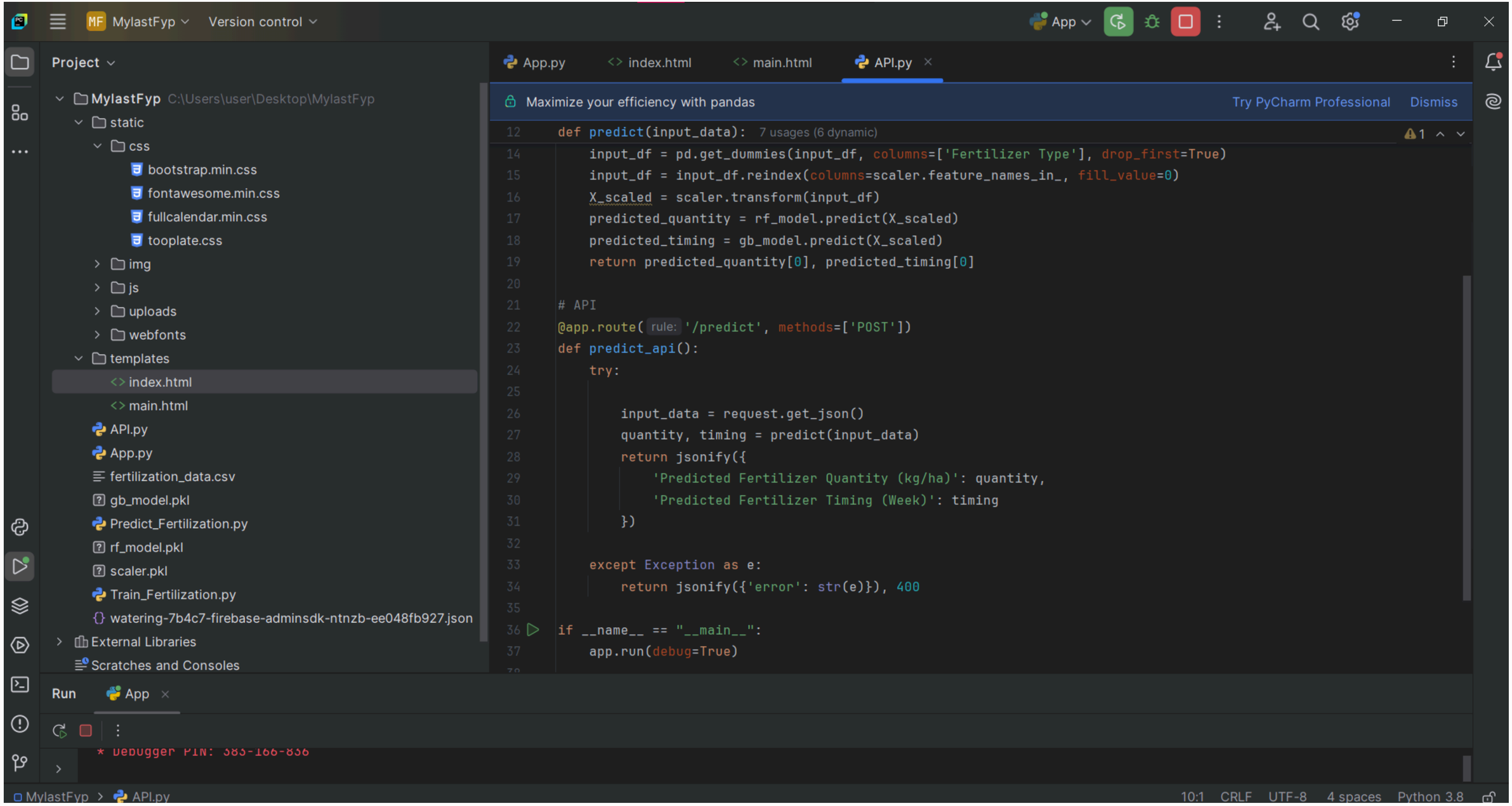


PREDICTION FUNCTION

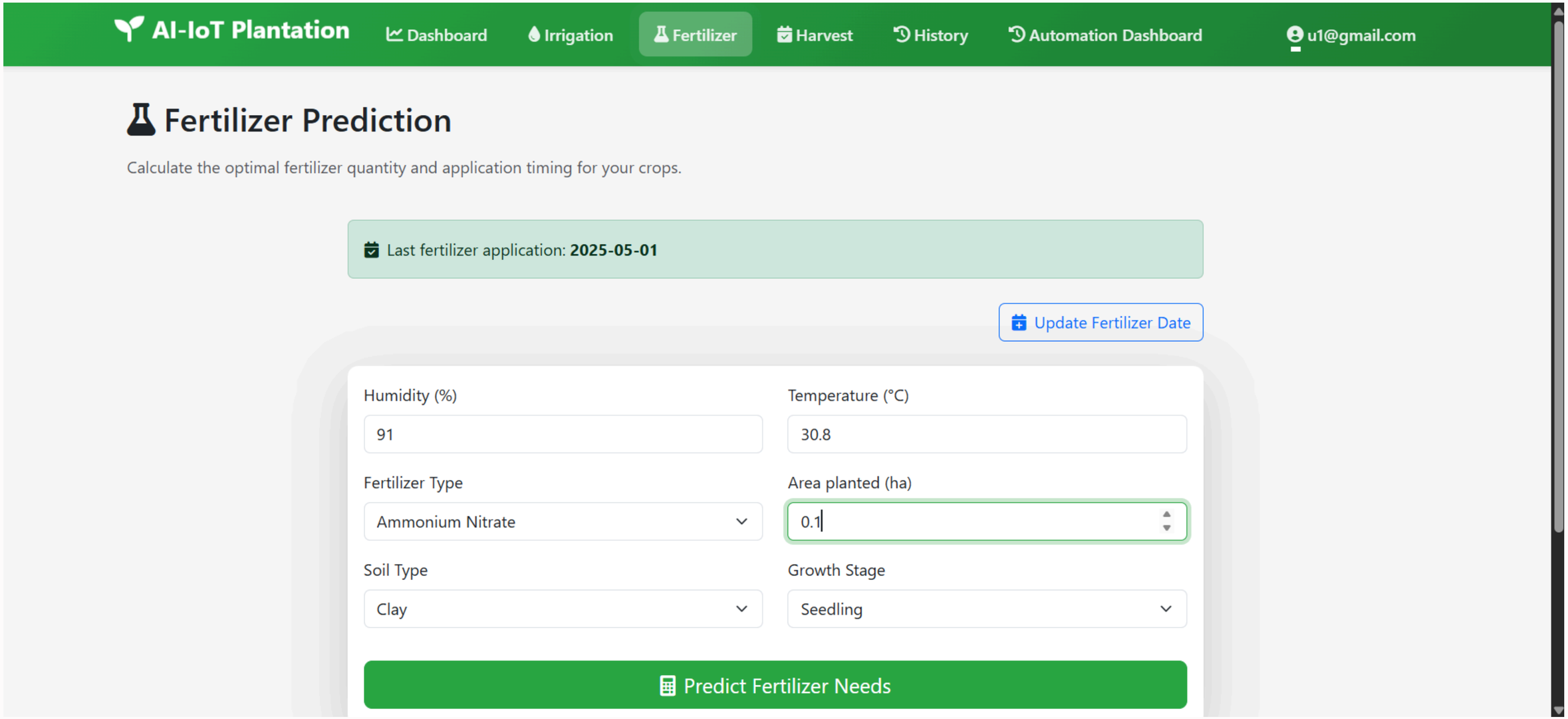




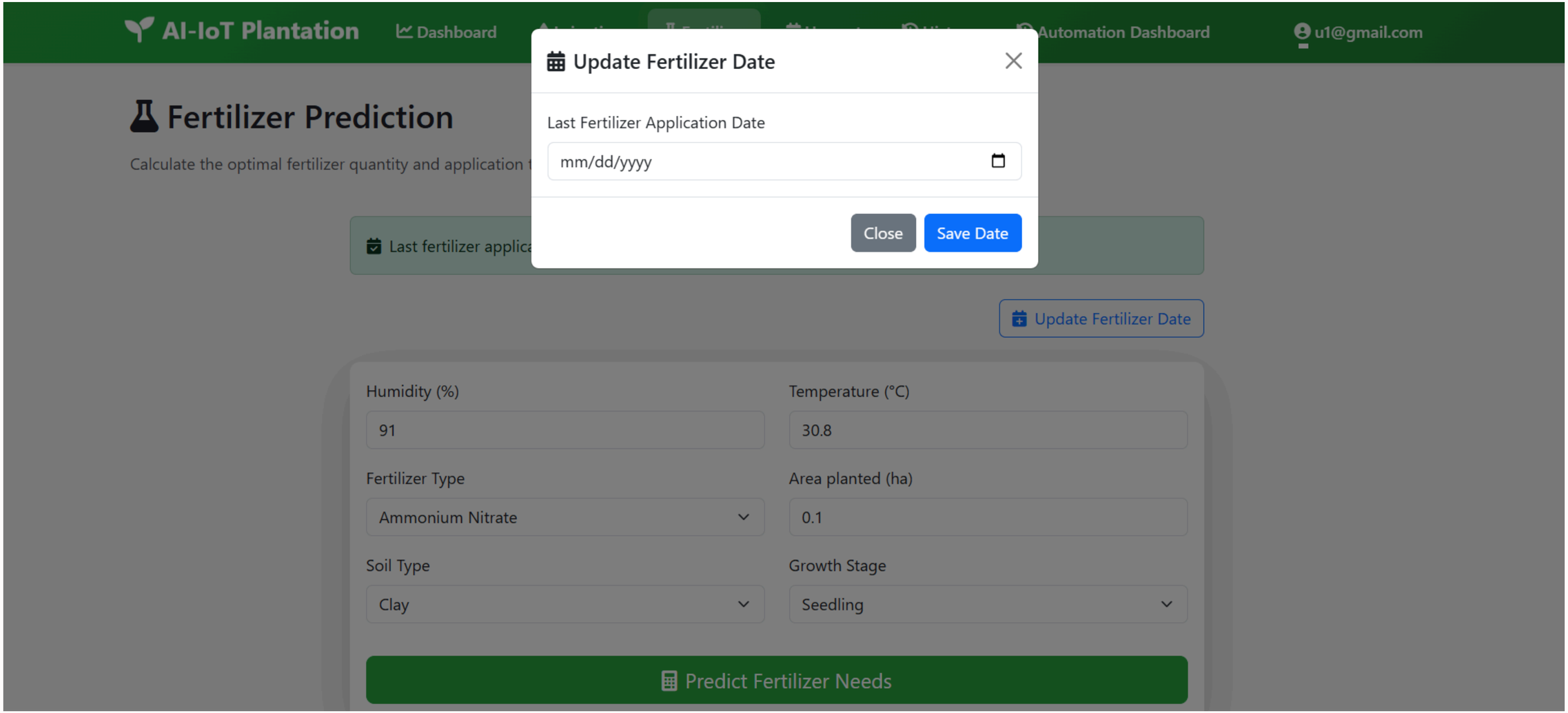
API S


MylastFyp
Version control
App
Project
MylastFyp C:\Users\user\Desktop\MylastFyp
static
css
bootstrap.min.css
fontawesome.min.css
fullcalendar.min.css
tooplate.css
img
js
uploads
webfonts
templates
index.html
main.html
API.py
App.py
fertilization_data.csv
gb_model.pkl
Predict_Fertilization.py
rf_model.pkl
scaler.pkl
Train_Fertilization.py
watering-7b4c7-firebase-adminsdk-ntnzb-ee048fb927.json
External Libraries
Scratches and Consoles
App.py
index.html
main.html
API.py
Maximize your efficiency with pandas
Try PyCharm Professional
Dismiss
def predict(input_data): 7 usages (6 dynamic)
input_df = pd.get_dummies(input_df, columns=['Fertilizer Type'], drop_first=True)
input_df = input_df.reindex(columns=scaler.feature_names_in_, fill_value=0)
X_scaled = scaler.transform(input_df)
predicted_quantity = rf_model.predict(X_scaled)
predicted_timing = gb_model.predict(X_scaled)
return predicted_quantity[0], predicted_timing[0]
API
@app.route(rule: '/predict', methods=['POST'])
def predict_api():
try:
input_data = request.get_json()
quantity, timing = predict(input_data)
return jsonify({
'Predicted Fertilizer Quantity (kg/ha)': quantity,
'Predicted Fertilizer Timing (Week)': timing
})
except Exception as e:
return jsonify({'error': str(e)}), 400
if __name__ == "__main__":
app.run(debug=True)
Run
App
* Debugger PIN: 383-166-836
MylastFyp
API.py
10:1
CRLF
UTF-8
4 spaces
Python 3.8

