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

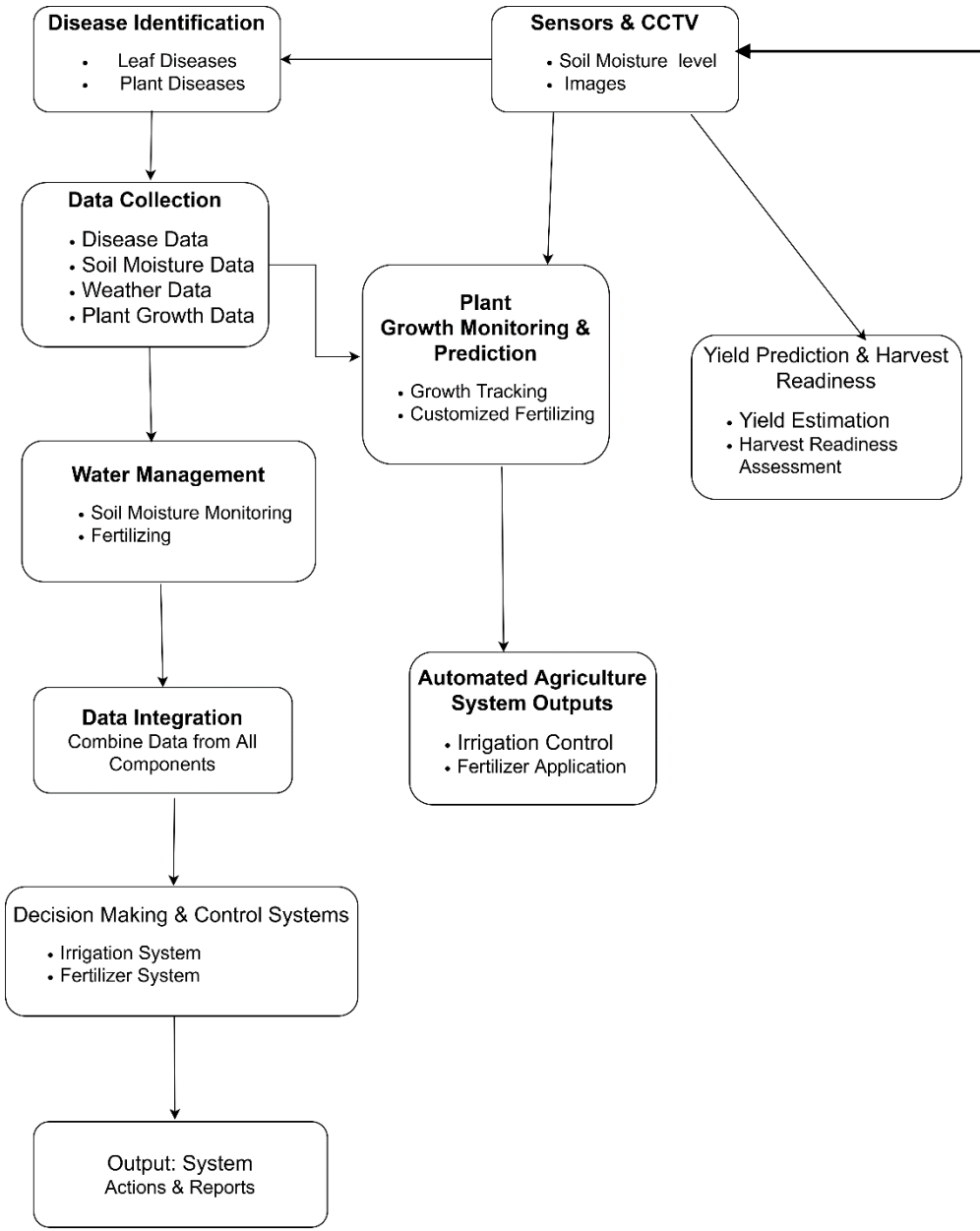
DEVELOP A DISEASE IDENTIFICATION SYSTEM

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

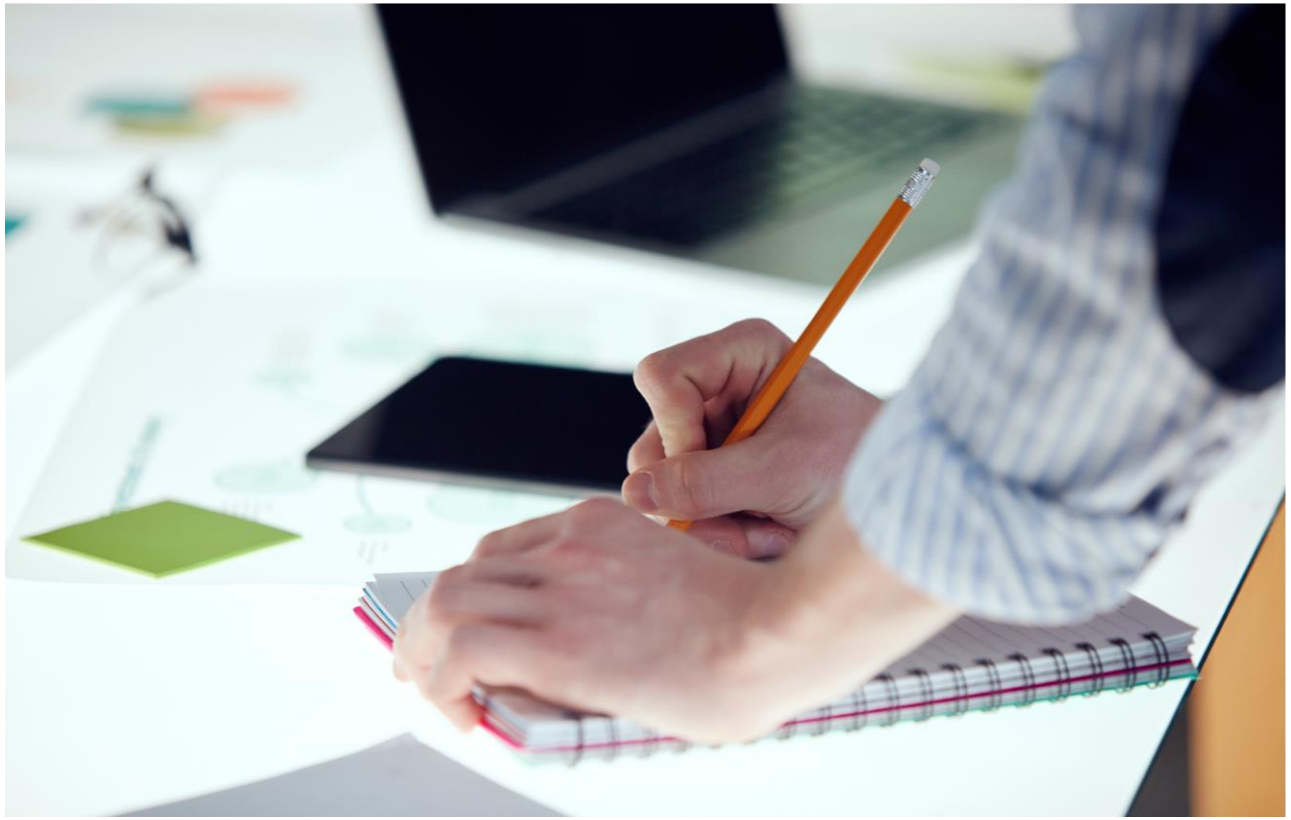
Inconsistent watering practices can lead to over- or under-watering, affecting crop health and yield

