Papaya Buddy

Enhancing Papaya Cultivation Through Advanced Monitoring

24-25J-069

Introduction

- Papaya is a vital crop in tropical agriculture, contributing significantly to both the economy and nutrition. It covers extensive cultivation areas and provides essential income for many farmers.

- The high incidence of pests and diseases poses significant threats to papaya production and farm productivity. Early detection and management of these diseases are critical to sustaining healthy crops and maximizing yields.

- A smart agricultural solution with automated plant disease detection and classification tools offers a valuable resource for supporting farm decision-making. Our system integrates advanced Deep Learning technologies to diagnose papaya plant diseases accurately and efficiently, even those undetectable by the untrained human eye.

Research Question

- Many diseases are difficult to detect in their early stages, leading to severe outbreaks and significant crop losses.

- Most farmers are not aware of the specific diseases and pests affecting papaya cultivation. They only recognize that their plants are diseased without understanding the exact nature of the problem.

- Due to this lack of knowledge, farmers often resort to using any available chemical treatments or removing affected plants entirely, leading to reduced income and yield.

How to Identify ?

- Papaya Mosaic and Cercospora Leaf diseases
- Papaya Ringspot Virus and Powdery Mildew Fruit diseases
- Mite and Mealy Bugs
- Maturity level of papaya fruit

Main & Sub Objectives

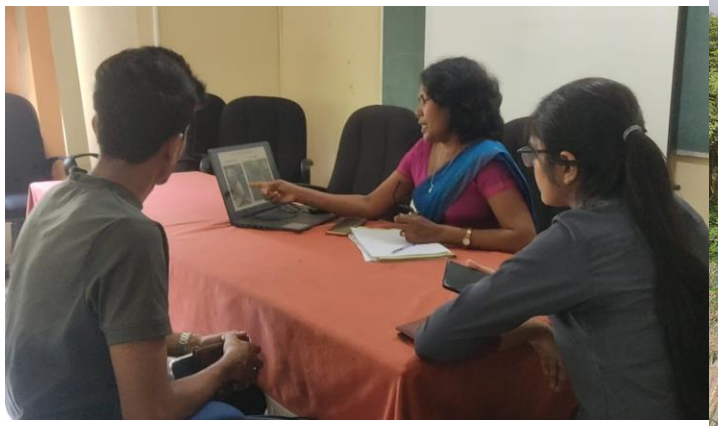
Main Objective

Advanced Monitoring system for disease and pest controlling for papaya cultivation.

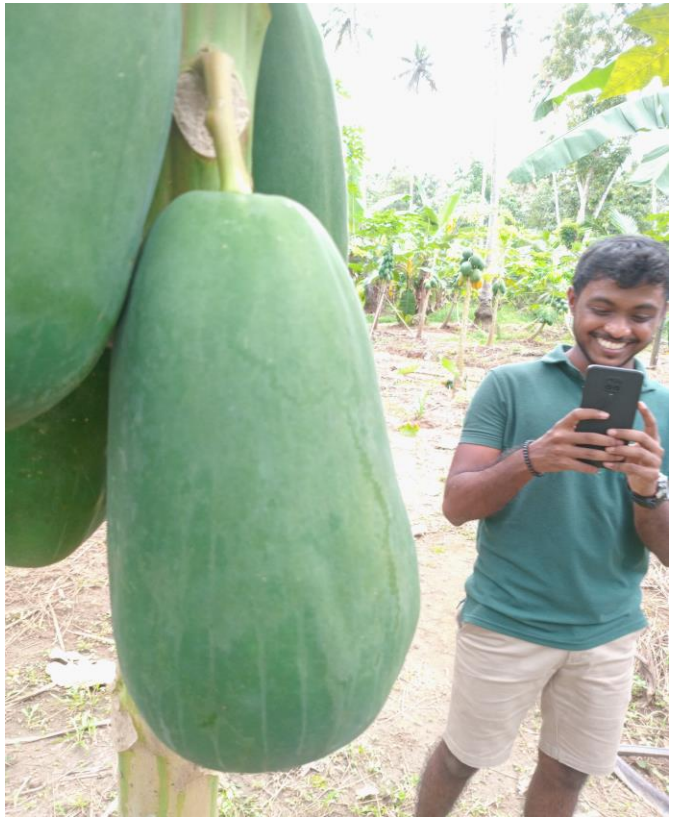
Sub Objectives

1. Identification and classification of Papaya Mosaic and Cercospora leaf disease.
2. Identification and classification of Papaya Ringspot Virus and Powdery Mildew fruit disease.
3. Identification and classification of Mite and Mealy Bug.
4. Identification and Classification of maturity level of papaya fruit

Field Visits



Field Visit – FRID Horana



Diwulapitiya Farm

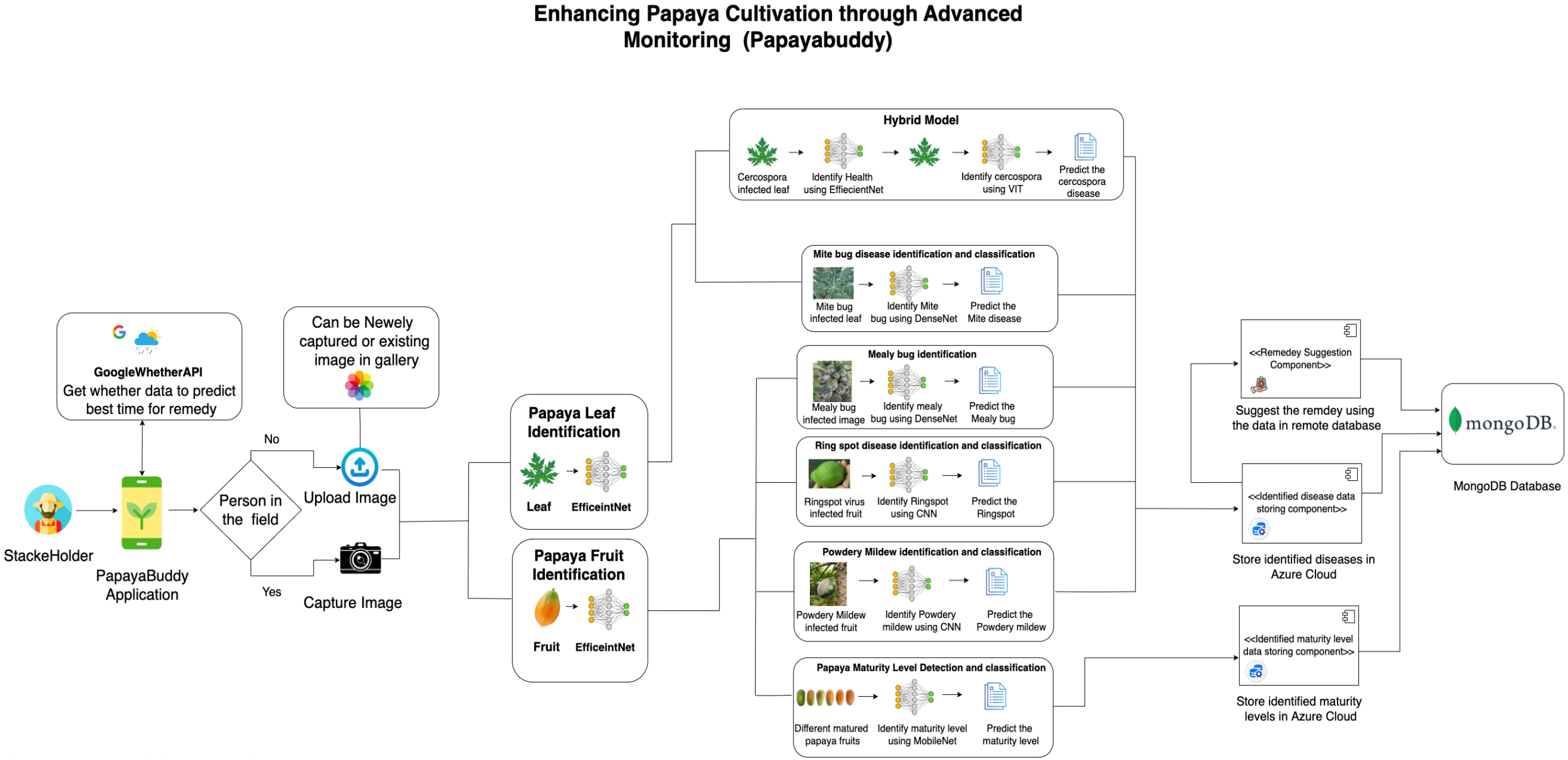
Infected Plants in early stage

After 4 months

The plants are infected with diseases

4th Field Visit - Yield Loss - Diwulapitiya

System Diagram



Why we selected Microservices Architecture

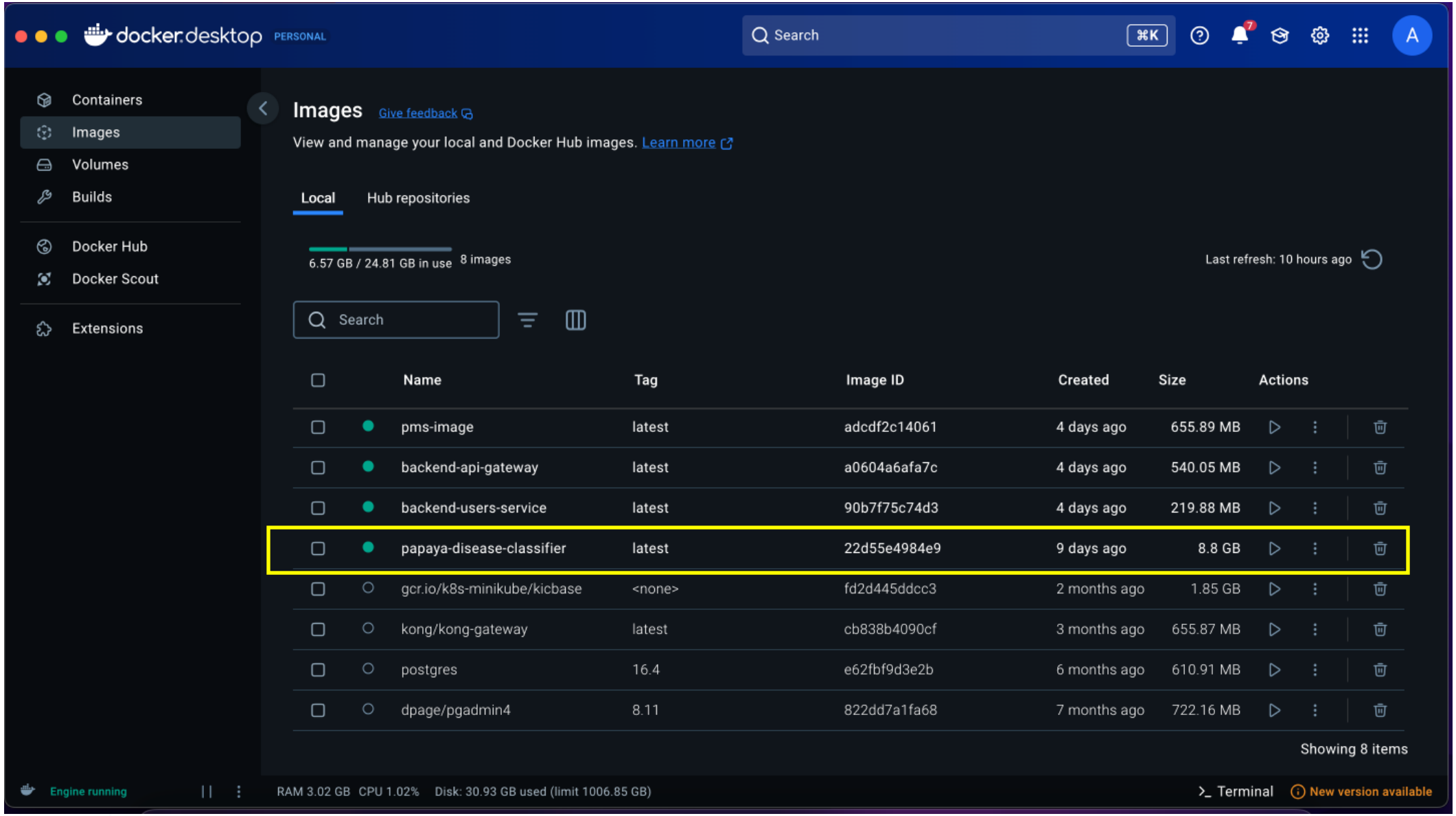
- **Dependency Conflicts**
  Different models require different libraries, leading to version mismatches.

- **Scalability Issues**
  Scaling one model requires scaling the entire system, increasing costs.