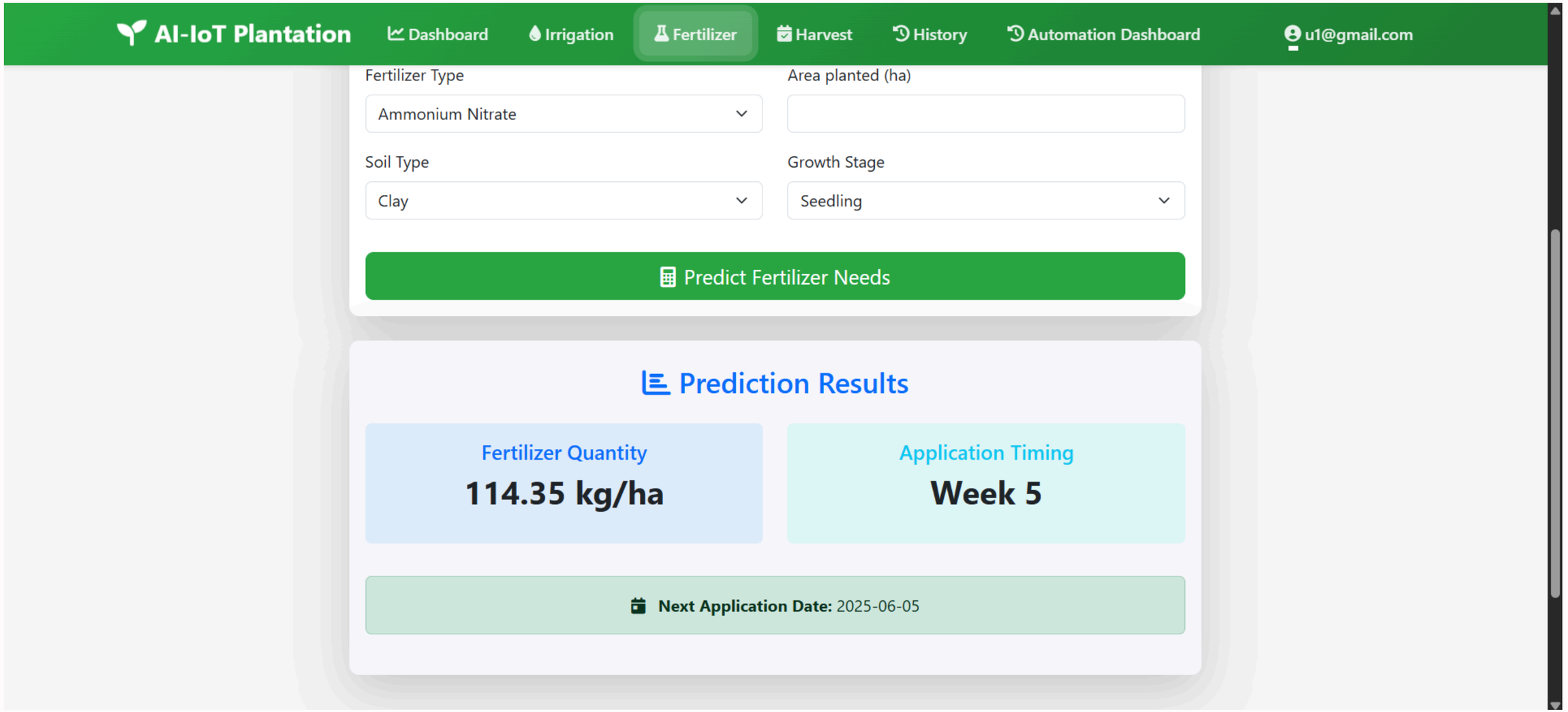
USER INTERFACE



USER INTERFACE



USER INTERFACE



REFERENCES

[1] "Big Data Analytics for Smart Agriculture" by Rajasekaran, et al. Explores the role of big data analytics in agriculture, including data processing and visualization. [https://www.researchgate.net/publication/379404505_Role_of_Data_Visualization_and_Big_Data_Analytics_in_Smart_Agriculture]

[2] Lee, E. Y., & Parker, R. S. (2019). Mobile App Development with React Native. Packt Publishing. [https://www.packtpub.com/en-us/product/react-cross-platform-application-development-with-react-native-9781789136081]

[3] Lee, G. J., & Lee, M. Y. (2018). Cloud Computing for Smart Agriculture: A Survey. Future Generation Computer Systems, 78, 287-299. [https://www.researchgate.net/publication/321896167_A_Survey_Smart_agriculture_IoT_with_cloud_computing]

[4] Kim, Y. H., et al. (2020). Optimizing Cloud Resources for Machine Learning Applications. IEEE Transactions on Cloud Computing, 8(1), 123-135.

IT21231414 | Jayaneththi I.H.N.S

Information technology

RESEARCH GAP

These research gaps contribute to the development of more advanced and effective devices for quality check of fresh tomatoes, ultimately benefit tomato producers, processors, and consumers a like.

According to that research,

Developing advanced algorithms for real-time data analysis to enable immediate feedback on the quality of fresh tomatoes

Understanding the impact of environmental factors on the performance of quality check devices

Research could focus on developing intuitive interfaces that provide clear and actionable insights into the quality of fresh tomatoes.

RESEARCH QUESTION

How can environmental and agronomic factors be integrated into a predictive model to accurately forecast the optimal harvest dates for different tomato varieties?

SPECIFIC AND SUBJECTIVE

Specific Objective

To develop and validate a machine learning-based model that accurately predicts the optimal harvest dates for tomatoes by analyzing key environmental factors (temperature, humidity, light) and agronomic practices (soil fertility, irrigation, and planting schedules).