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

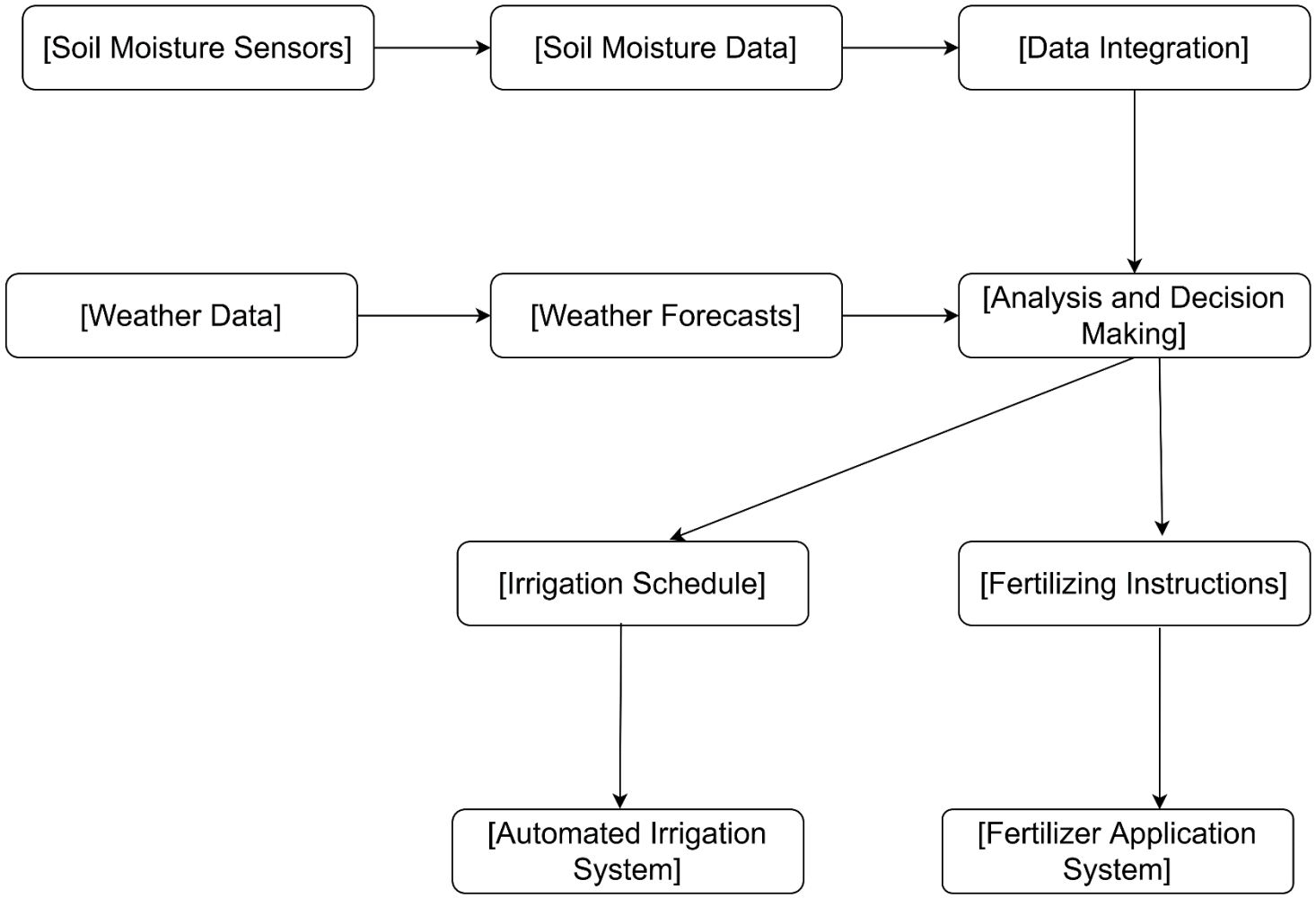


01 Soil Moisture-Based Watering

02 Weather Forecast Integration

03 Automated Watering System

System Diagram



Technologies

- Python

- Flask

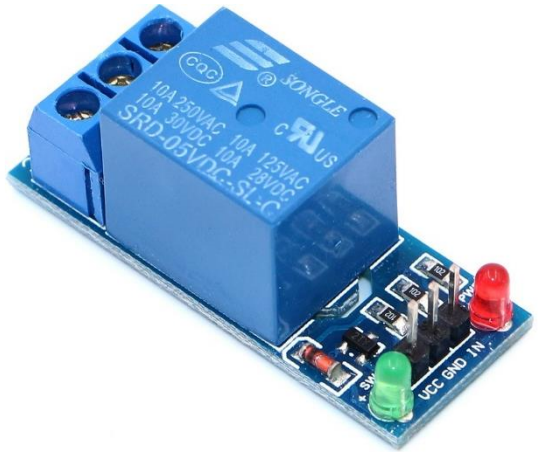
- Arduino IDE
- PyCharm

- Firebase

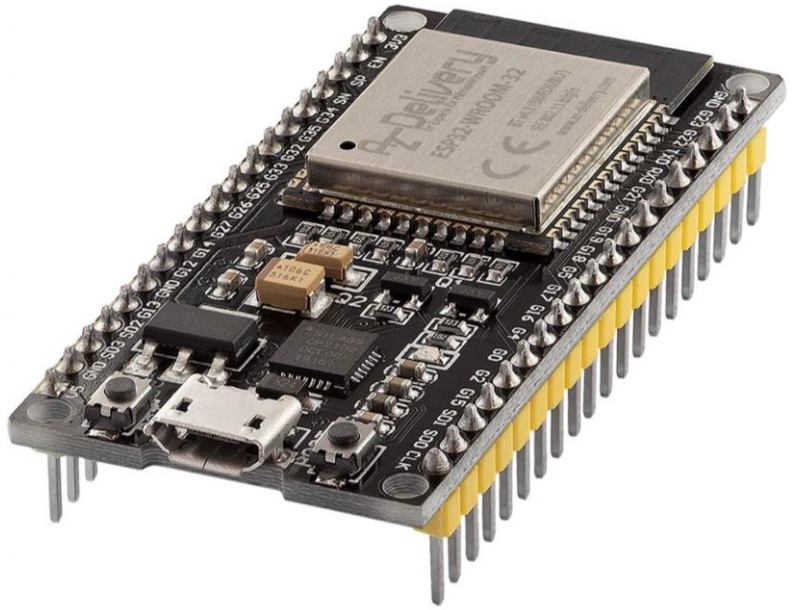
Tools

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.

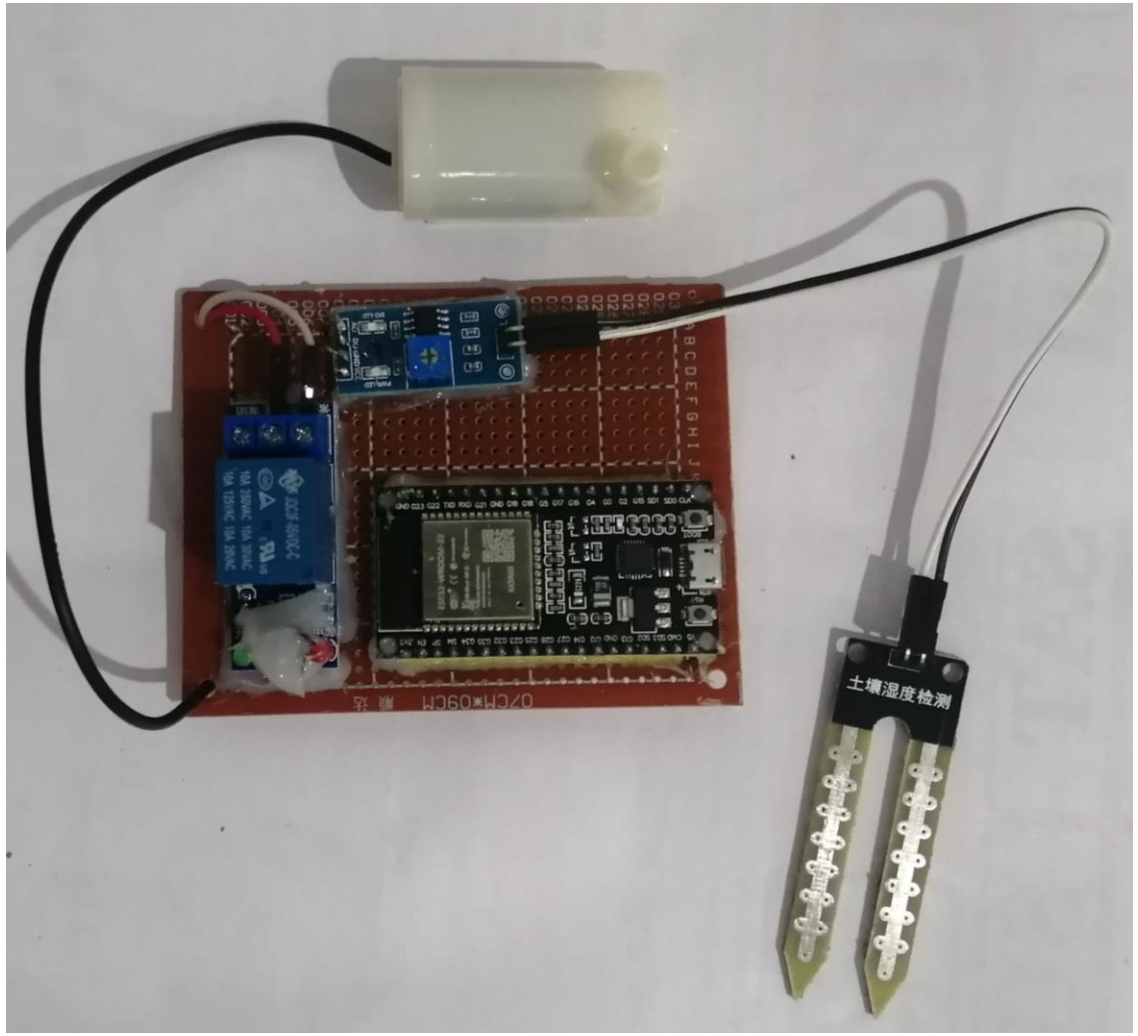


Channel-5V-Relay-Module :
turn on and off the pump automatically as the required water level is reached



ESP32 Microcontroller:
For data processing and communication.