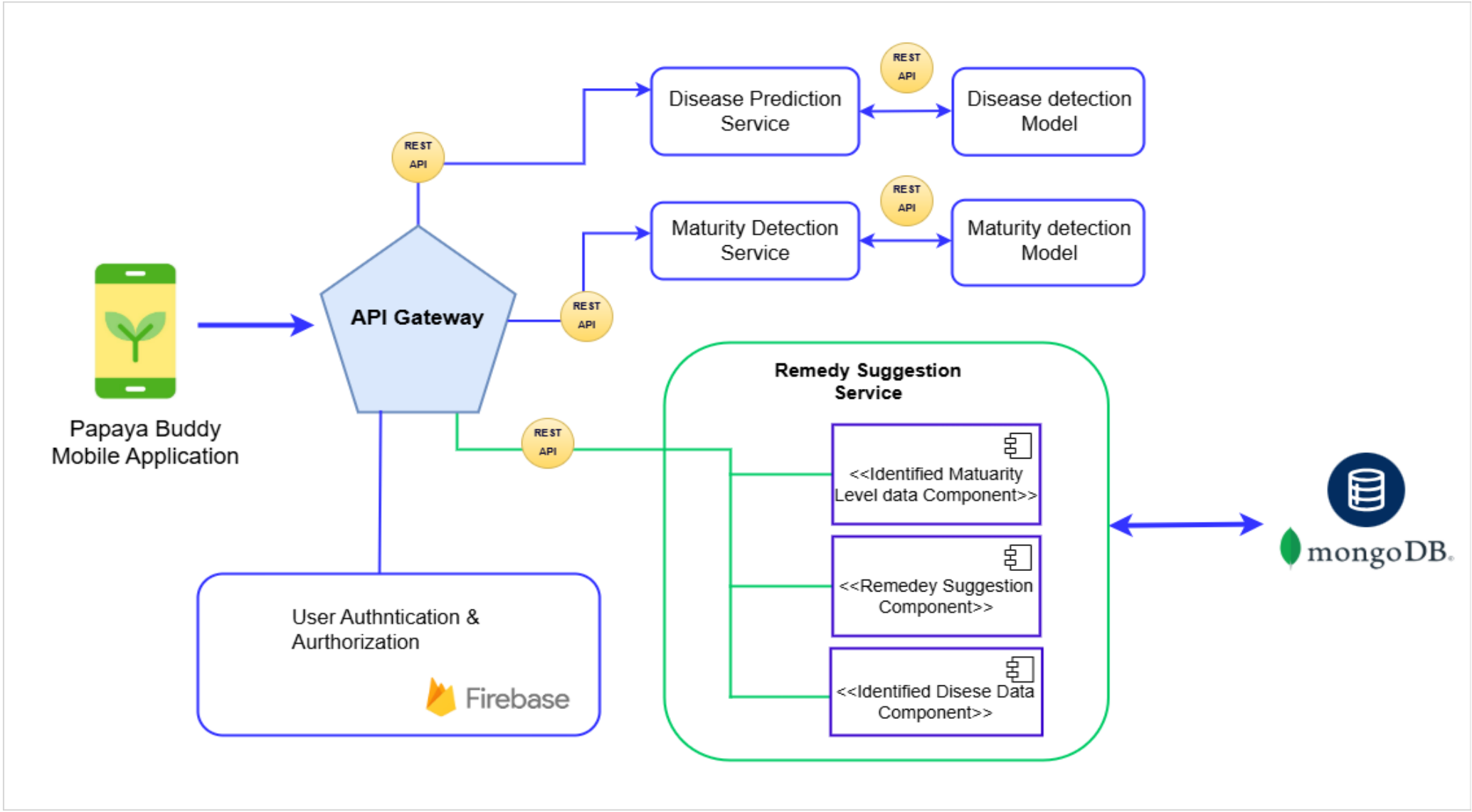
- **Single Point of Failure**
  If one component fails, the entire application may become unresponsive.

- **Technology Flexibility**
  Each microservice can be developed using the most suitable programming language
  And framework, allowing teams to work independently without constraints.

Dockerization Issue



Microservices Architecture



System Architecture

📱 Flutter Mobile App – User-friendly interface for disease detection, ripeness classification, and remedy suggestions.

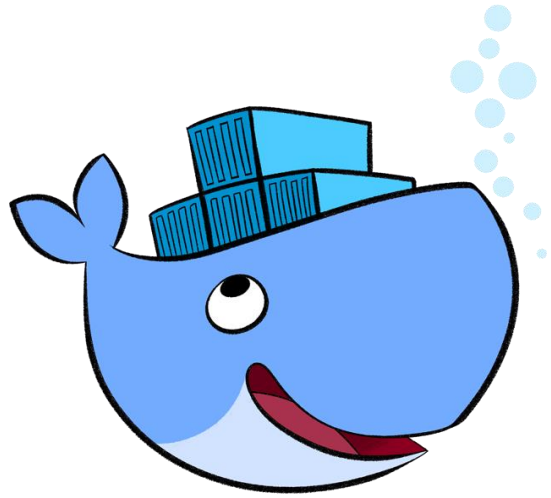
🔐 Firebase (User Management & Authentication) – Handles user authentication, role-based access control, and real-time sync.

🗂️ Node.js (Backend Service) – Fetches and stores remedy data, disease information, and maturity data.

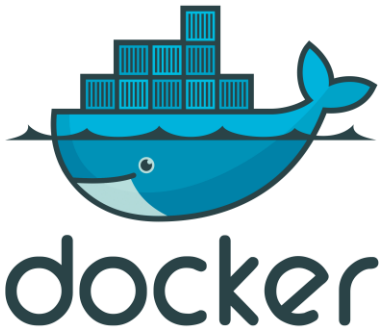
🤖 Fast API (Model Hosting) – Load CNN models and disease prediction

🌐 Kong API Gateway – Provides authentication, authorization, and load balancing, ensuring security and efficient request handling.

🚀 **Microservices with Firebase ensure a secure, scalable, and high-performance system!**



Used Technologies

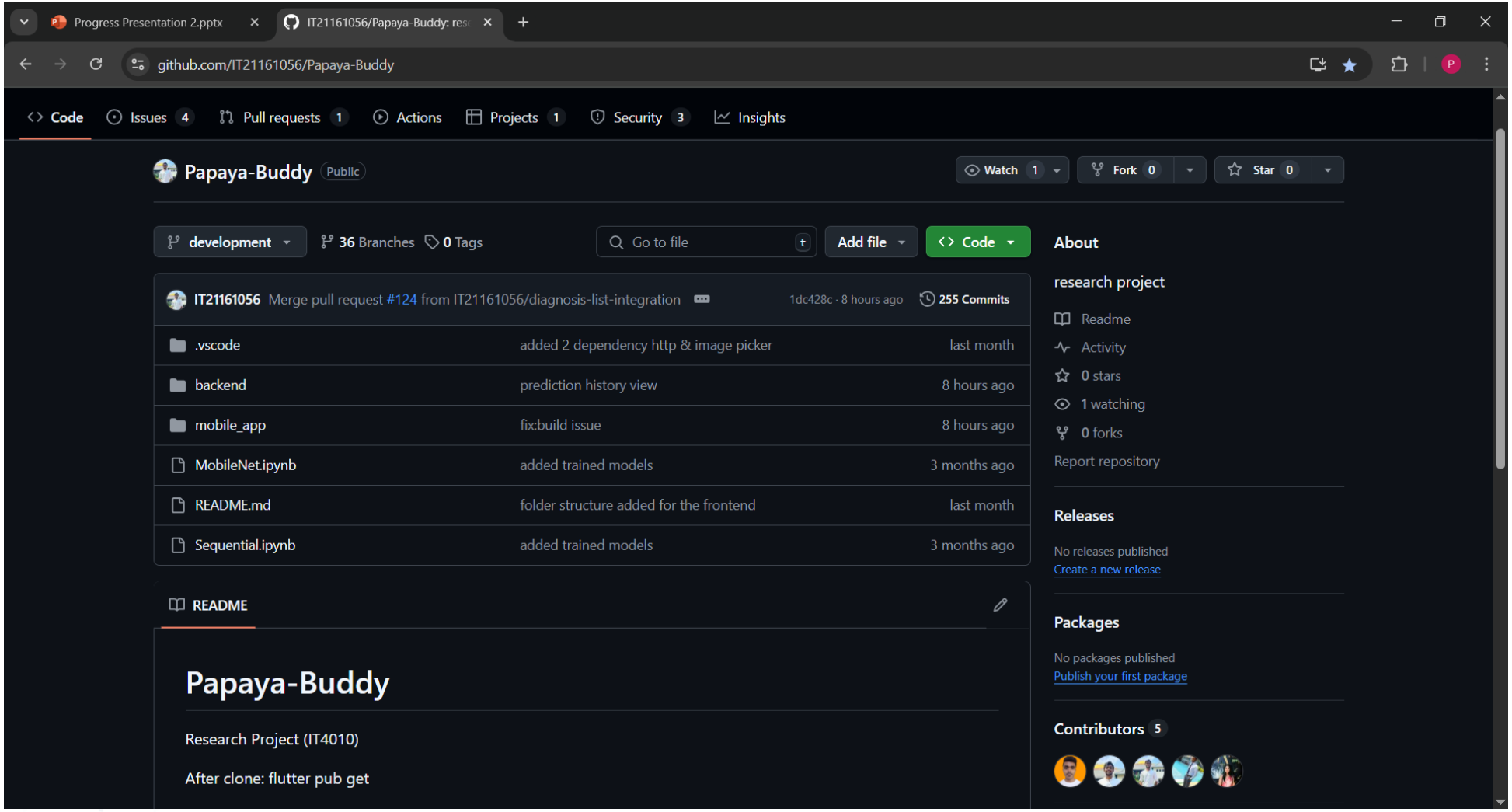


Coding Best Practices

- **Write Clean & Readable Code**
  Maintain clarity and structure for better collaboration.
- **Modular Design**
  Break code into smaller, reusable functions for efficiency.
- **Meaningful Documentation**
  Add concise comments to enhance code understanding.
- **Version Control & Agile Workflow**
  Use GitHub with a proper branching strategy and a Kanban board for streamlined project management.
- **Continuous Improvement**
  Stay updated and refine skills through learning.
- **Follow DRY Principle**
  Avoid redundancy by reusing code effectively.

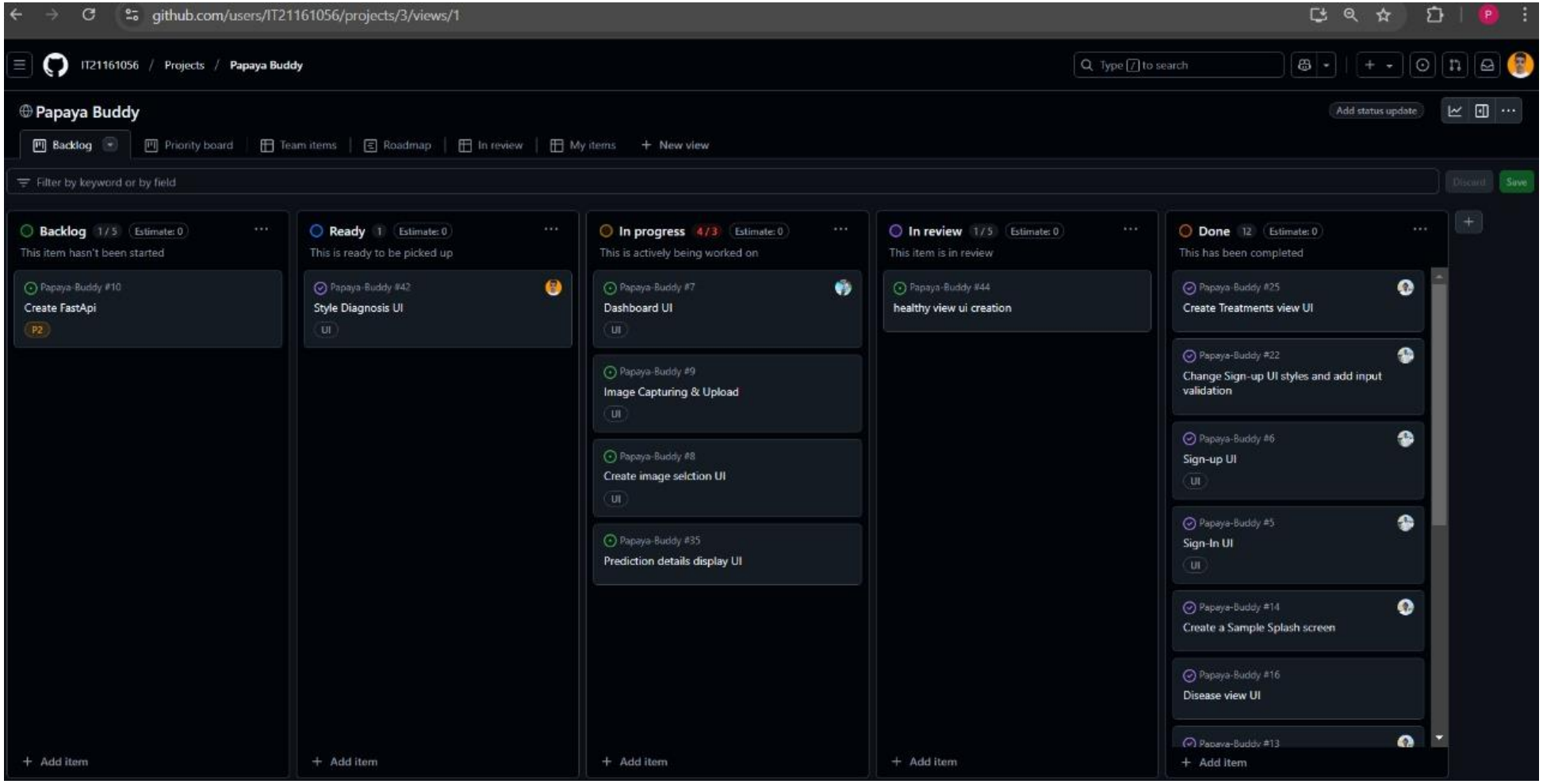
Git Hub Repository

Code and resources for the project, available for collaboration and implementation