Sub Objective

- Develop a predictive model using machine learning algorithms
  - Identify and engineer relevant features from the collected data
    - Design and implement a decision support tool that integrates the predictive model

TECHNOLOGIES & ALGORITHMS

1. Real-time data collection and analysis

2. Machine Learning for prediction models

Algorithms

- **Machine Learning Models**: random forest Regression algorithms for predicting schedule

PROGRESS

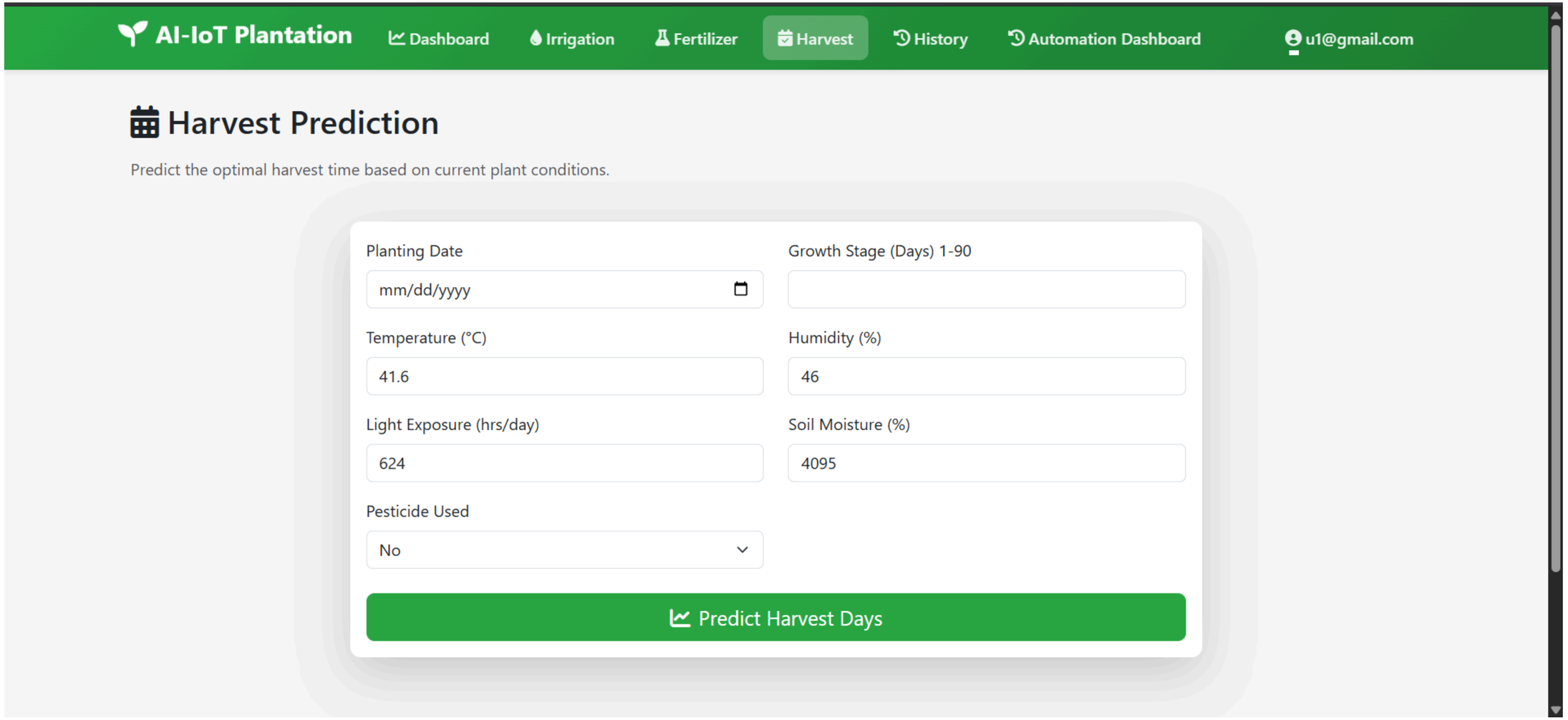
100% COMPLETION OF THE PROJECT

DATASET

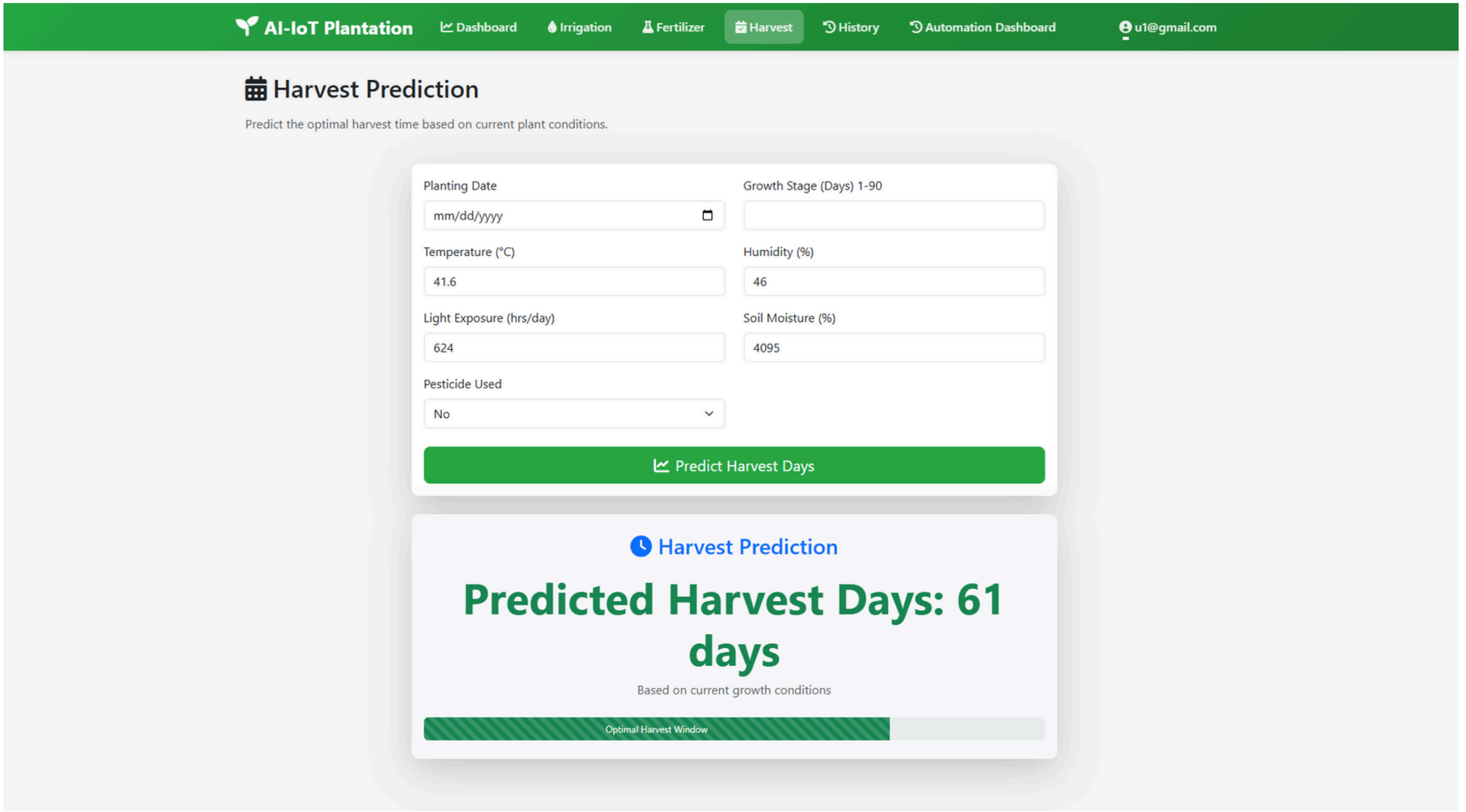
harvest_dataset

Planting Date	Growth Stage (Days)	Temperature (°C)	Humidity (%)	Light Exposure (hrs/day)	Soil Moisture (%)	Pesticide Used (Yes=1, No=0)	Harvest Days (Target)
04/25/2024	52	28.3	67.6	9.6	25.9	1	55
04/25/2024	24	27.2	59.2	10.0	31.0	0	55
04/25/2024	60	29.9	58.9	10.1	33.9	0	55
04/25/2024	51	26.4	59.9	10.1	30.8	0	55
04/25/2024	2	25.0	59.2	9.2	38.6	0	63
04/25/2024	4	32.6	66.0	10.3	38.8	0	58
04/25/2024	87	27.5	68.2	10.7	34.1	0	55
04/25/2024	29	25.6	63.1	11.5	36.6	1	55
04/25/2024	41	26.7	65.8	9.6	35.0	1	55
04/25/2024	44	26.9	65.0	10.5	36.4	0	55
04/25/2024	14	21.8	64.1	10.1	29.7	0	64
04/25/2024	23	24.2	61.0	10.2	35.4	0	76
04/25/2024	42	28.8	61.6	9.2	38.4	1	55
04/25/2024	89	26.5	58.2	11.9	37.9	0	55
04/25/2024	72	30.9	59.2	8.6	27.5	0	55
04/25/2024	88	27.5	59.6	7.1	29.3	1	55
04/26/2024	2	26.6	61.5	9.9	39.8	0	58
04/26/2024	68	24.4	56.7	9.8	39.8	0	55
04/26/2024	16	25.0	57.2	10.1	35.2	1	55
04/26/2024	67	25.6	58.7	9.8	32.8	1	55
04/26/2024	11	26.3	62.8	11.6	33.3	0	69
04/26/2024	2	27.1	62.5	9.8	29.2	1	69
04/26/2024	80	27.5	61.7	9.0	27.4	1	55
04/26/2024	84	28.1	59.2	9.8	31.7	0	55
04/26/2024	33	24.3	59.9	12.0	35.8	0	55
04/26/2024	48	24.3	56.1	11.3	34.5	0	58
04/26/2024	30	28.7	62.0	10.0	33.6	0	62
04/26/2024	17	23.4	58.8	10.5	39.3	1	55