Prototype



Requirements

Functional

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

Non Functional

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

Completed

- Built prototype 50%
- Integrate Tools
- Model Training

To be Complete

- Built IOT Device 100%
- 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.

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.

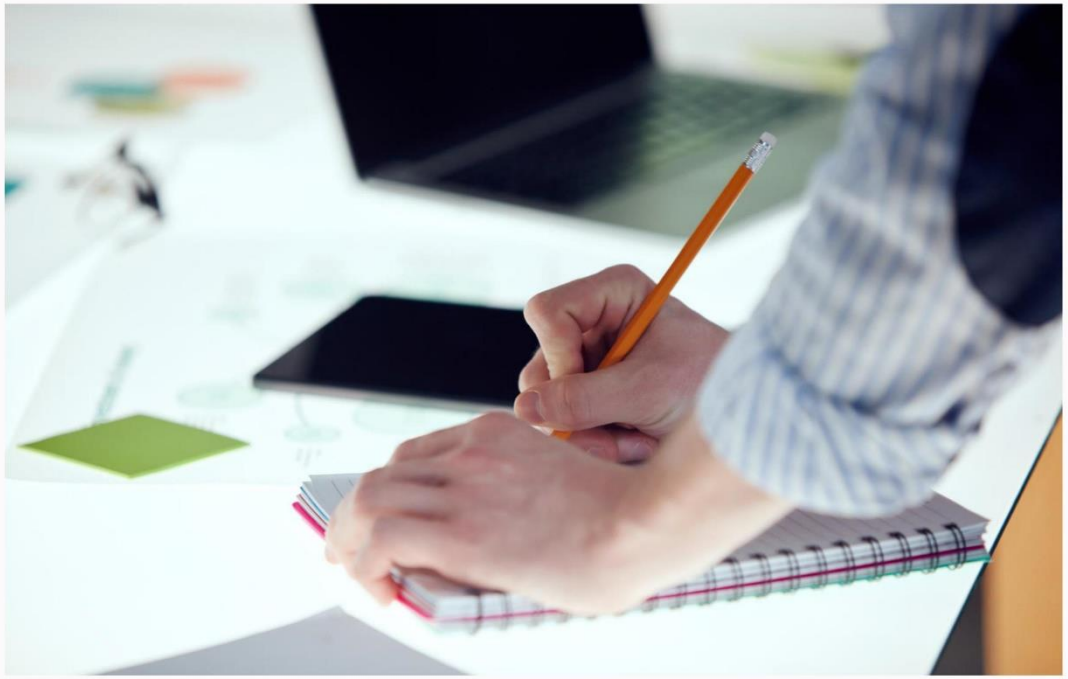
OBJECTIVES

Weather Forecast Integration

Develop a system for monitoring plant growth.

Predict growth outcomes based on customized fertilization

Methodology

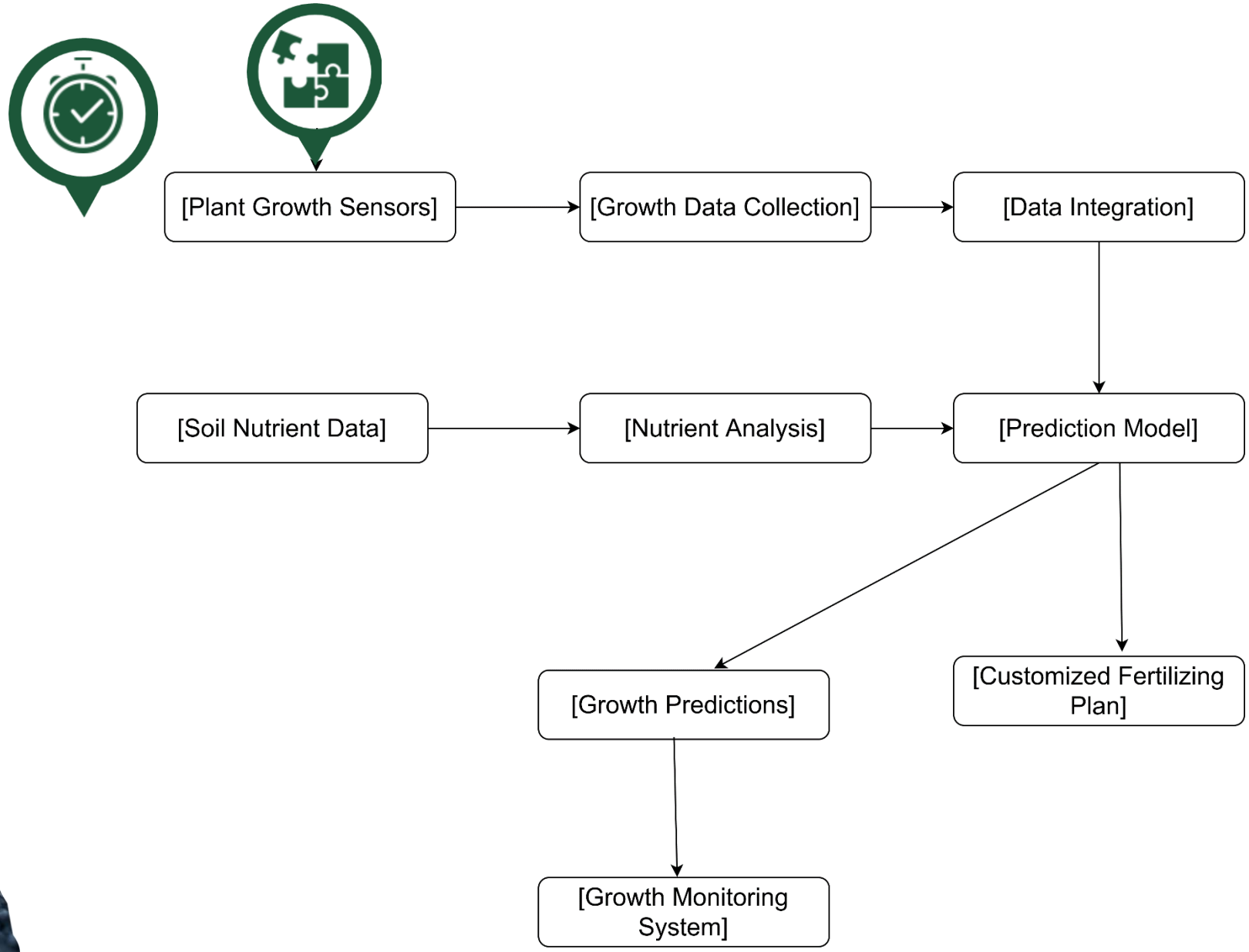


01 System Diagram

02 Technologies

03 Requirements

System Diagram



Technologies

- Python

- Flask

- Arduino IDE
- PyCharm

- Firebase

Requirements

Functional

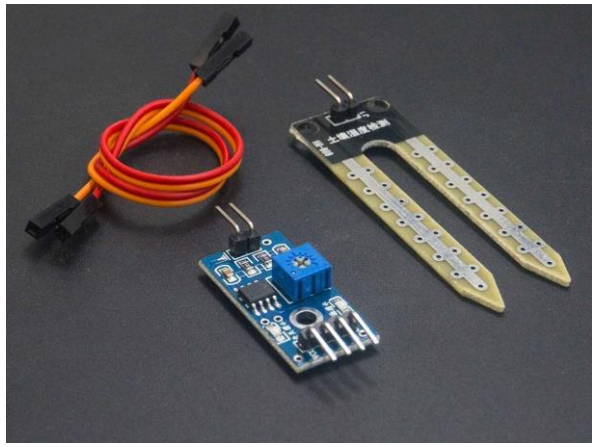
- Growth monitoring software
- Machine learning models for prediction
- Analytical software for growth predictions