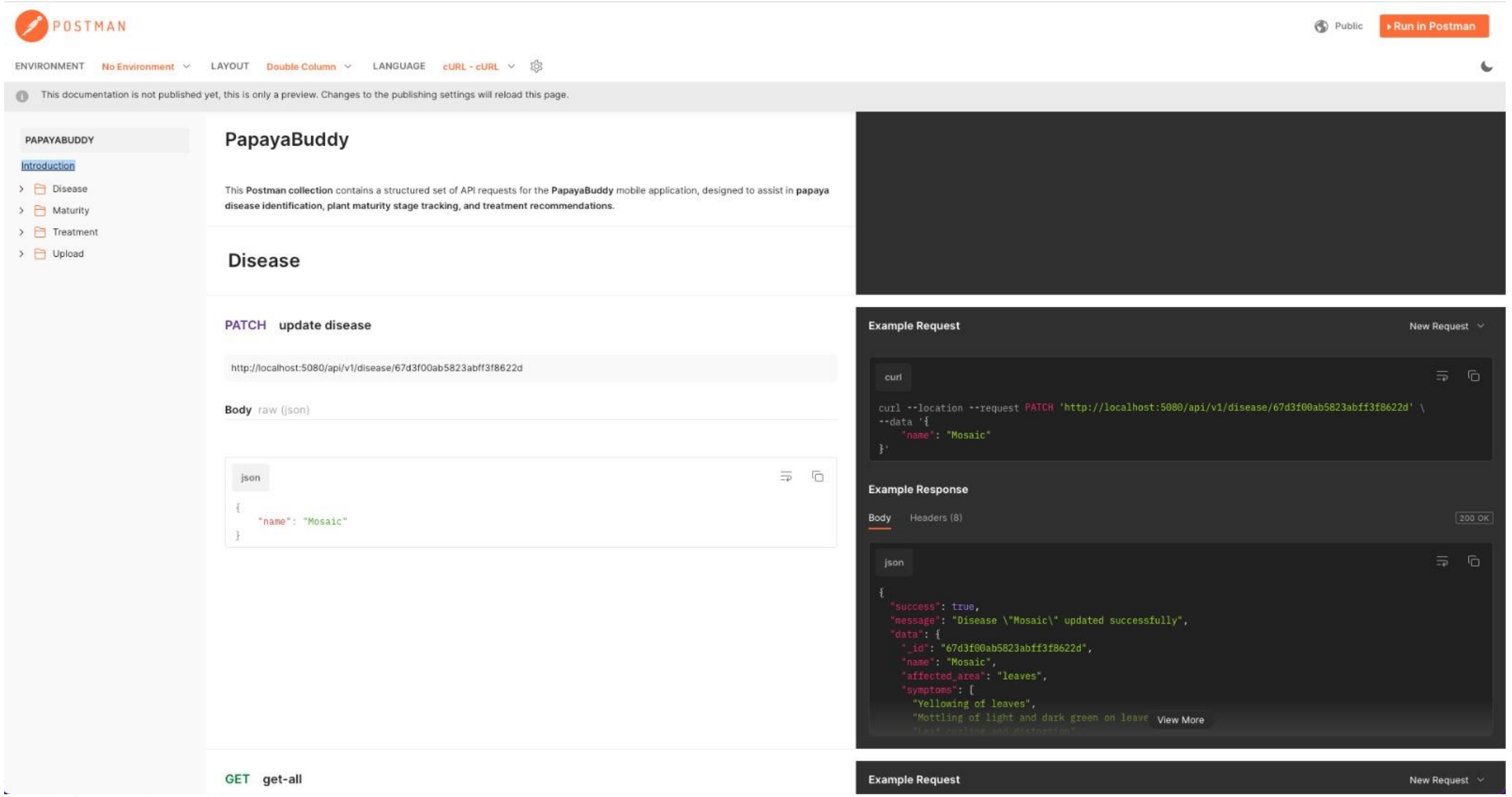
Kanban Dashboard

For project management to flow smoothly and meet the deadline



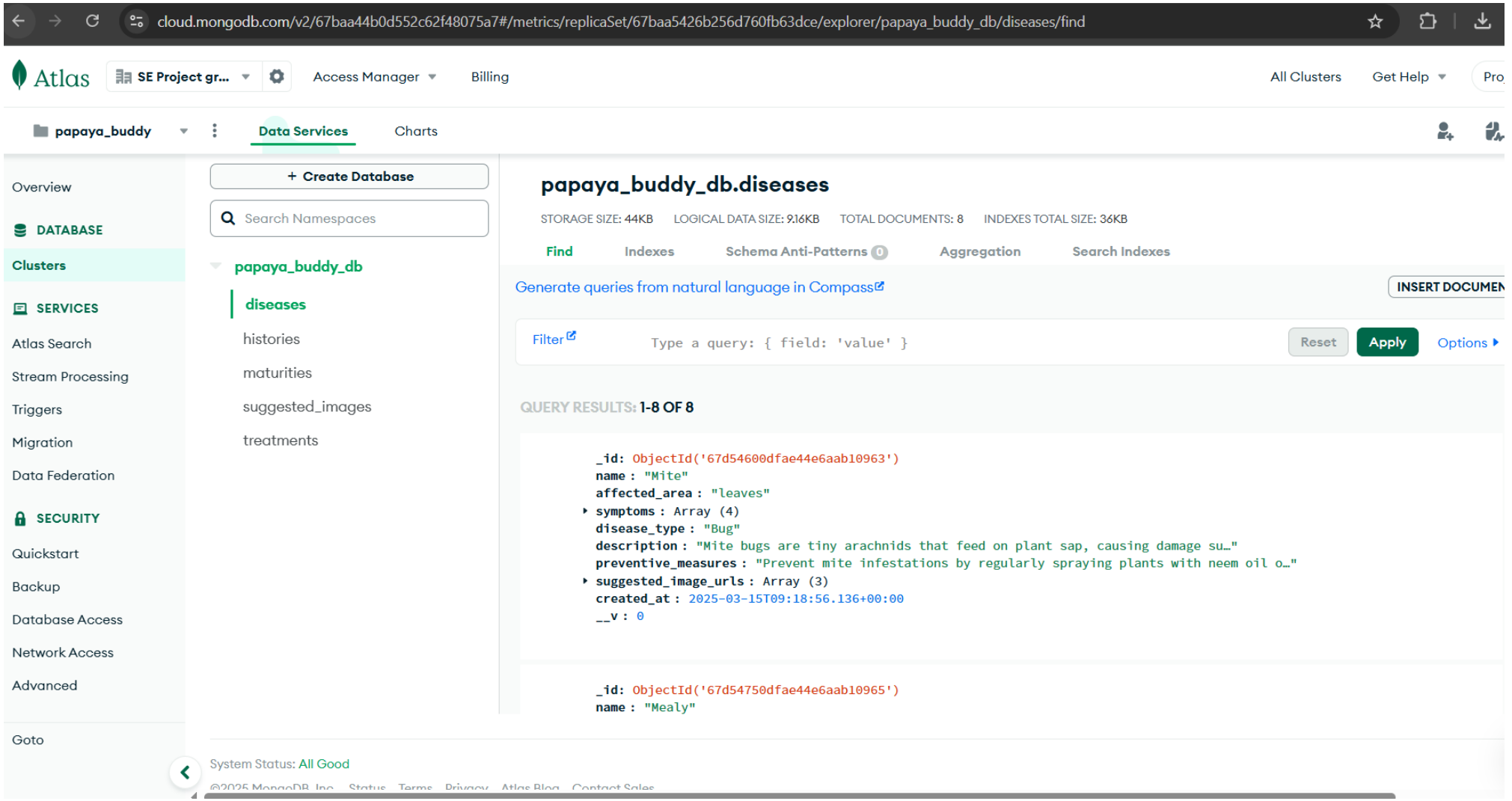
Postman Collection

Postman collection for API testing and documentation.



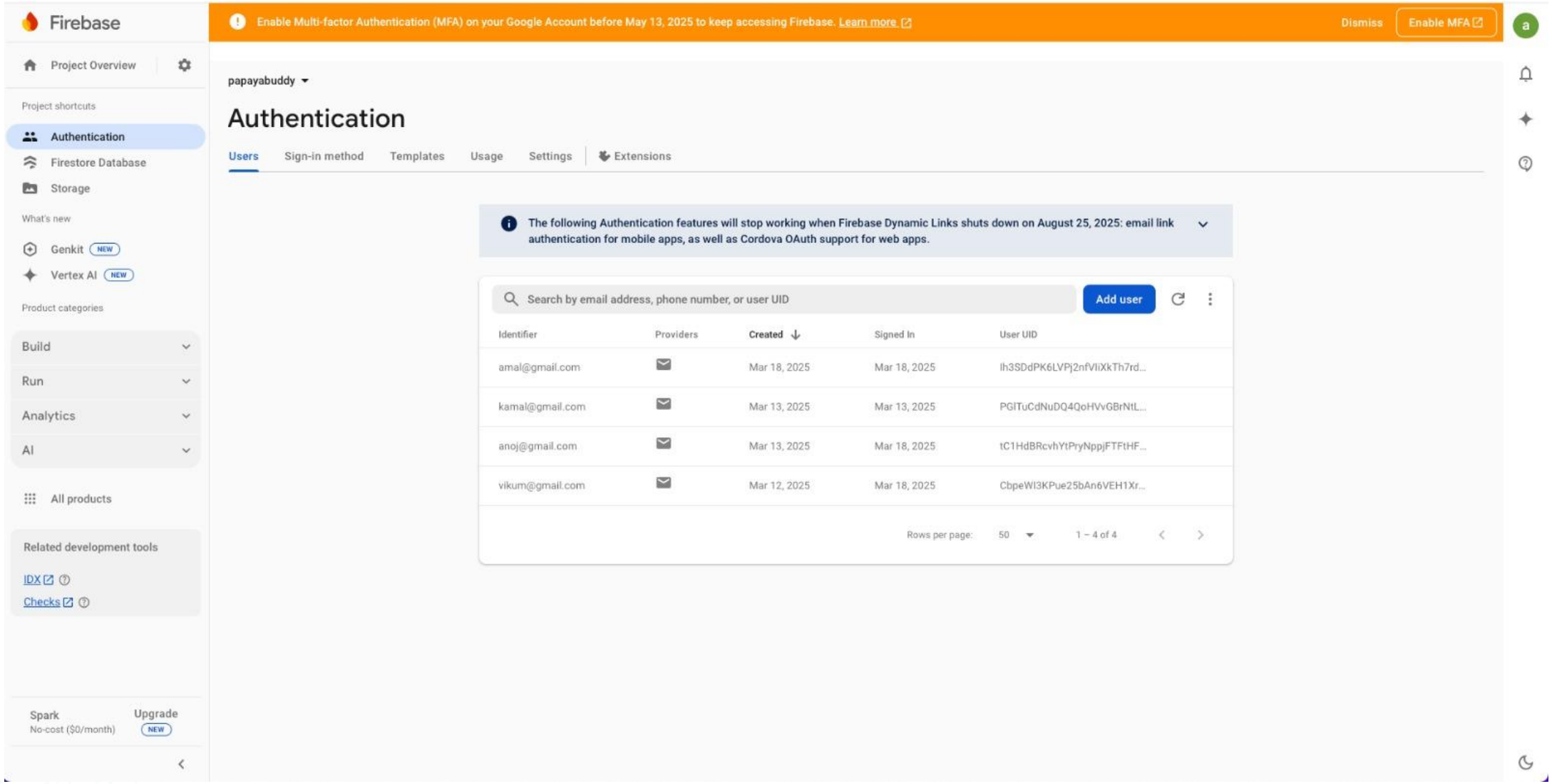
MongoDB Schema

MongoDB database setup and schema for the project.