04/26/2024	46	25.3	52.4	9.3	27.5	1	55
04/26/2024	54	25.2	60.8	10.1	38.2	0	55

User Interface



User Interface



REFERENCES

[1] Gutiérrez, S., Diacono, M., Castrignanò, A., & Sardone, R. (2016). Application of multivariate geostatistical methods for managing precision viticulture. *Computers and Electronics in Agriculture*, 121, 386–396. https://doi.org/10.1016/j.compag.2016.02.018

[2] Hewett, E. W. (2006). An overview of pre-harvest factors influencing postharvest quality of horticultural products. *International Journal of Postharvest Technology and Innovation*, 1(1), 4–15. https://doi.org/10.1504/IJPTI.2006.009629

[3] Saggi, M.K. & Jain, S. A. (2022). Survey Towards Decision Support System on Smart Irrigation Scheduling Using Machine Learning approaches. Archives of Computational Methods in Engineering, 29, 4455–4478. https://doi.org/10.1007/s11831-022-09746-3.

[4] Salim, O., Fouad, K. & Hassan, B. (2022). Dual-Level Sensor Selection with Adaptive Sensor Recovery to Extend WSNs' Lifetime, Human-centric Computing and Information Sciences, Vol. 12, Article number: 18, https://doi.org/10.22967/HCIS.2022.12.018.

[5] Goap, A., Sharma, D., Shukla, A.K. & Krishna, R. (2018). An IoT based smart irrigation management system using Machine learning and open source technologies, Computers and Electronics in Agriculture, Vol. 155, PP: 41-49, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2018.09.040.

[6] Fouad, K. & Elbably, D. (2020).Intelligent approach for large-scale data mining. Int. J. Computer Applications in Technology, Vol. 63, Nos. 1/2, PP: 93–112.

[7] Fouad, K., Hassan, B. & Salim, O. (2022). Hybrid Sensor Selection Technique for Lifetime Extension of Wireless Sensor Networks. Computers, Materials & Continua 2022, 70(3), 4965-4985. https://doi.org/10.32604/cmc.2022.020926.