Non Funcational

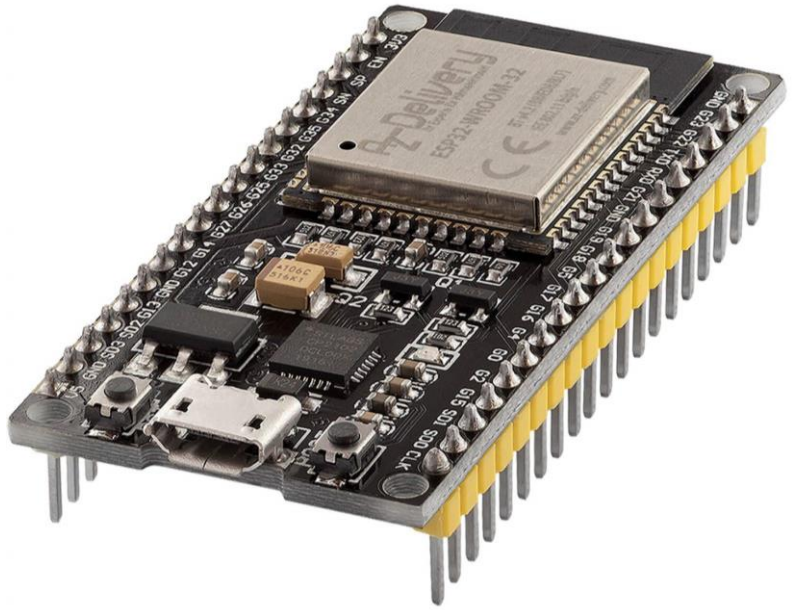
- Accuracy
- Performance
- Availability
- Usability

Tools

IOT-Device Circuits Items

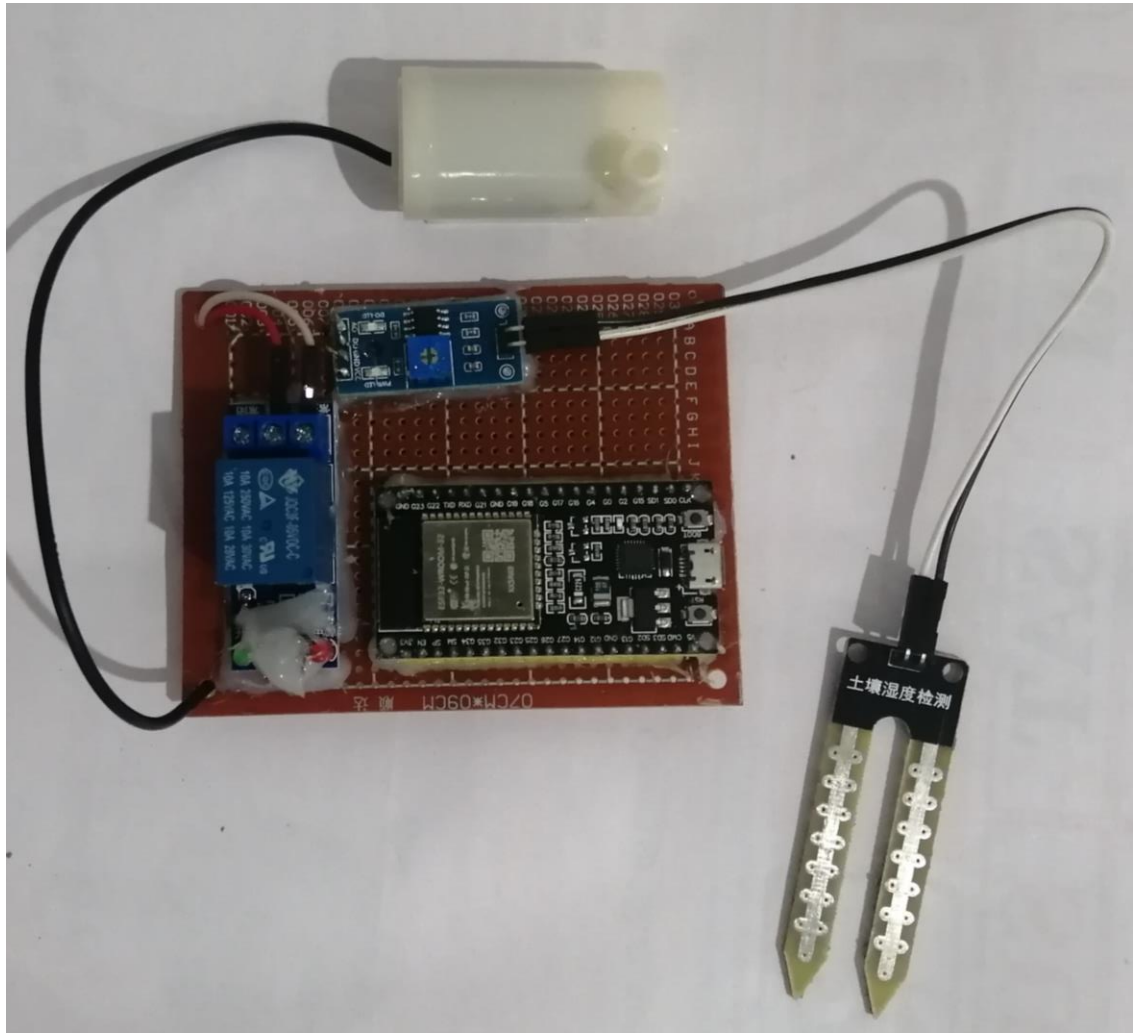


Soil Moisture Sensor Module:
measure or estimate the amount of water in the soil.



ESP32 Microcontroller:
For data processing and communication.

Prototype



Completed

- Built prototype 50%
- Integrate Tools
- Model Training

To be Complete

- Built IOT Device 100%
- 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.