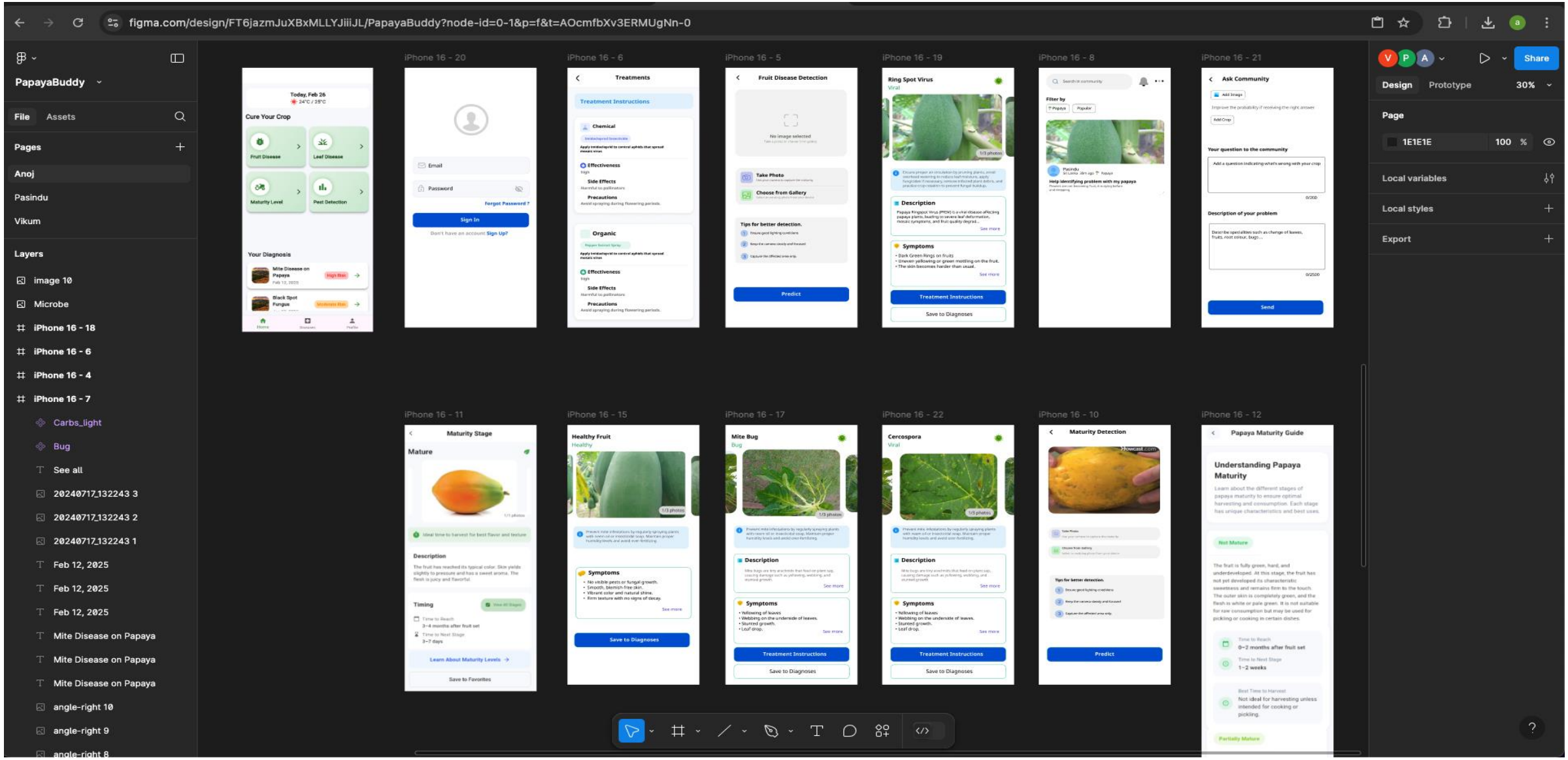


Firebase

Firebase user management for authentication and access control



Figma UI Creation



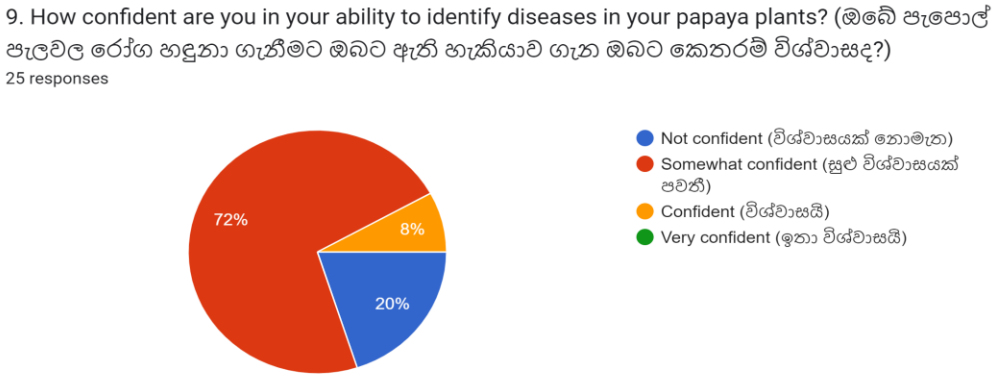
IT21161056 | Peiris M. M. A. E

Software Engineering

"How can we revolutionize papaya farming in Sri Lanka?"

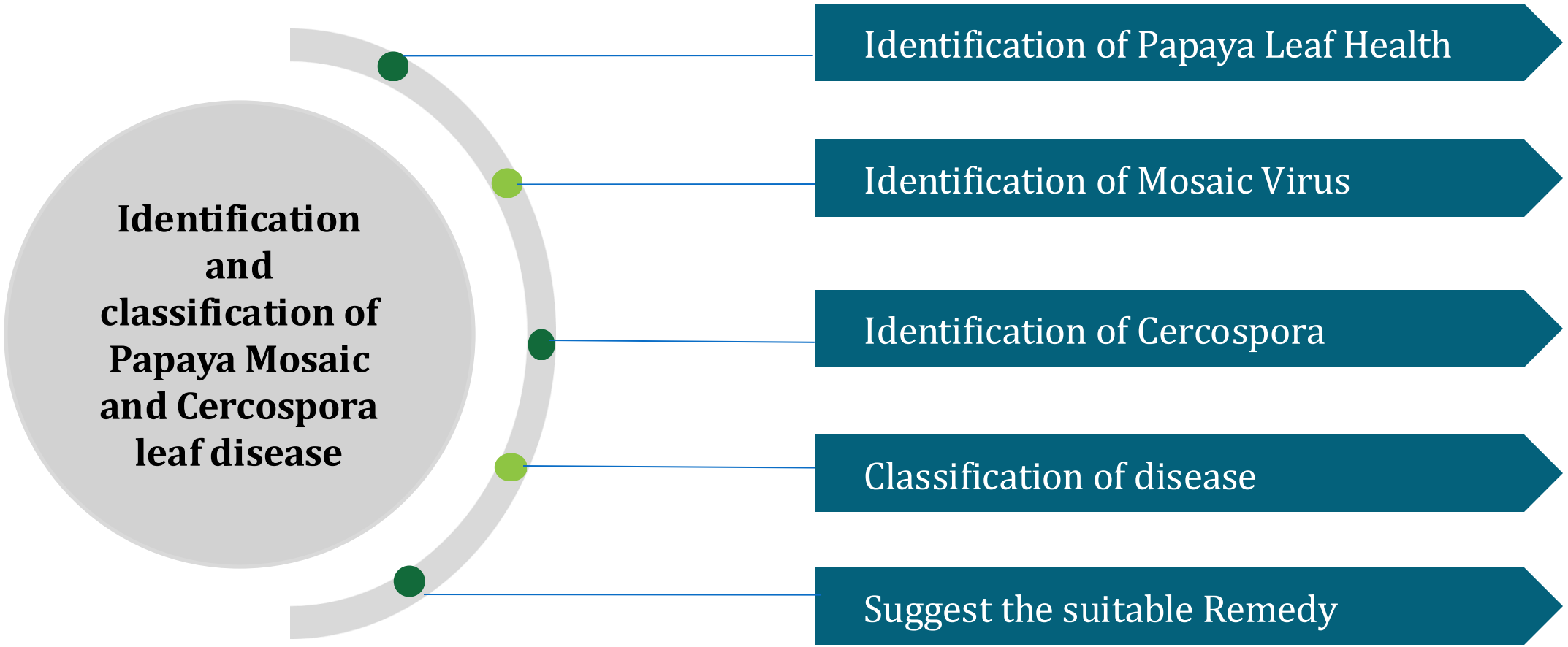
Leveraging cutting-edge mobile technology and machine learning,
to precisely detect and manage:

- Papaya Mosaic Virus
- Cercospora Leaf Spot

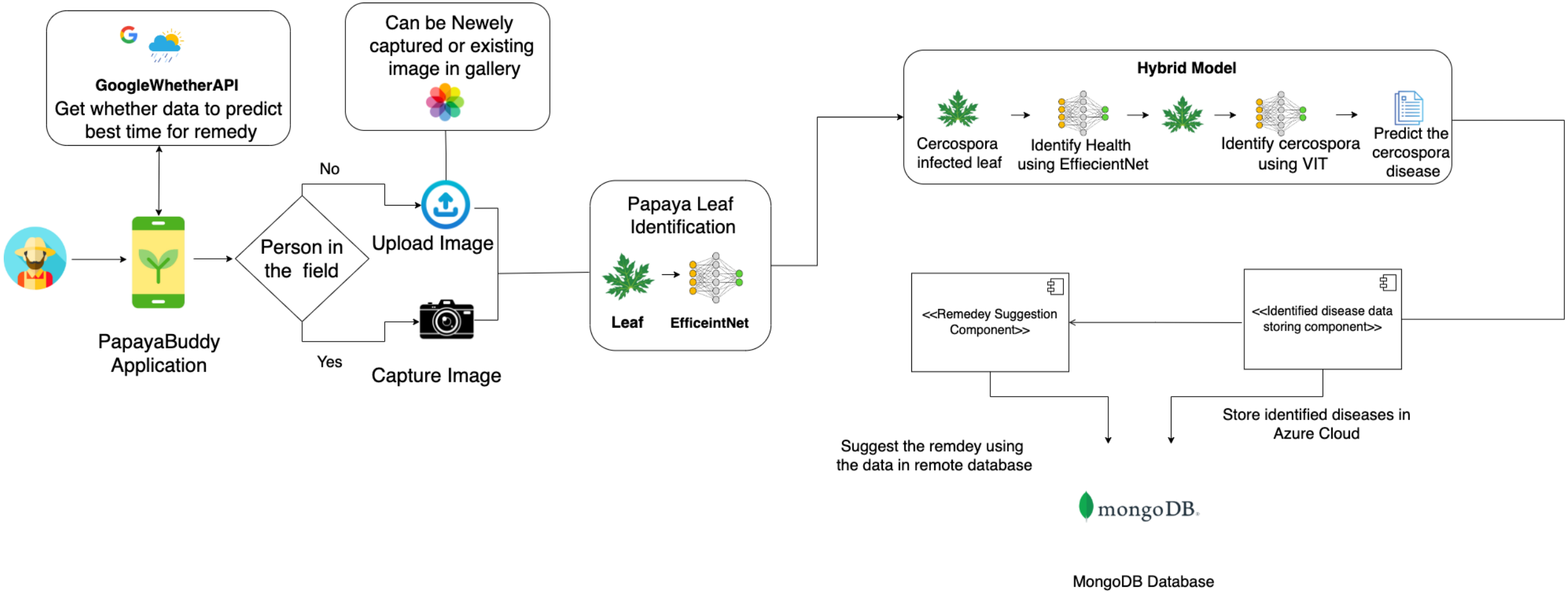


Can this innovation empower rural farmers with real-time, accurate disease diagnosis and effective management strategies?

Specific and Sub Objectives



System Diagram

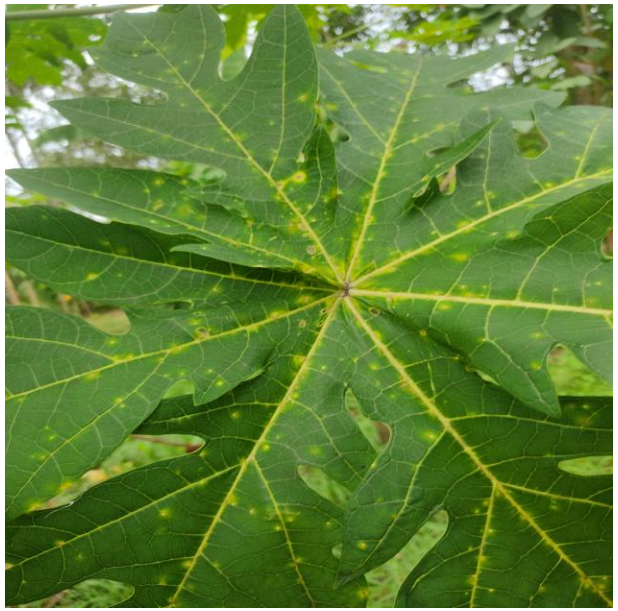


Evidence of Completion

- Data Collection & Pre-Processing
- Identify Cercospora & Mosaic Virus - ViTs
- Data Augmentation
- Create Hybrid Model
- Data Visualization
- Fast API for call Hybrid model
- Identify Healthy & Unhealthy Leaves - EfficientNet
- Display the results on the Mobile application

Data collection

Cercospora



Mosaic

Healthy

Class Name	Data set
Healthy	339 Images
Cercospora	429 Images
Mosaic	250 Images

Data Augmentation & Preprocessing

- Preprocessed to a size of 224 x 224, making it best suited for model training.

```
IMG_SIZE = 224
transform = transforms.Compose([
    transforms.RandomResizedCrop(IMG_SIZE, scale=(0.8, 1.0), ratio=(0.75, 1.33),
                                 interpolation=transforms.InterpolationMode.BICUBIC),  # Crop the best area
    transforms.RandomHorizontalFlip(p=0.5),  # Flip with 50% probability
    transforms.RandomRotation(degrees=5, expand=False),  # Restrict rotation
    transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.02),  # Minor color adjustments
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Standard normalization
])
```

Hybrid Inference

Model Name	google/vit-base-patch16-224
Pretrained	Yes
Data set Size	669 Images
Classes	2
Batch Size	Batch size = 32
Image Size	224×224
No of Parameters	86.6M params
Model Size	343 MB ~
Used Memory/ RAM	Tesla T4 GPU
Used Environment	Google Collab