IT21186592 | Najas MNM

Information technology

COMPONENT

Detecting Tomato Diseases and Recommending Appropriate Treatment

IT21186592 | NAJAS MNM | 24-25J-064

SLIIT
FACULTY OF COMPUTING

INTRODUCTION BACKGROUND

- This project aims to develop a comprehensive system that leverages image analysis and machine learning to detect tomato diseases and recommend appropriate treatments.
- The Impact of Diseases on Tomato Crops.
- Challenges in Disease Management.

RESEARCH QUESTIONS

How can we develop a decision support system for tomato greenhouses ?

RESEARCH GAP

Existing Products / Research	Predictive Analytics				
	Image Analysis Tools	Manual Inspection	Disease Specific Detection	Automated Treatment Recommendation	Predictive Analytics
Current Methods	✗	✓	✓	✗	✗
Basic Automated Tools	✓	✓	✗	✗	✗
Proposed System	✓	✗	✓	✓	✓

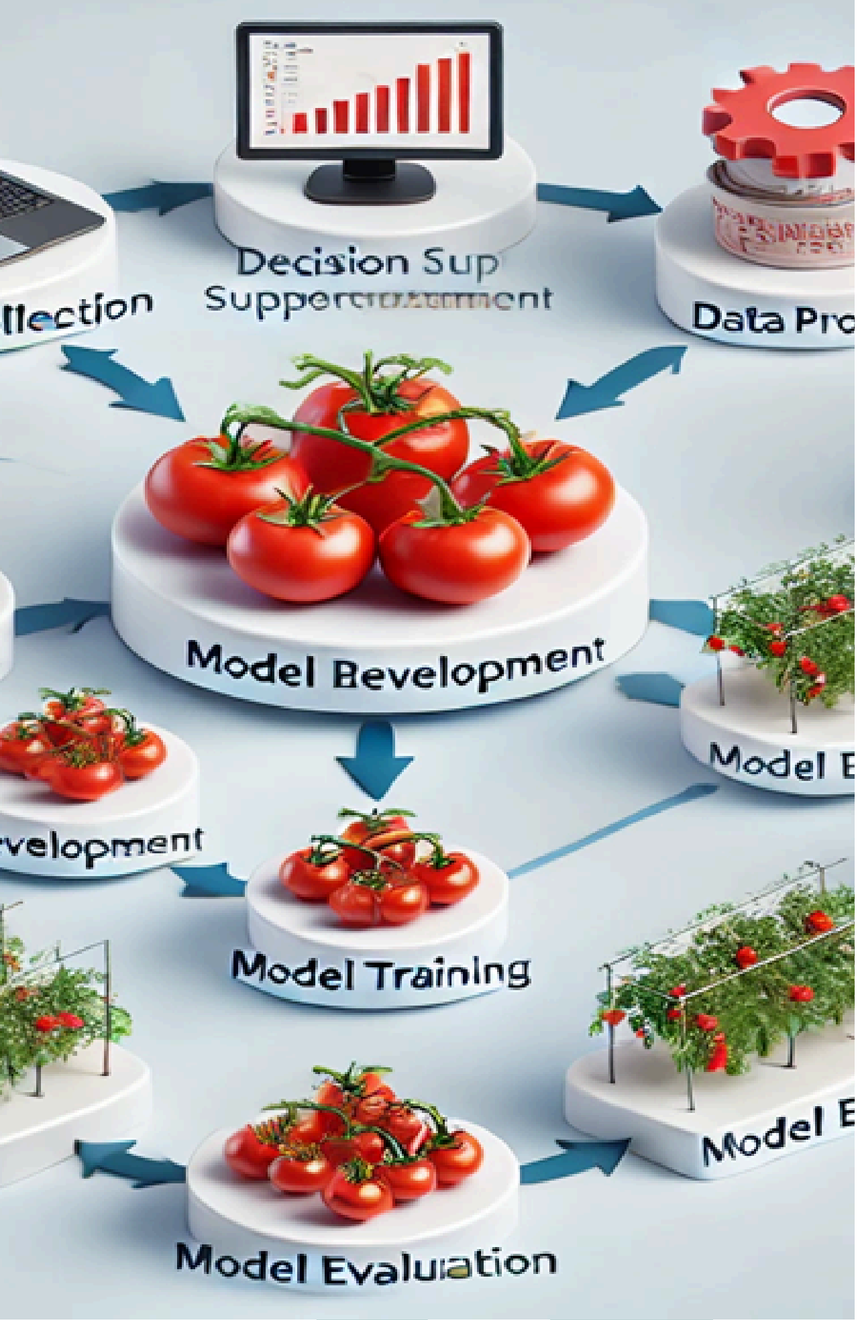
MAIN & SPECIFIC OBJECTS

Main objectives

Develop a comprehensive system to detect tomato diseases and recommend appropriate treatments using image analysis and machine learning.