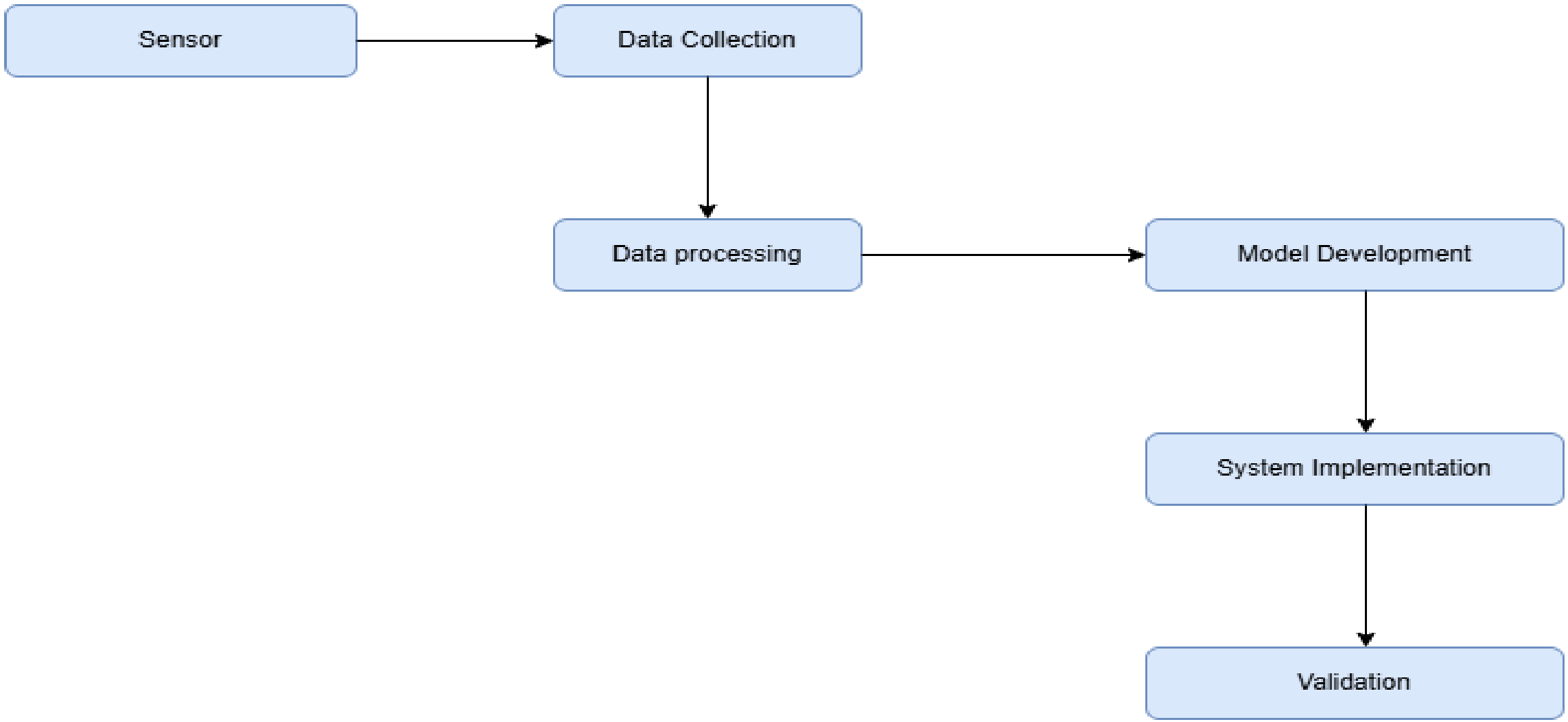
04 System Implementation

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

- Arduino IDE
- PyCharm

- Firebase

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

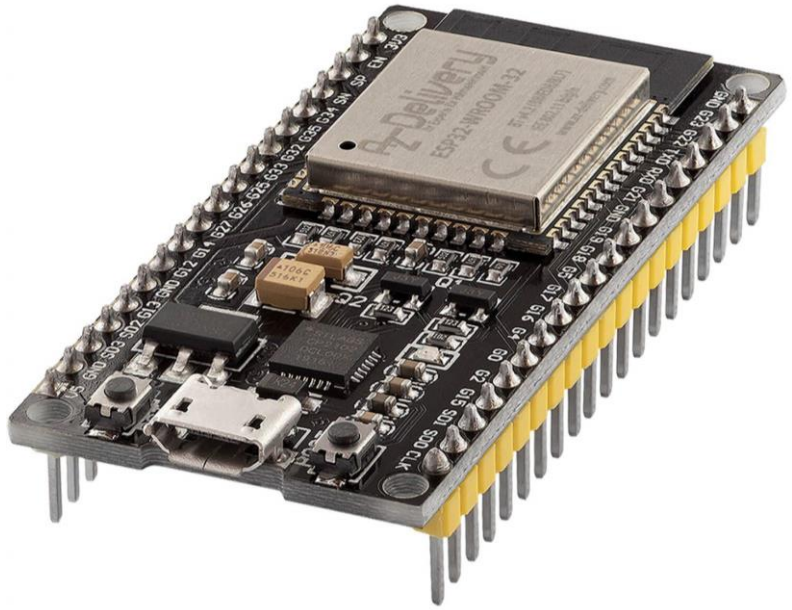
Non Funcational

- **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

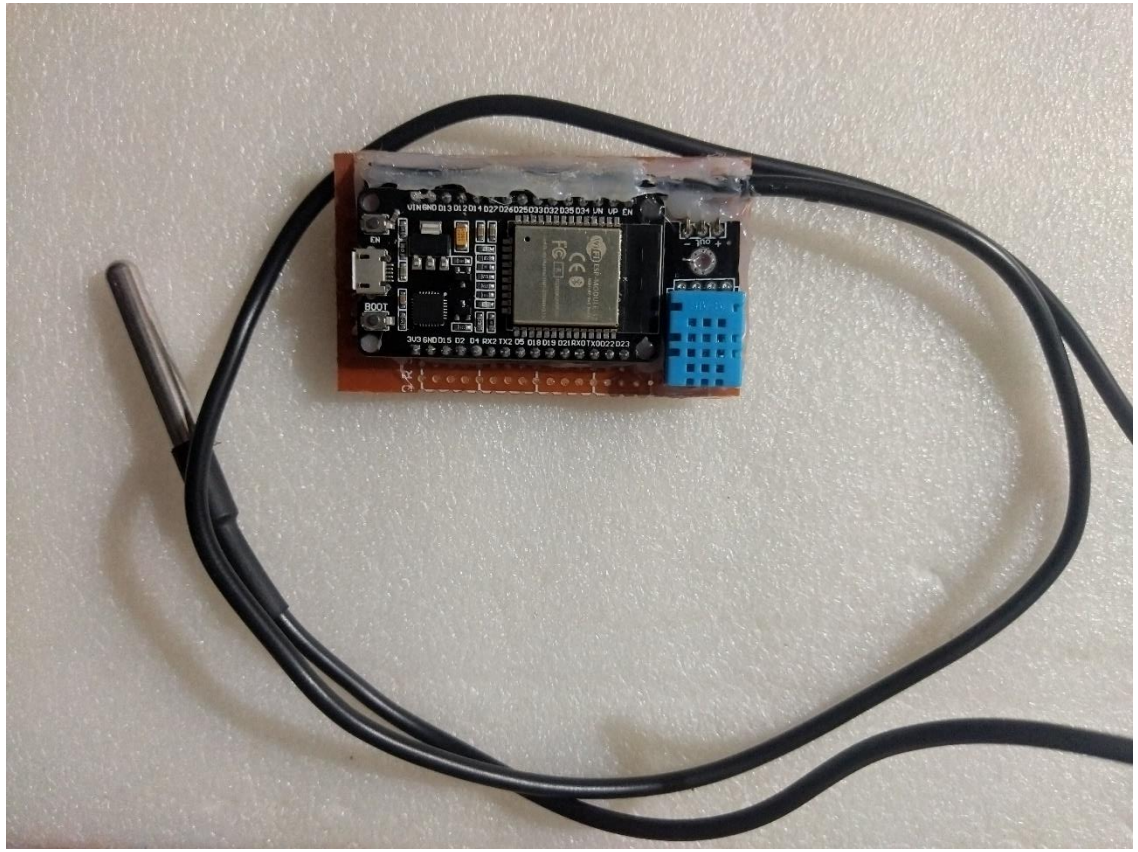
Soil Temperature Sensor :
DS18B20 or similar



Environment Temperature sensor ,Environment humidity sensor :
DHT22/DHT11

ESP32 Microcontroller:
For data processing and communication.

Prototype



Completed

- Built prototype 50%
- Integrate Sensors & Tools
- Model Development

To be Complete

- Built IOT Device 100%
- 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.