+

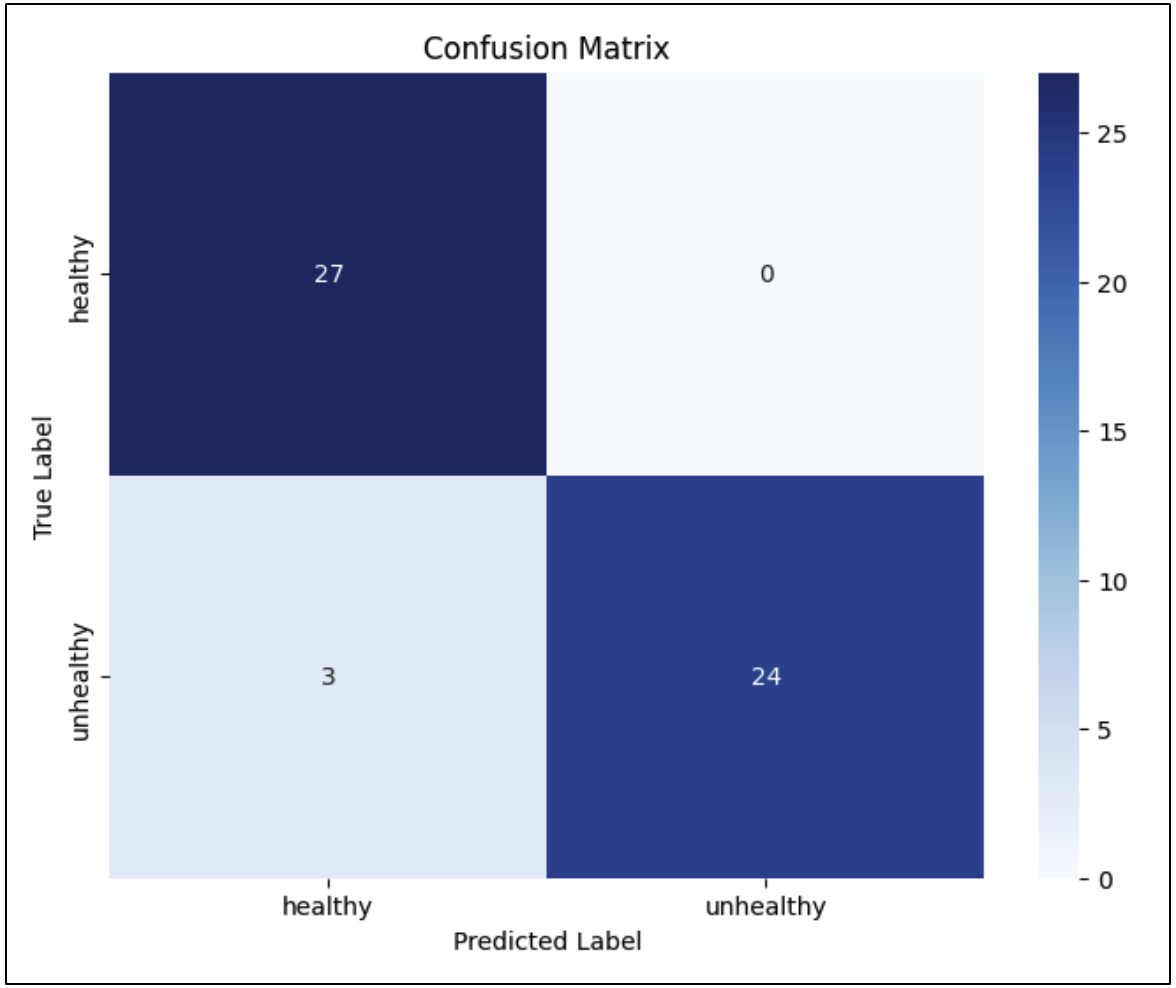
Combine Models

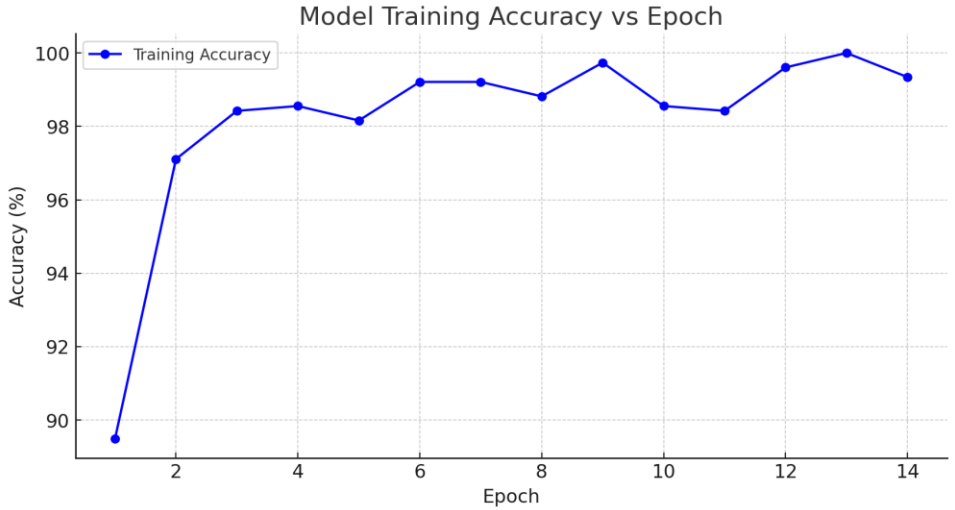
Model Name	efficientnet_b0
Pretrained	Yes
Data set Size	761 Images
Classes	2
Batch Size	Batch size = 32
Image Size	224×224
No of Parameters	5.33M params
Model Size	16 MB ~
Used Memory/ RAM	Tesla T4 GPU
Used Environment	Google Collab

Hybrid Inference

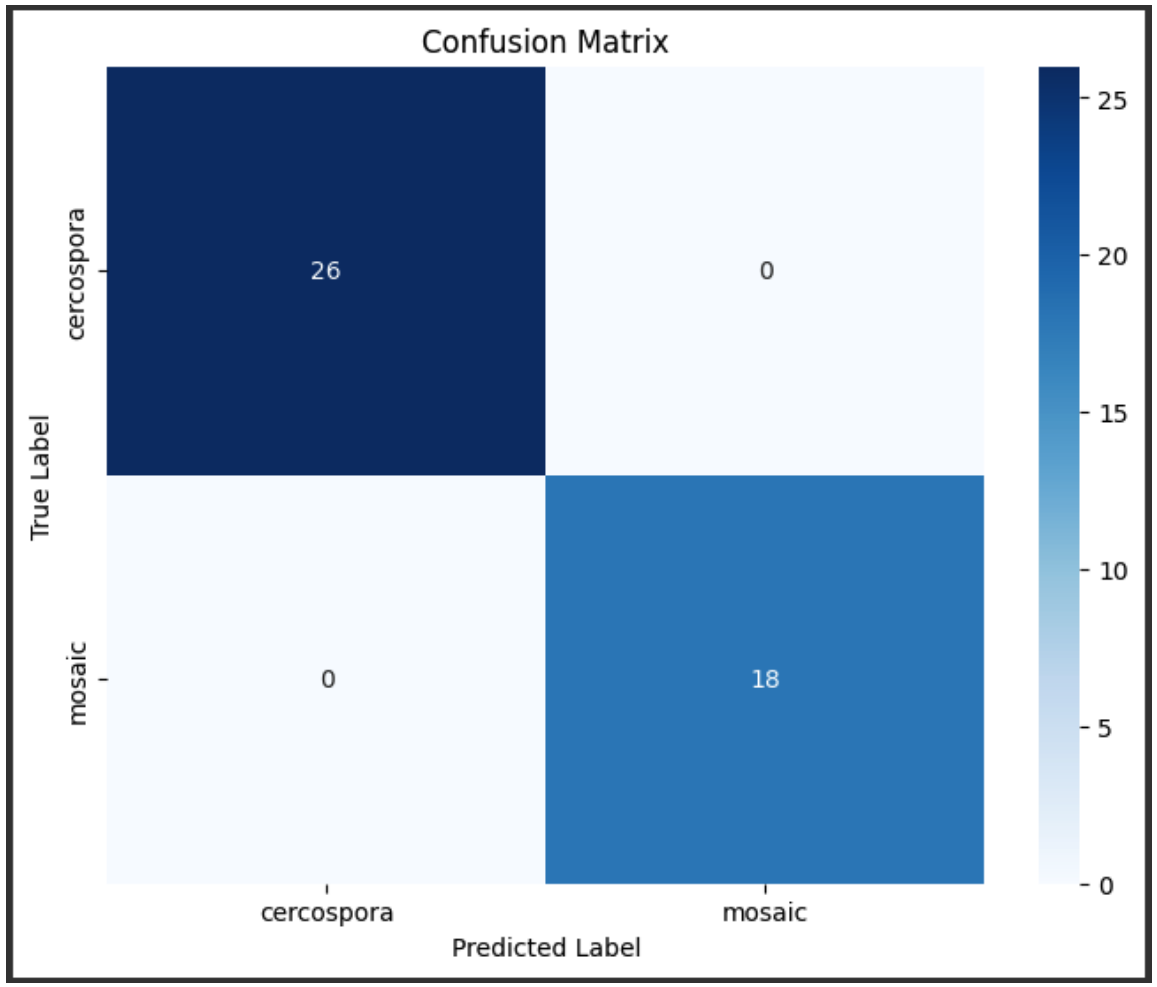
```python
class HybridModel(nn.Module):
    def __init__(self, efficientnet_model, vit_model):
        super(HybridModel, self).__init__()
        self.efficientnet = efficientnet_model
        self.vit = vit_model

    def forward(self, x):
        eff_output = self.efficientnet(x)
        _, eff_pred = torch.max(eff_output, 1)

        if eff_pred.item() == 1:
            vit_output = self.vit(x).logits
            _, vit_pred = torch.max(vit_output, 1)
            return eff_pred.item(), vit_pred.item()
        else:
            return eff_pred.item(), None  # Healthy, so no disease classification
```