Sub objectives

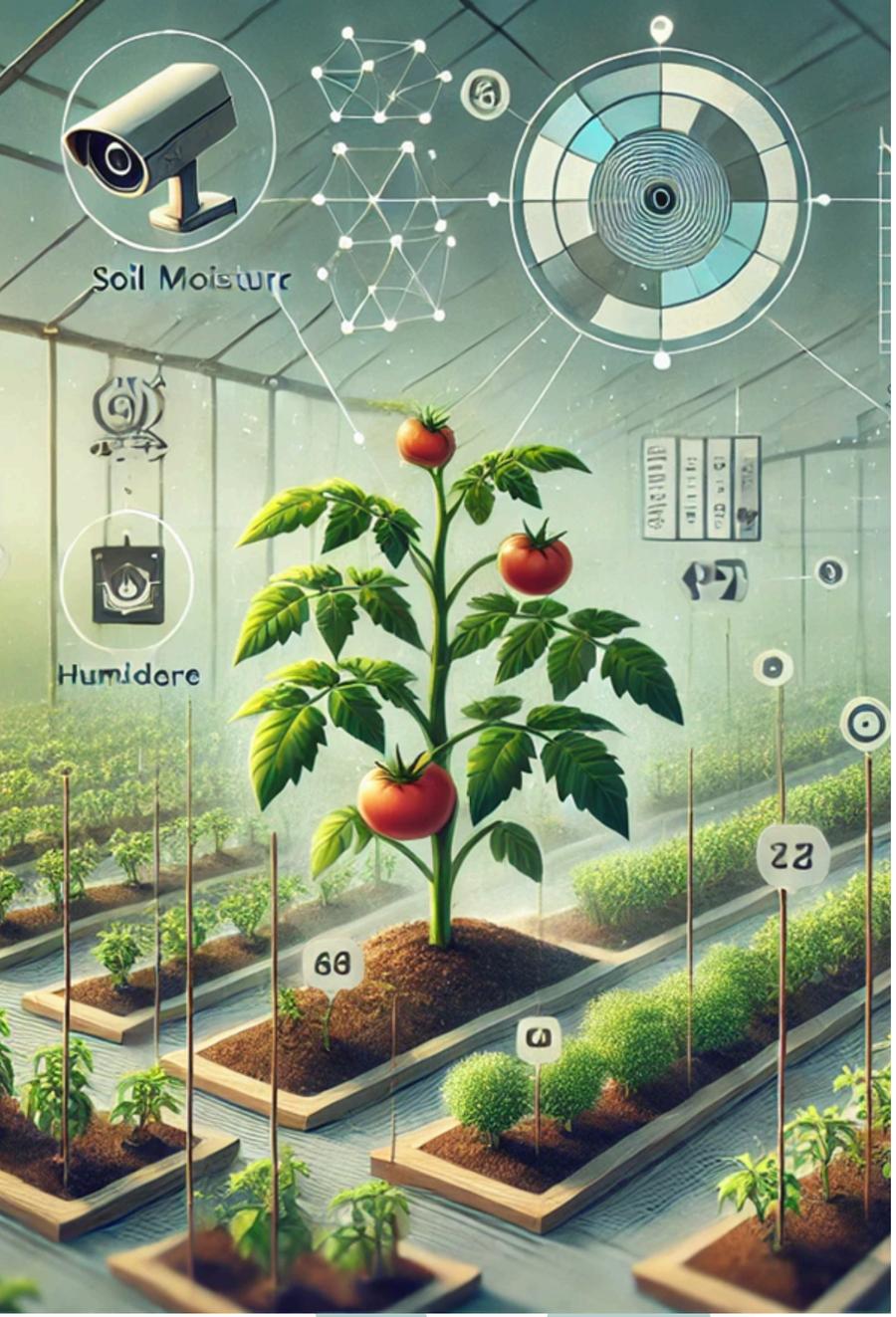
- Select data from Kaggle.
- Label the images with corresponding disease types.
- Preprocess the images (resize, normalize, augment) for model training.
- Develop a model architecture suitable for image classification.
- Train the CNN model on the labeled dataset and evaluate its performance.



METHODOLOGY

- Python, TensorFlow(Open-Source ML framework), Keras.
- Image preprocessing techniques.
- CNN model design and training.
- Data visualization tools.

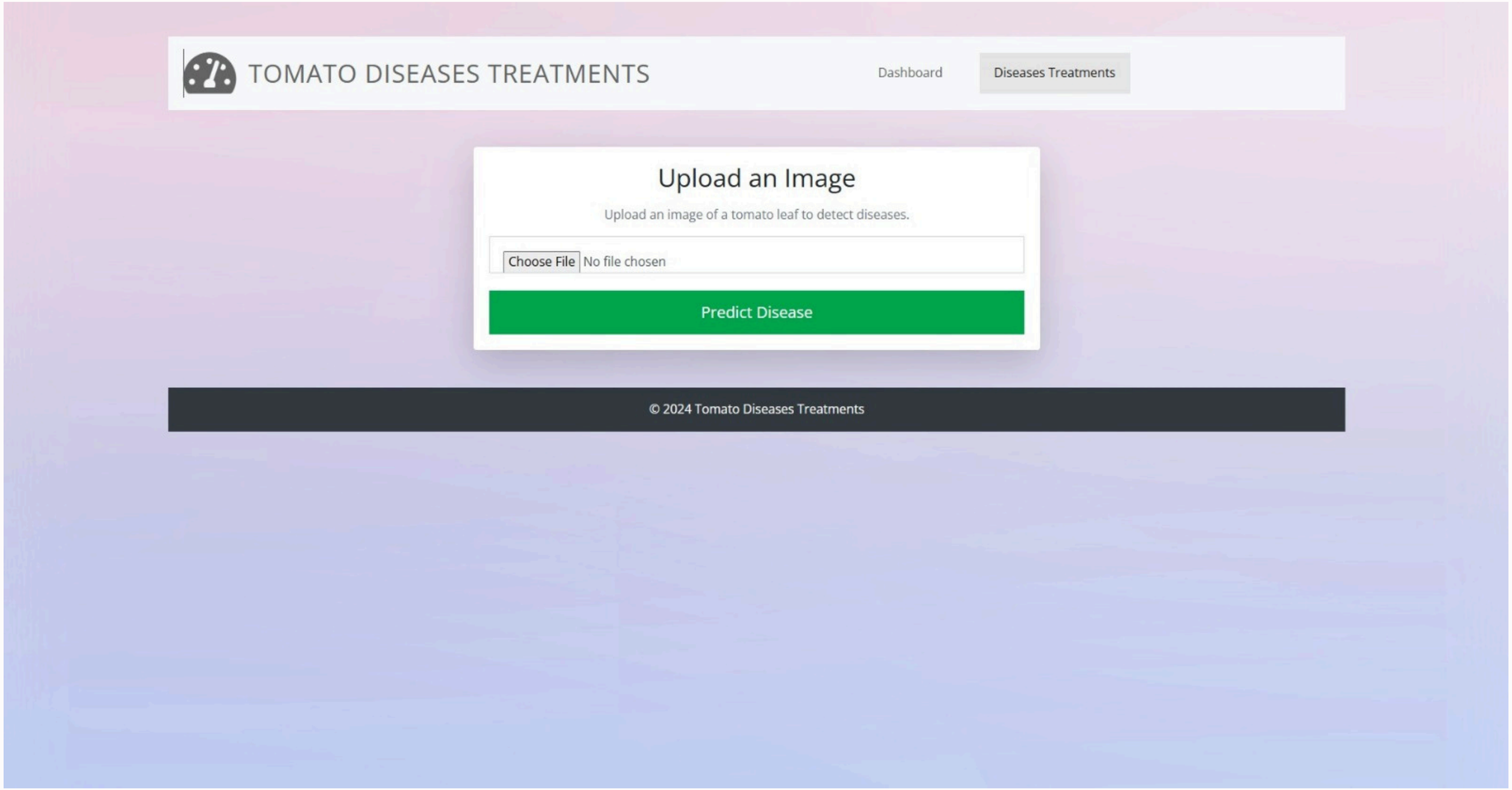
•**Data Collection and Labeling**: Select and annotate images from Kaggle.
•**Image Preprocessing**: Resize, normalize, and augment images.
•**CNN Model Development**: Design and train a CNN model for disease classification.
•**Performance Evaluation**: Evaluate the model's performance using metrics such as accuracy, precision, and recall.
•**Implementation of Recommendation System**: Provide treatment recommendations based on disease detection.