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

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

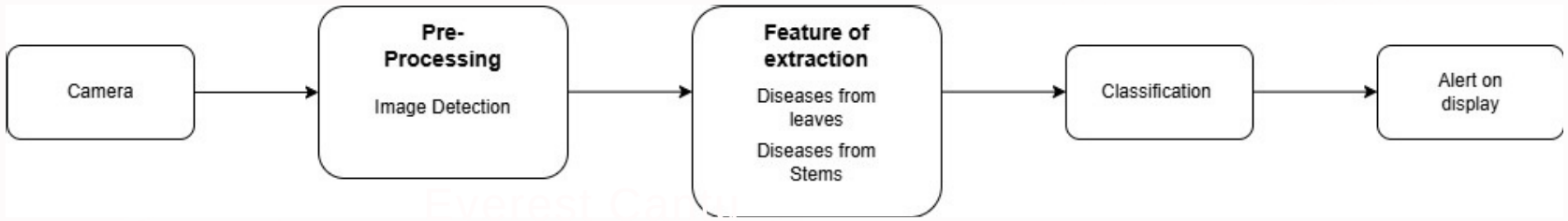
Frame Works
CNN ,
Keras/Tensorflow

Tools
GIT

Project Management
Ms Planner

Programming language
React Js , Python

System Diagram



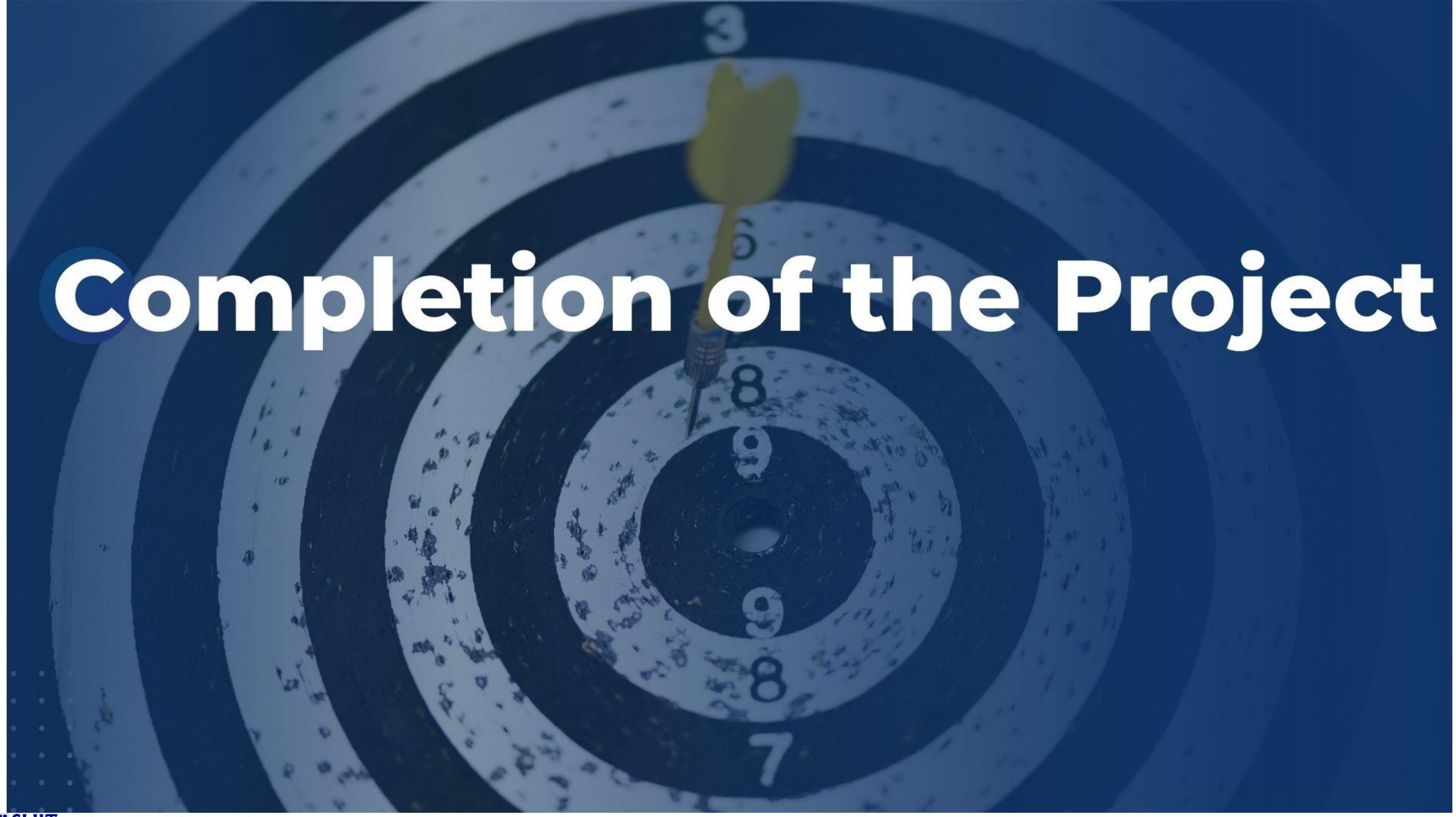
Requirements

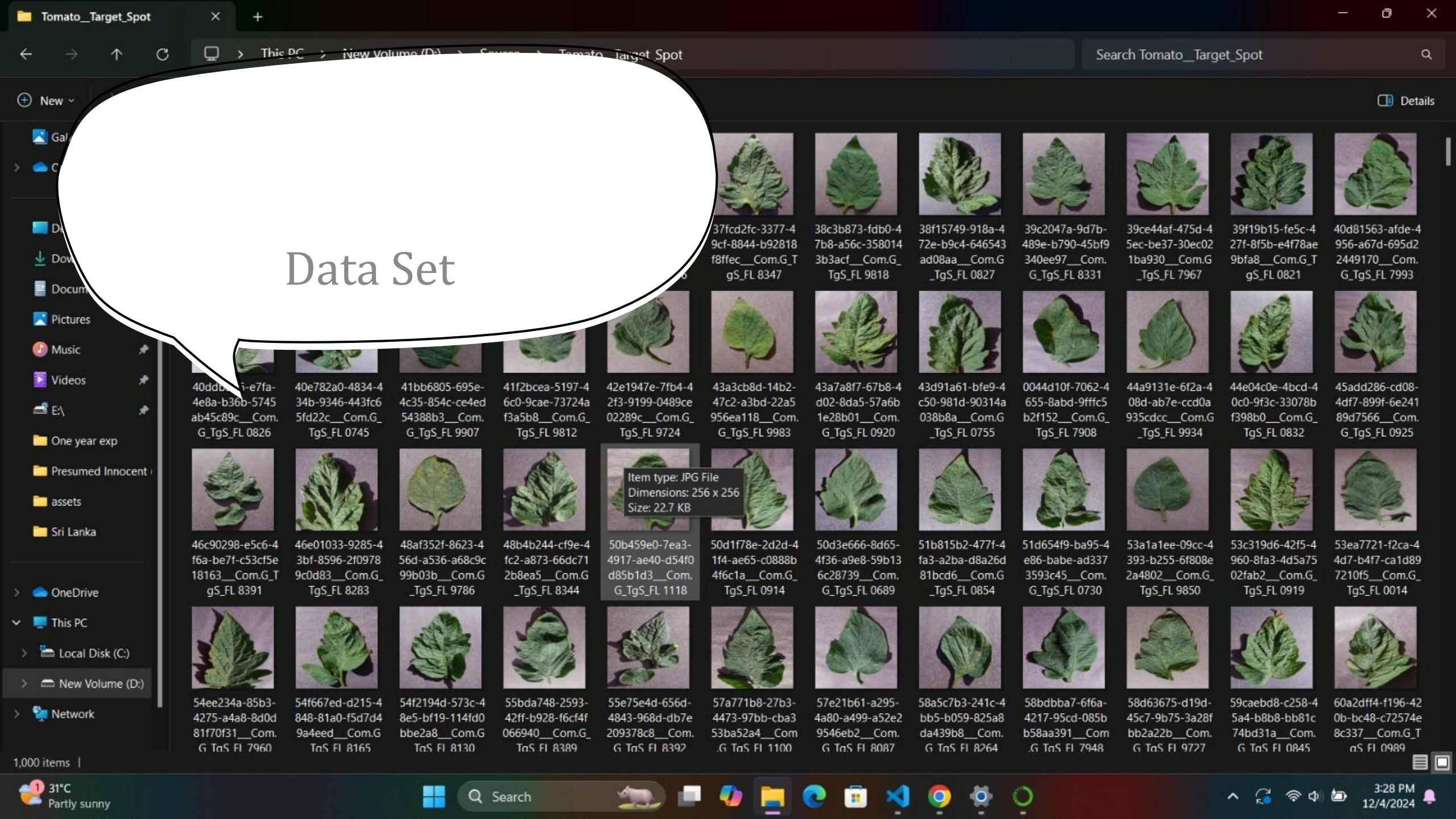
Functional

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

Non Funcational

- Accuracy
- performance
- Availability
- usability


Completion of the Project


Tomato_Target_Spot
Search Tomato_Target_Spot
New
Details
Data Set
Item type: JPG File
Dimensions: 256 x 256
Size: 22.7 KB
1,000 items
31°C
Partly sunny
Search
3:28 PM
12/4/2024

Model traning

[94]:
```
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=20,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_steps=val_generator.samples // val_generator.batch_size
)
```

```
D:\Anaconda\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 501s 3s/step - accuracy: 0.4100 - loss: 1.6853 - val_accuracy: 0.7386 - val_loss: 0.7765
Epoch 2/20
  1/167 ━━━━━━━━━━━━━━━━━━━━ 6:00 2s/step - accuracy: 0.7188 - loss: 0.7873
D:\Anaconda\Lib\contextlib.py:158: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` function when building your dataset.
  self.gen.throw(value)
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.7188 - loss: 0.7873 - val_accuracy: 0.6129 - val_loss: 1.1197
Epoch 3/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 508s 3s/step - accuracy: 0.7031 - loss: 0.8617 - val_accuracy: 0.7699 - val_loss: 0.6860
Epoch 4/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.7188 - loss: 0.7269 - val_accuracy: 0.7419 - val_loss: 0.8467
Epoch 5/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 496s 3s/step - accuracy: 0.7578 - loss: 0.7389 - val_accuracy: 0.8159 - val_loss: 0.5531
Epoch 6/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - accuracy: 0.8750 - loss: 0.4716 - val_accuracy: 0.8387 - val_loss: 0.5377
Epoch 7/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 508s 3s/step - accuracy: 0.7726 - loss: 0.6701 - val_accuracy: 0.7812 - val_loss: 0.6225
Epoch 8/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.7500 - loss: 0.6024 - val_accuracy: 0.6452 - val_loss: 0.7693
Epoch 9/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 505s 3s/step - accuracy: 0.7999 - loss: 0.6041 - val_accuracy: 0.8108 - val_loss: 0.5596
Epoch 10/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8125 - loss: 0.5451 - val_accuracy: 0.8387 - val_loss: 0.4991
Epoch 11/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 502s 3s/step - accuracy: 0.8032 - loss: 0.5691 - val_accuracy: 0.8006 - val_loss: 0.5571
Epoch 12/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.8125 - loss: 0.5418 - val_accuracy: 0.8065 - val_loss: 0.5554
Epoch 13/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 487s 3s/step - accuracy: 0.8238 - loss: 0.5130 - val_accuracy: 0.8284 - val_loss: 0.5220
Epoch 14/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - accuracy: 0.8125 - loss: 0.6151 - val_accuracy: 0.8065 - val_loss: 0.5095
Epoch 15/20
167/167 ━━━━━━━━━━━━━━━━━━━━ 483s 3s/step - accuracy: 0.8150 - loss: 0.5357 - val_accuracy: 0.8040 - val_loss: 0.5898
Epoch 16/20
```