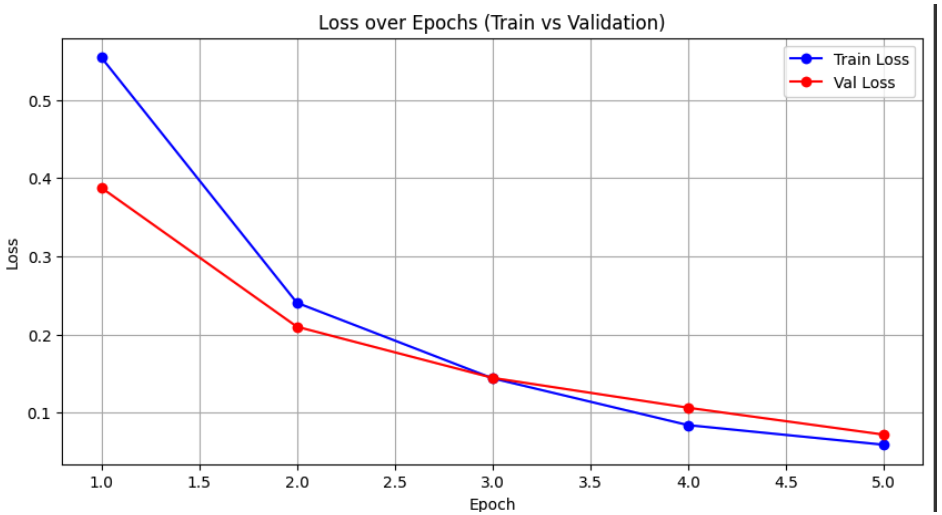
EfficientNet Performance

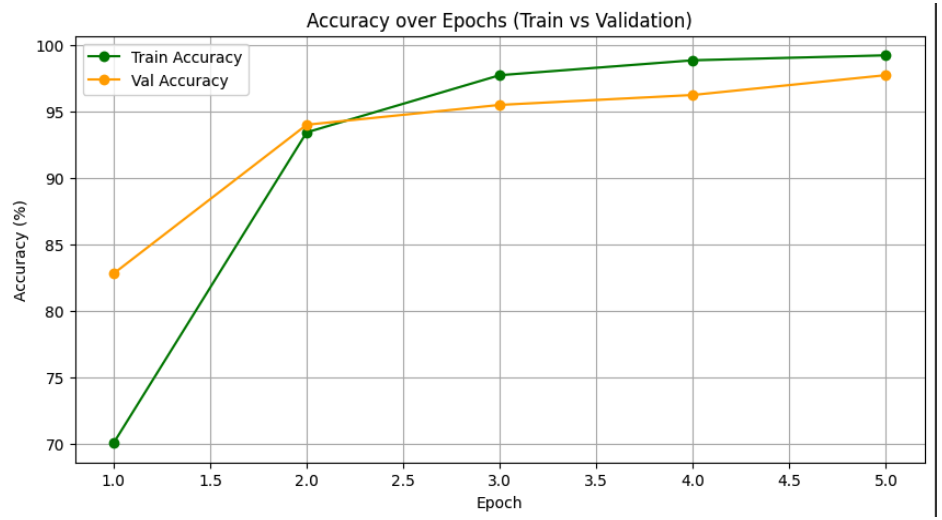




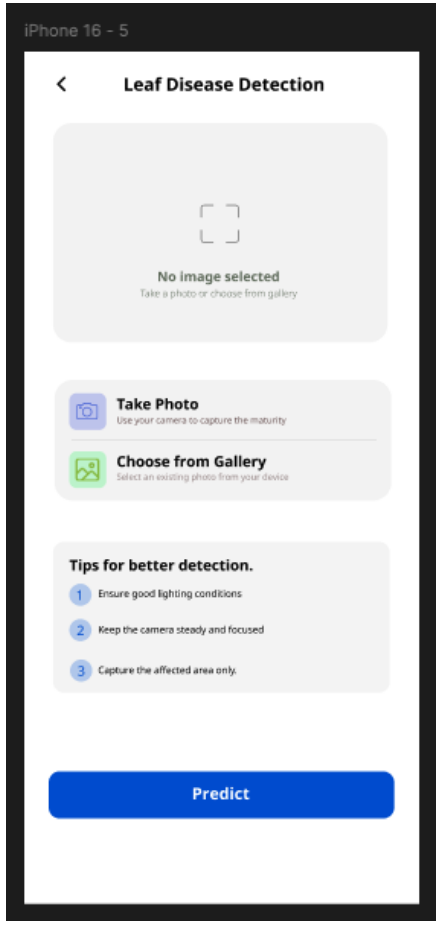
Vit Model Performance

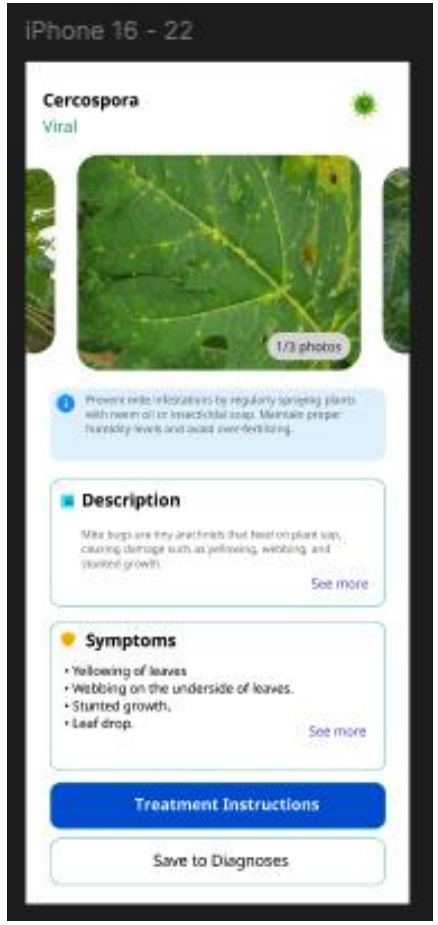


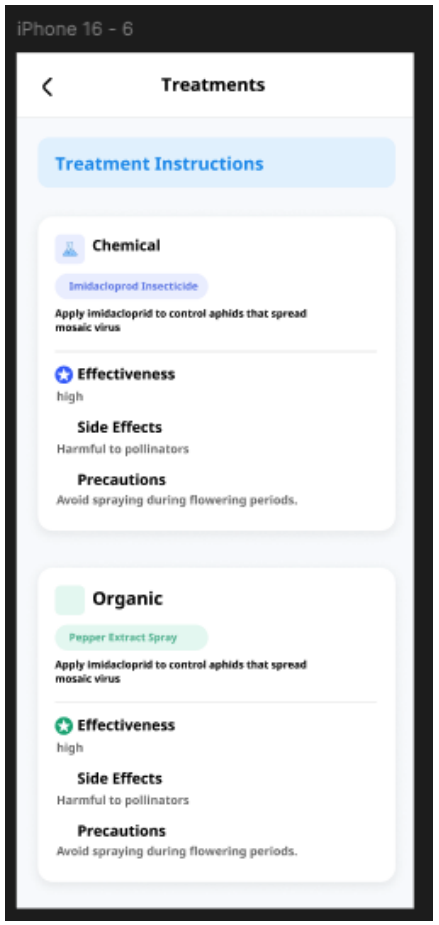


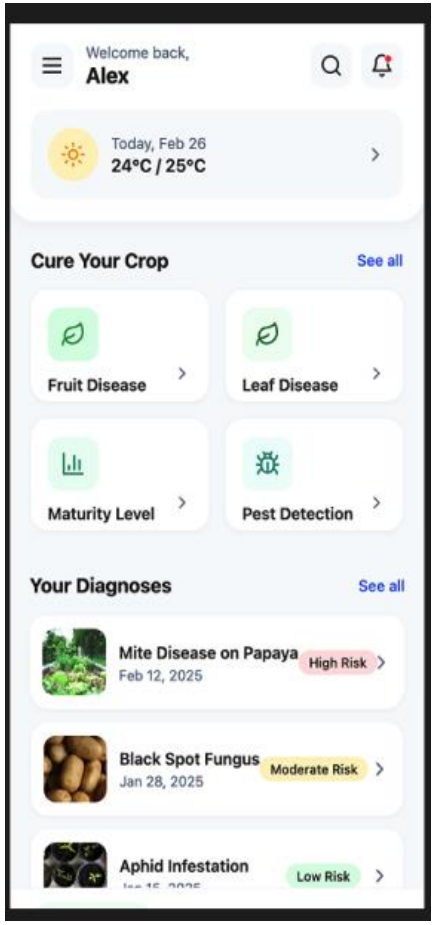


Mobile Application Mockups

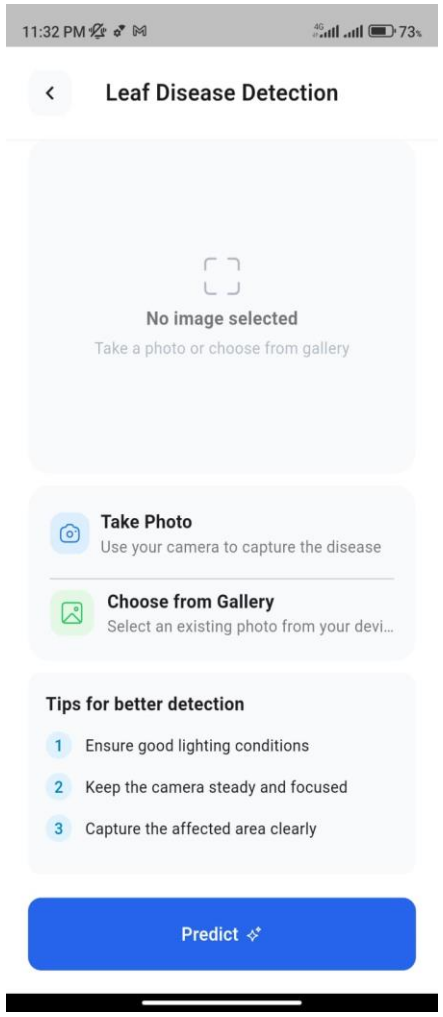


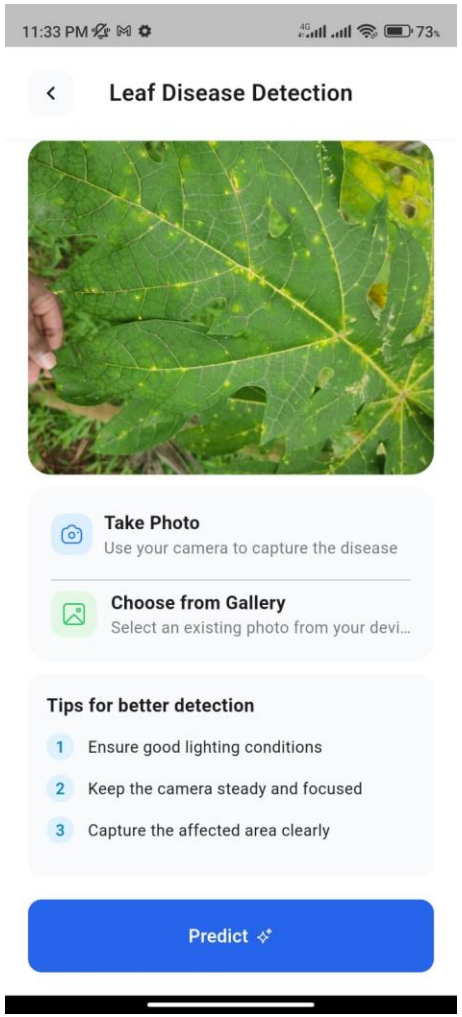


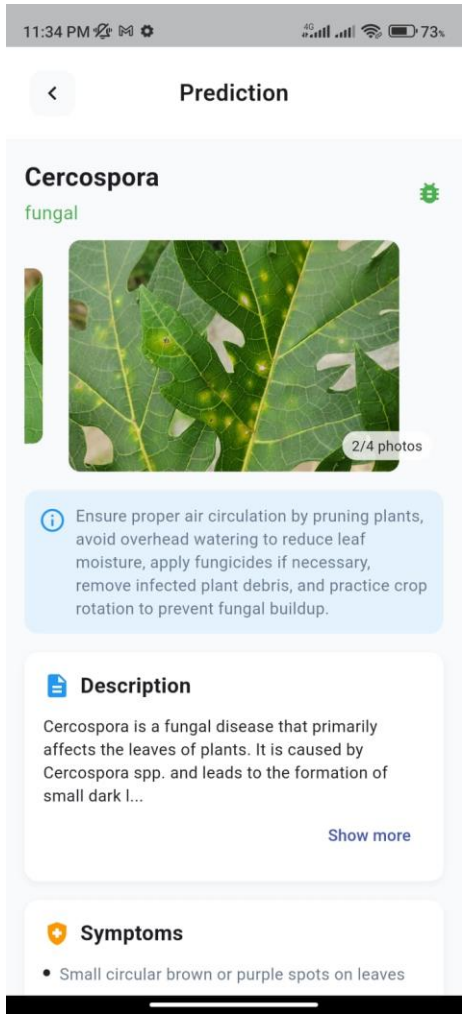


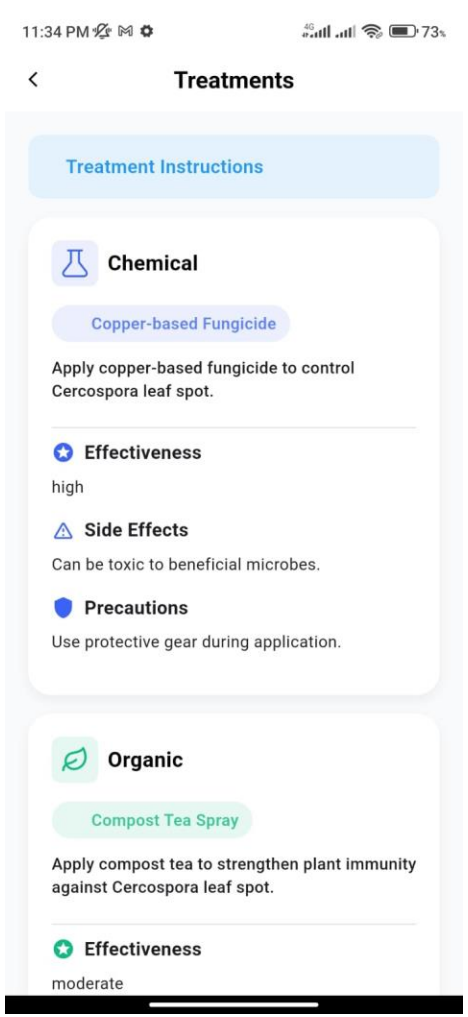


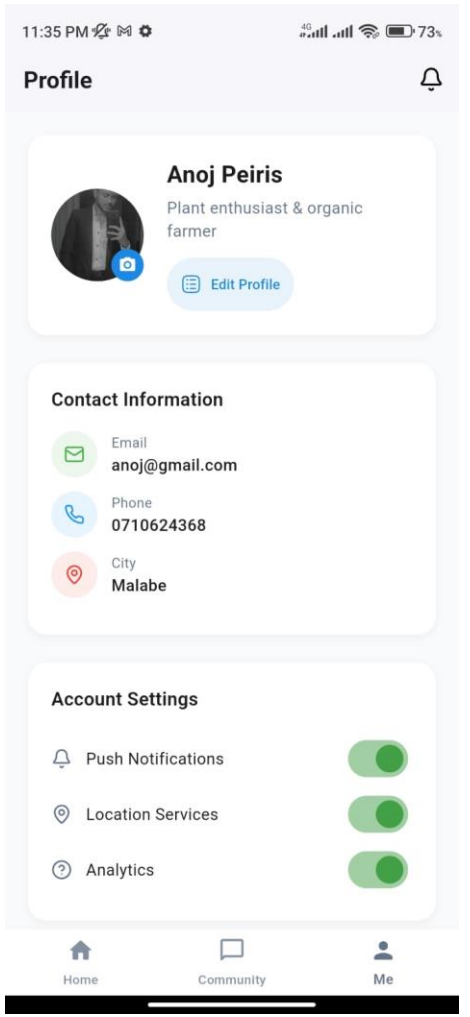
Mobile Application UIs







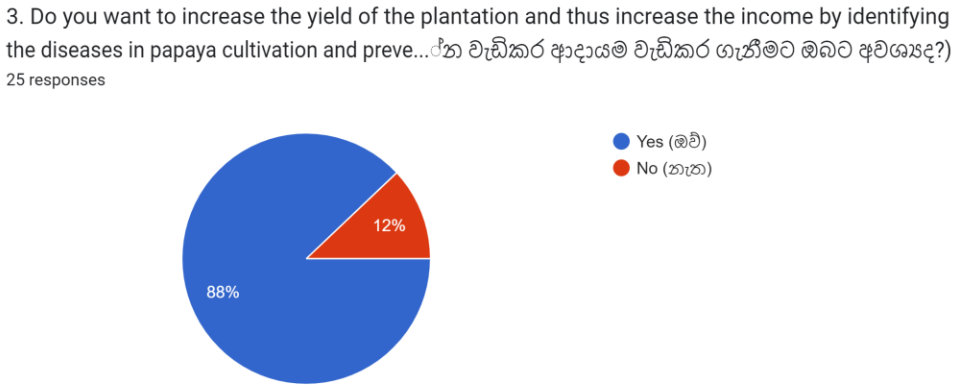


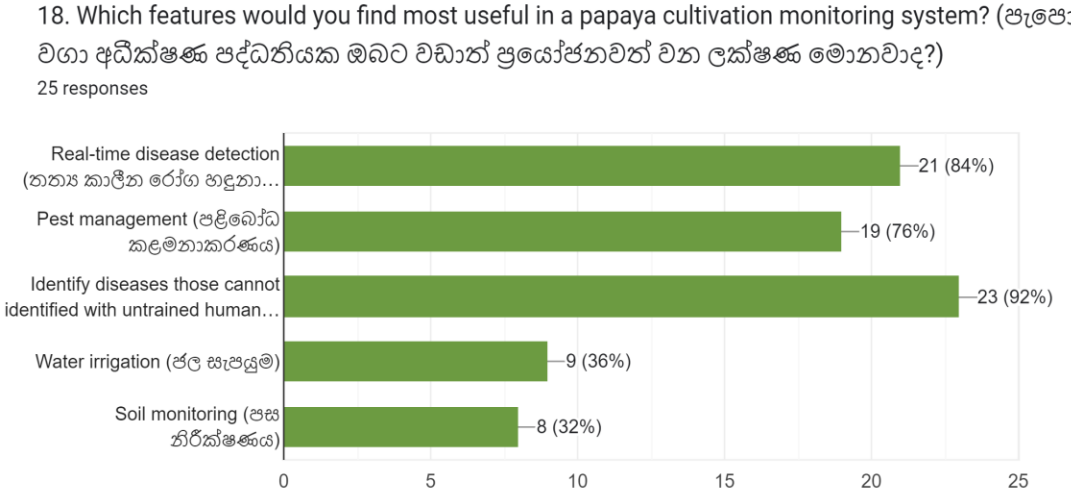


IT21160066 | K.K.P. Perakum

Software Engineering

Research Question

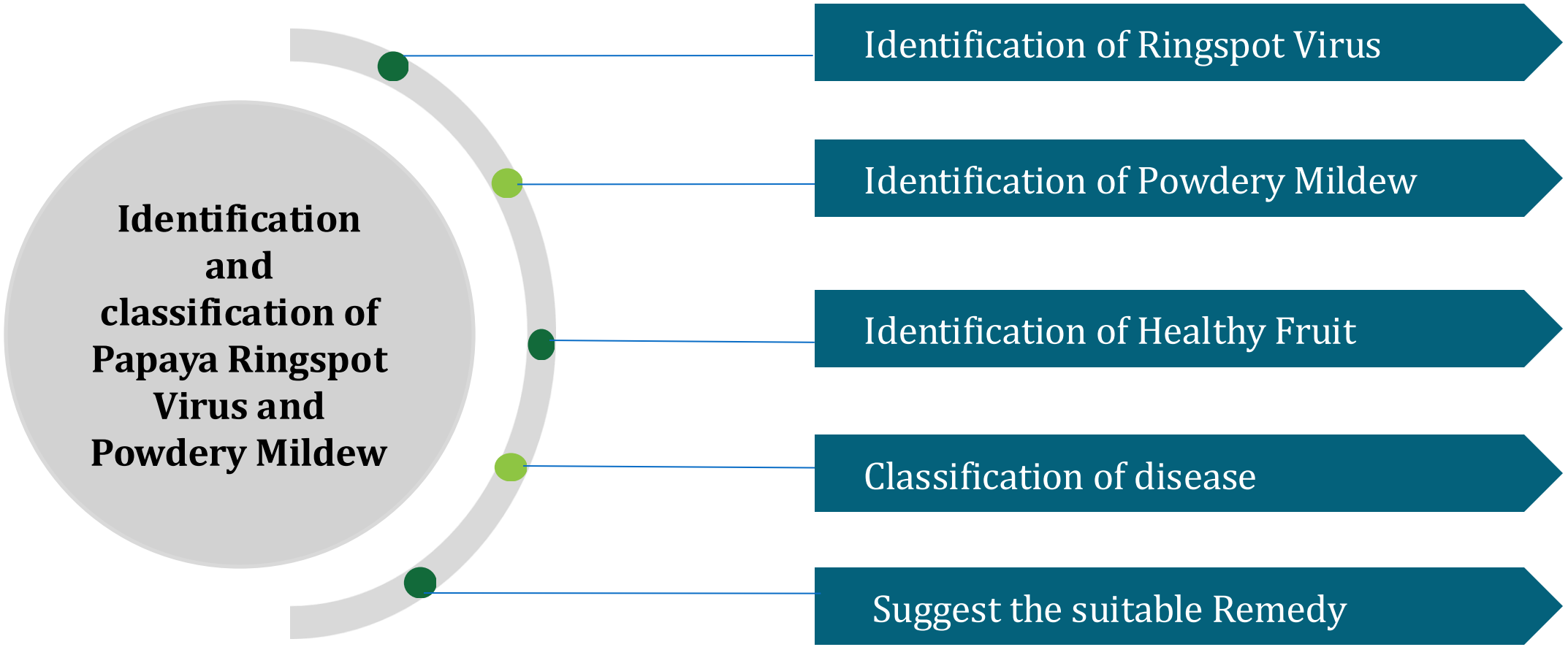




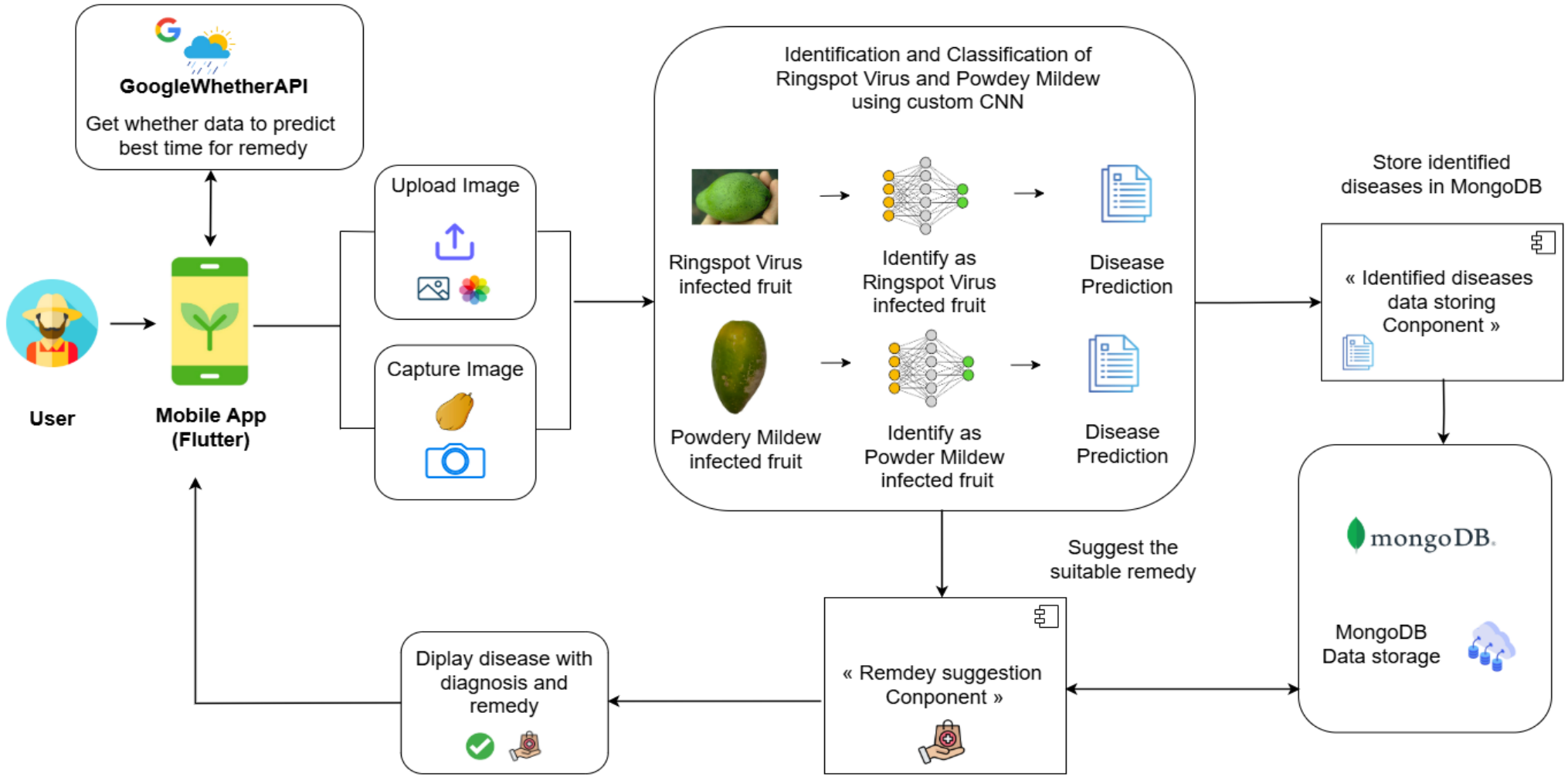
How to identify Papaya Ringspot Virus?

How to identify Powdery Mildew?

Specific and Sub Objectives



System Diagram



Evidence of completion

- Data Collection
- Data pre-processing
- Data visualization
- Train YOLOv5 model
- Built a custom CNN sequential model
- Identify Ring Spot, Powdery Mildew & Healthy Fruit – custom CNN
- Host the finalized model on Fast API
- Create the Mobile Application

Data Collection and Pre-Processing

Class Name	Data Set
Ring Spot	500 Images
Powdery Mildew	500 Images
Healthy Fruit	500 Images

Ring Spot

Powdery Mildew

Healthy Fruit

Data collection and pre-processing

CNN_papaya_RePr.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