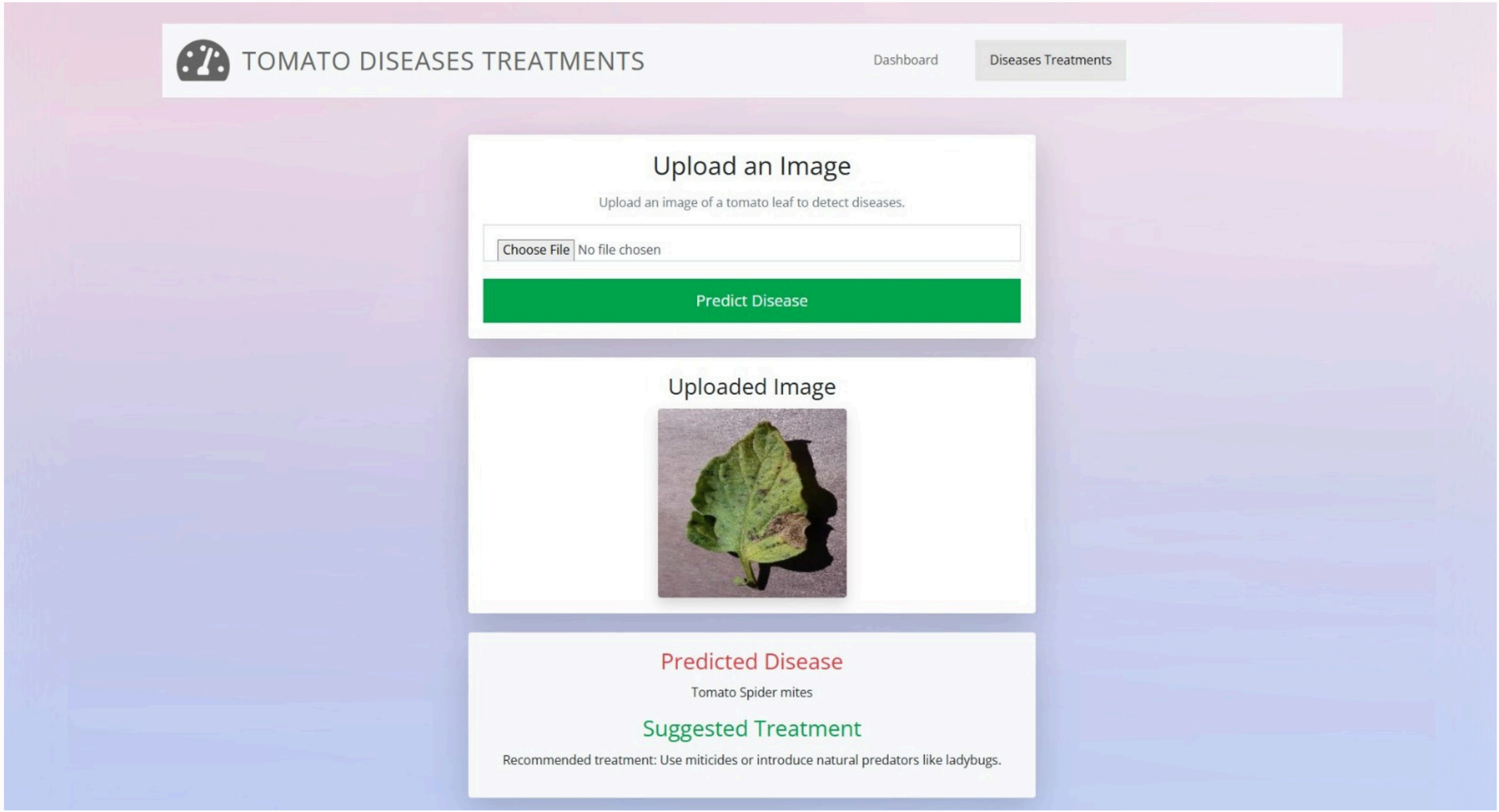
DATA COLLECTION

- **Kaggle datasets.**

UI


TOMATO DISEASES TREATMENTS
Dashboard
Diseases Treatments
Upload an Image
Upload an image of a tomato leaf to detect diseases.
Choose File
No file chosen
Predict Disease
© 2024 Tomato Diseases Treatments

OUTPUT


TOMATO DISEASES TREATMENTS
Dashboard
Diseases Treatments
Upload an Image
Upload an image of a tomato leaf to detect diseases.
Choose File
No file chosen
Predict Disease
Uploaded Image
Predicted Disease
Tomato Spider mites
Suggested Treatment
Recommended treatment: Use miticides or introduce natural predators like ladybugs.

REQUIREMENTS

System requirements

- High-quality annotated image datasets.
- Efficient CNN model for disease detection.
- Treatment recommendation algorithms.
- User-friendly dashboard for real-time alerts and recommendations.

Non-functional Requirements

- Performance
- Reliability
- Scalability
- Security
- Usability
- Maintainability
- Accuracy

REFERENCES

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

Zhang, S., Huang, W., Zhang, C., & Wang, X. (2018). Plant disease recognition based on plant leaf image. Journal of Agricultural Mechanics Research, 9, 94-99.

Barbedo, J. G. A. (2016). A review on the main challenges in automatic plant disease identification based on visible range images. Biosystems Engineering, 144, 52-60.