Working component

```python
# Function to make a prediction
def predict_image_class(img_path, class_names):
    # Preprocess the image
    img_array = load_and_preprocess_image(img_path)

    # Make a prediction
    predictions = model.predict(img_array)

    # Get the class with the highest probability
    predicted_class = np.argmax(predictions, axis=1)

    # Get the class name
    class_name = class_names[predicted_class[0]]

    print(f"Predicted class: {class_name}")
    return class_name

# Specify the path to your image
img_path ="D:/Base/test/Tomato_Leaf_Mold/4babadaf-7d3f-4cc8-8179-05d86893dd97___Crnl_L.Mold 9096.JPG" # Replace with the actual path to your image

# Define class names (must match the class names used during training)
class_names = ['Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_Late_blight' ,'Tomato_Leaf_Mold','Tomato_Septoria_leaf_spot','Tomato_Spider_mites_

# Predict the image class
predicted_class = predict_image_class(img_path, class_names)
```

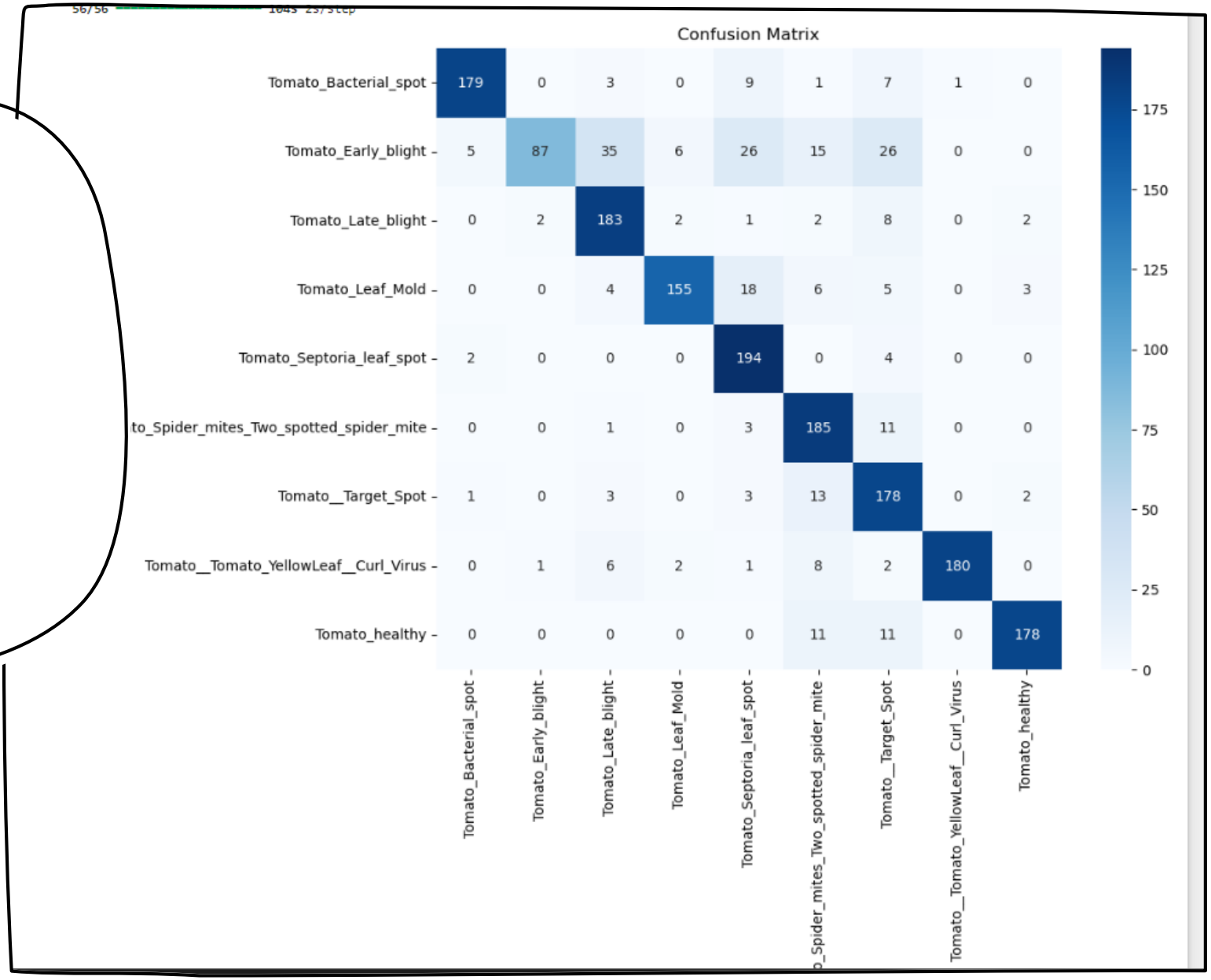
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 252ms/step
Predicted class: Tomato_Leaf_Mold
```

Model accuracy

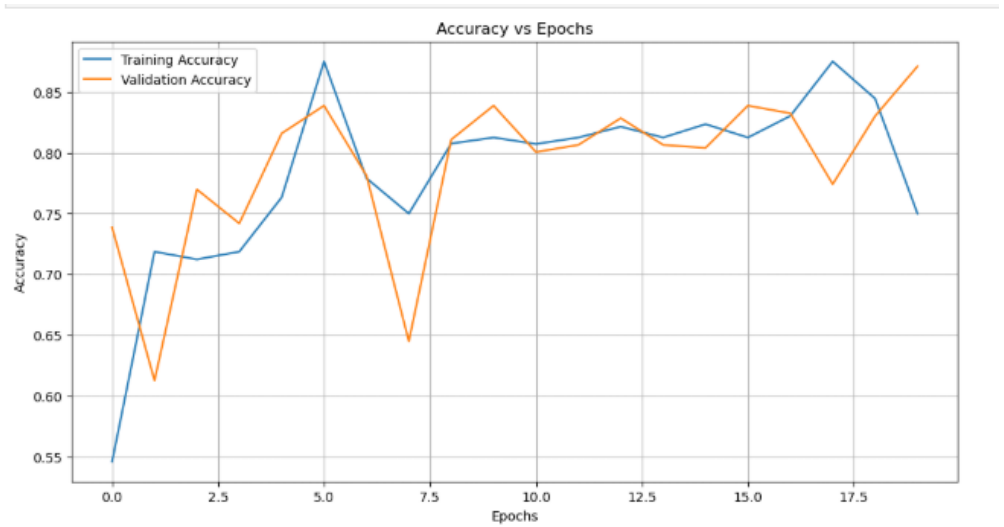
```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test accuracy: {test_accuracy:.4f}")
```

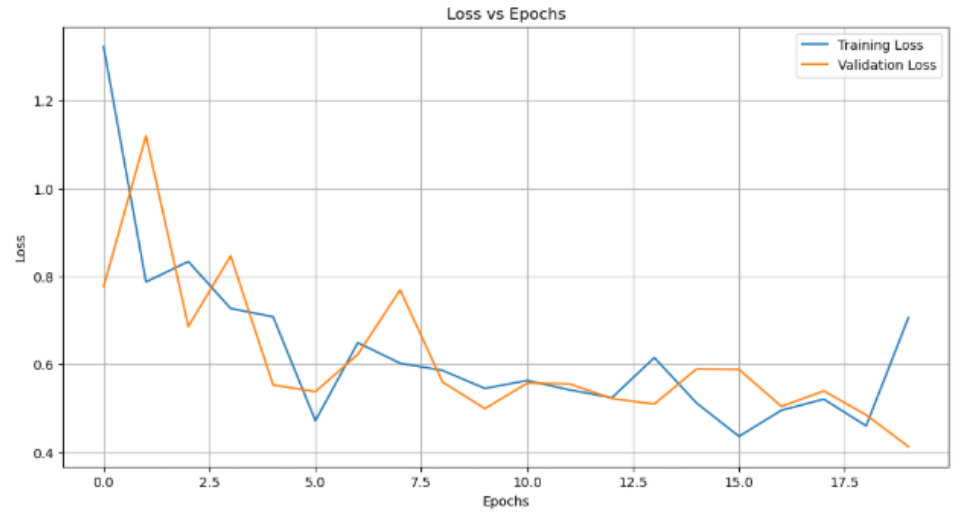
```
56/56 ━━━━━━━━━━━━━━━━━━━━ 120s 2s/step - accuracy: 0.8432 - loss: 0.4751
Test accuracy: 0.8481
```

Confusion matrix

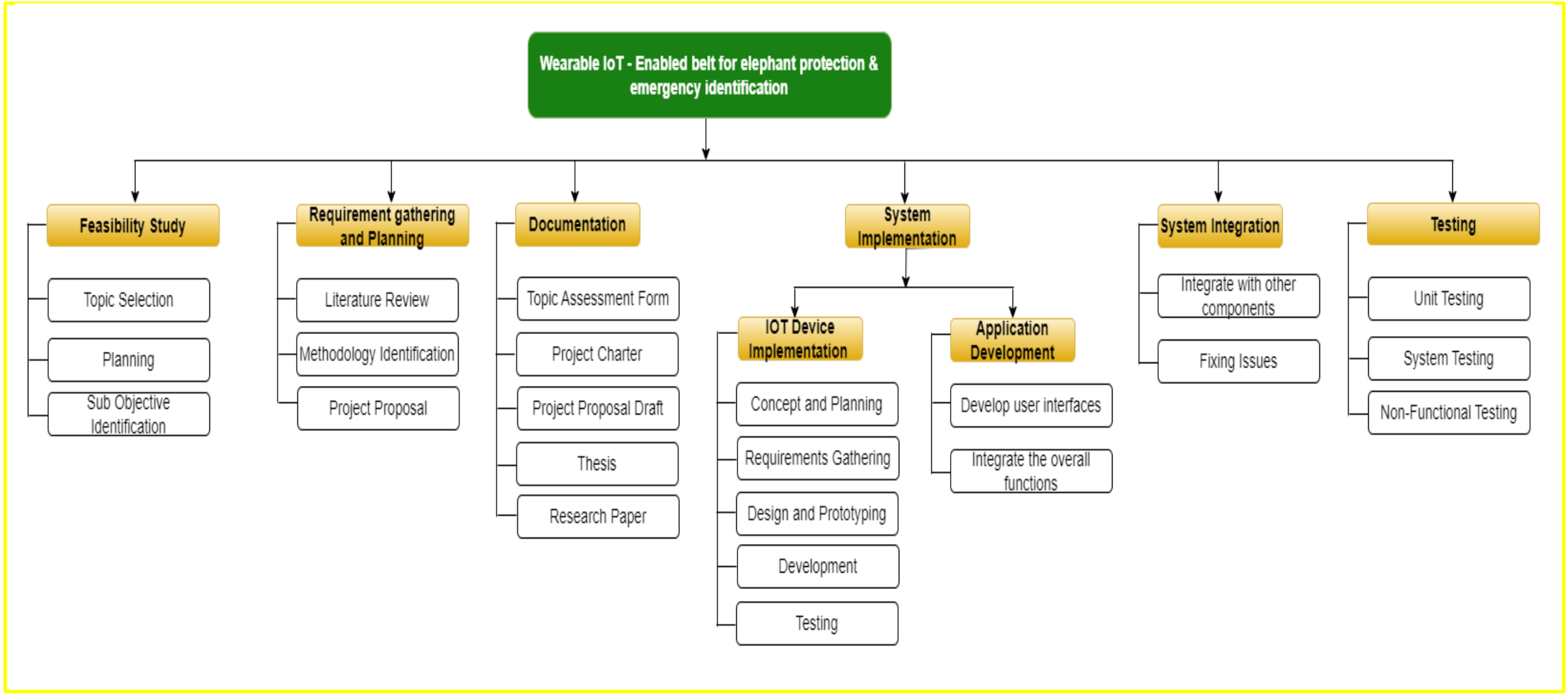


Accuracy


Accuracy vs Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epochs
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Loss vs Epochs
Training Loss
Validation Loss
Loss
Epochs
0.4
0.6
0.8
1.0
1.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Work Breakdown Structure

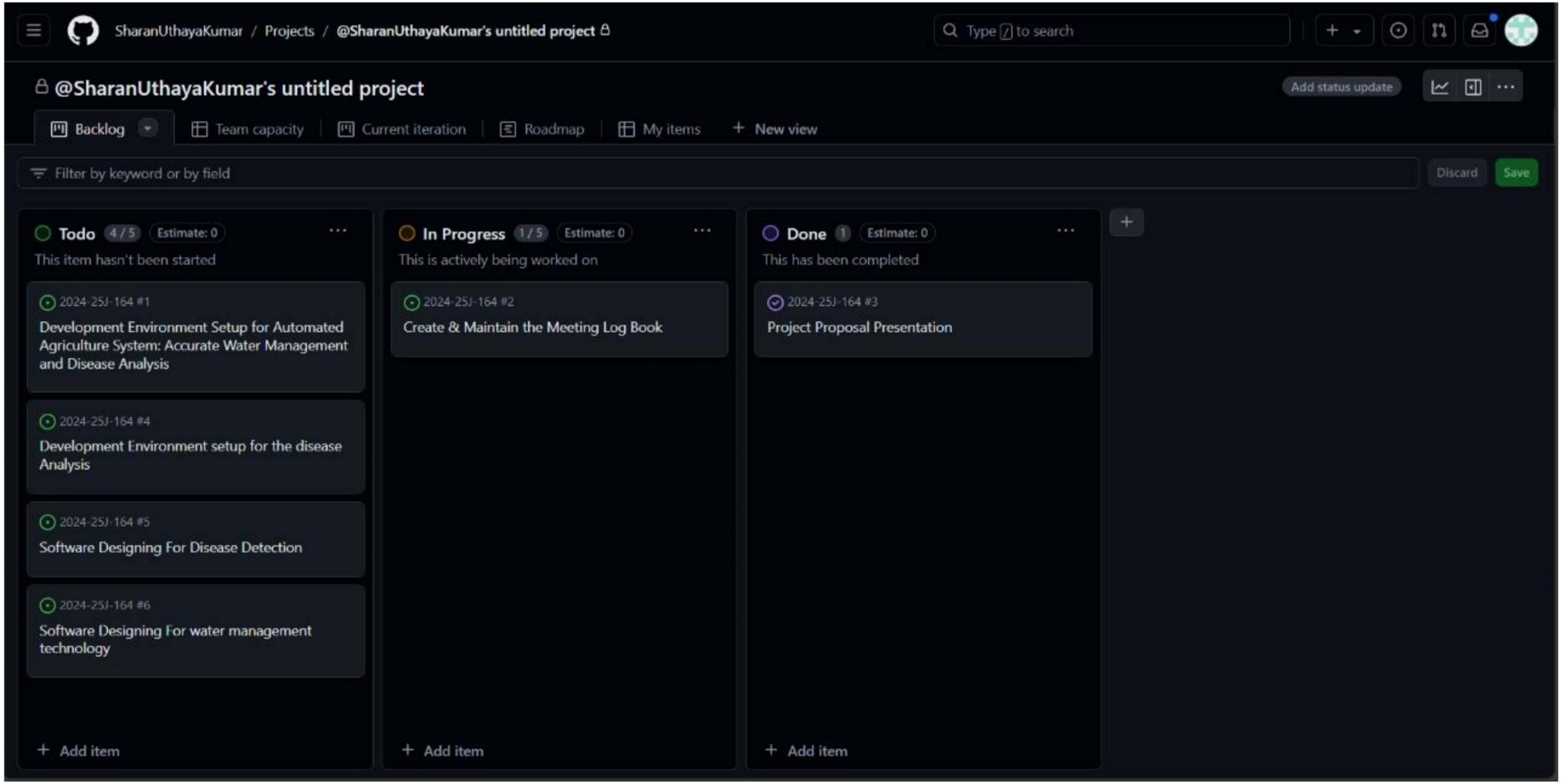


Cost Management Plan

Budget

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

Project Management Plan




Q
A

Thanks