[ ] def prepare_dataset(data_dir, img_height=256, img_width=256, batch_size=32):
        # Load the full dataset
        dataset = tf.keras.utils.image_dataset_from_directory(
            data_dir,
            image_size=(img_height, img_width),
            batch_size=None,
            shuffle=True,
            seed=42
        )

        # Get dataset size and class names
        dataset_size = tf.data.experimental.cardinality(dataset).numpy()
        class_names = dataset.class_names

        # Calculate split sizes
        train_size = int(0.7 * dataset_size)
        val_size = int(0.2 * dataset_size)
        test_size = dataset_size - train_size - val_size

        # Split dataset
        train_ds = dataset.take(train_size)
        temp_ds = dataset.skip(train_size)
        val_ds = temp_ds.take(val_size)
        test_ds = temp_ds.skip(val_size)

        def configure_for_performance(ds, batch_size=16, shuffle_buffer=1000):
            ds = ds.cache()
            ds = ds.shuffle(buffer_size=shuffle_buffer)
            ds = ds.batch(batch_size)
            ds = ds.prefetch(buffer_size=tf.data.AUTOTUNE)
```

CNN_papaya_RePr.ipynb

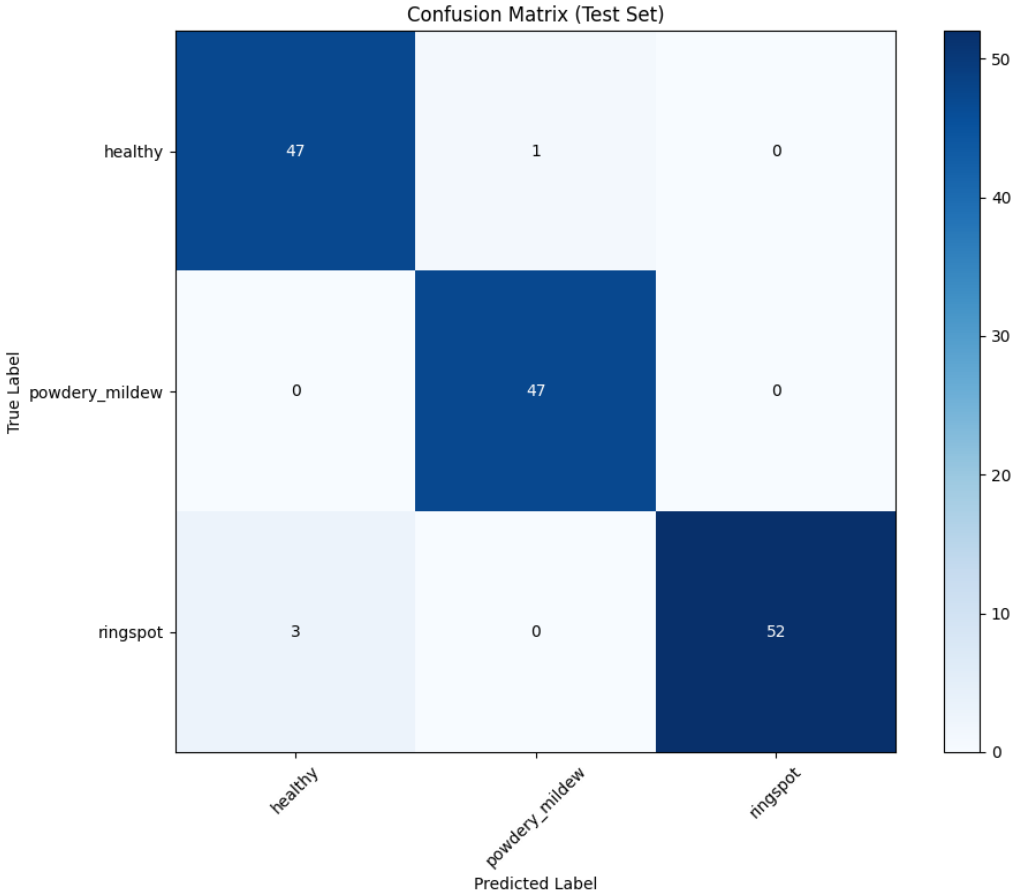
File Edit View Insert Runtime Tools Help

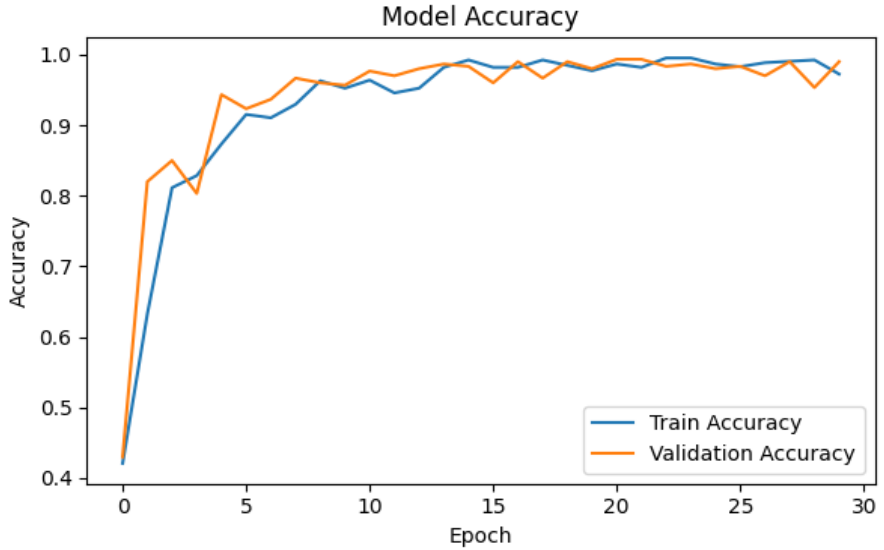
Commands + Code + Text

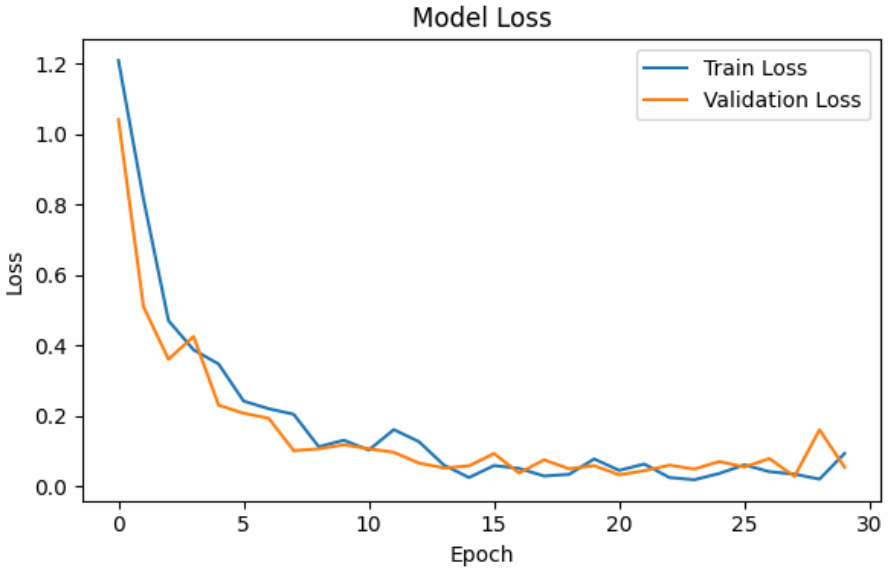
```
[ ]     # Split dataset
        train_ds = dataset.take(train_size)
        temp_ds = dataset.skip(train_size)
        val_ds = temp_ds.take(val_size)
        test_ds = temp_ds.skip(val_size)

        def configure_for_performance(ds, batch_size=16, shuffle_buffer=1000):
            ds = ds.cache()
            ds = ds.shuffle(buffer_size=shuffle_buffer)
            ds = ds.batch(batch_size)
            ds = ds.prefetch(buffer_size=tf.data.AUTOTUNE)
            return ds

        def normalize_img(image, label):
            return tf.cast(image, tf.float32) / 255., label

        # Apply configurations
        train_ds = train_ds.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
        val_ds = val_ds.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)
        test_ds = test_ds.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)

        train_ds = configure_for_performance(train_ds, batch_size)
        val_ds = configure_for_performance(val_ds, batch_size)
        test_ds = configure_for_performance(test_ds, batch_size)

        print(f"Total dataset size: {dataset_size}")
        print(f"Training samples: {train_size}")
        print(f"Validation samples: {val_size}")
        print(f"Test samples: {test_size}")

        return train_ds, val_ds, test_ds, class_names
```

Best performing model

Custom CNN





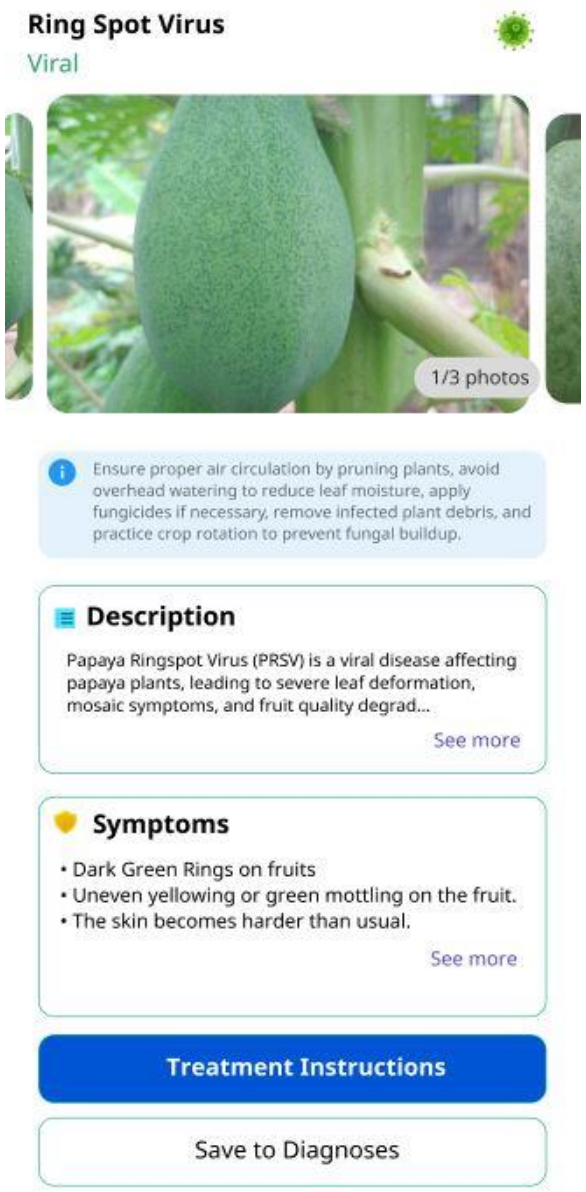


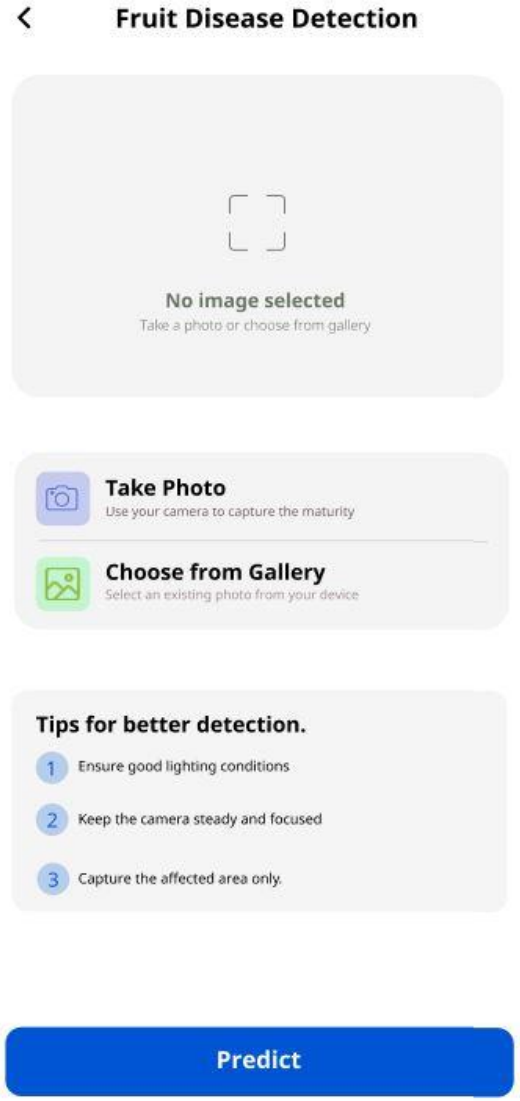
Model Comparison

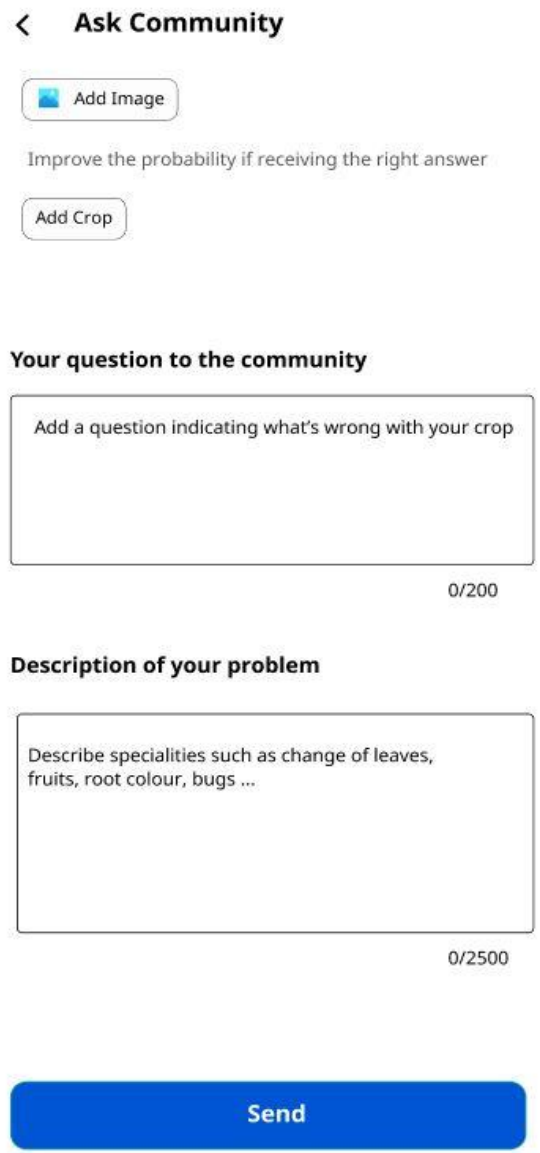
Data set Size	1500 Images for custom CNN	1500 Images for YOLOv5 1500 label files(txt)
No of Epochs	30 Epochs used for all classes	
Patience	Patience used as 20 for early stopping	
Image Size	256×256	640×640
No of Parameters	Total params: 8,445,443 (32.22 MB) Trainable params: 8,445,443 (32.22 MB) Non-trainable params: 0 (0.00 B)	Total params: 7,018,216
Model Size	32 MB ~	14 MB ~
Used Memory/ RAM	Tesla T4 GPU	
Used Environment	Google Collab	

Mobile Application Mockups

Evidence of completion

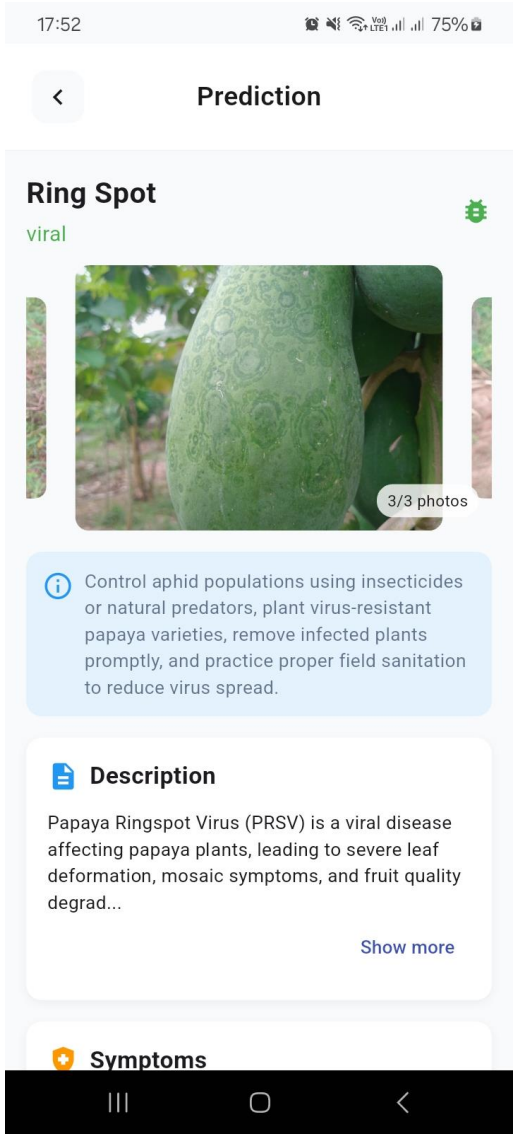


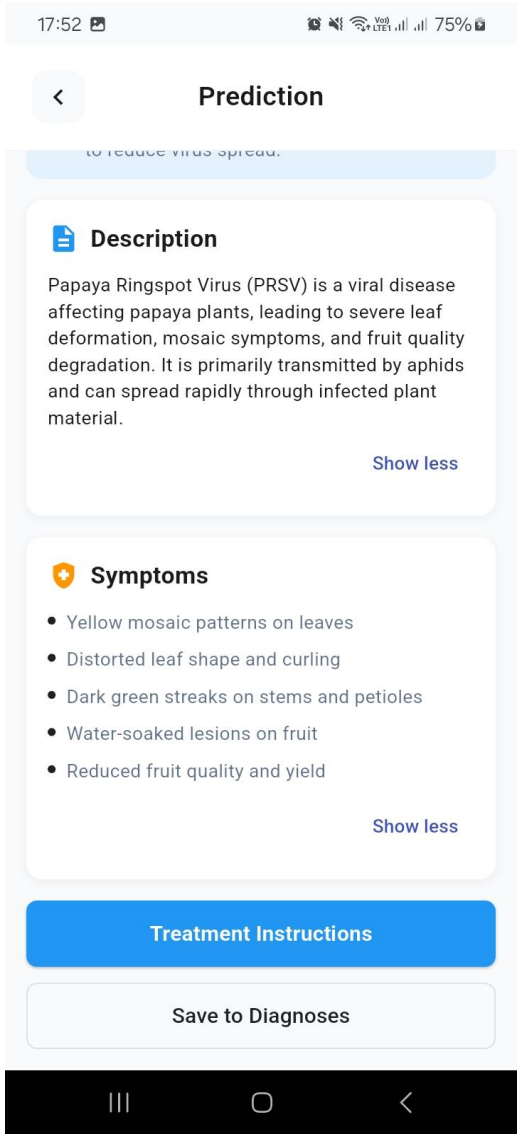


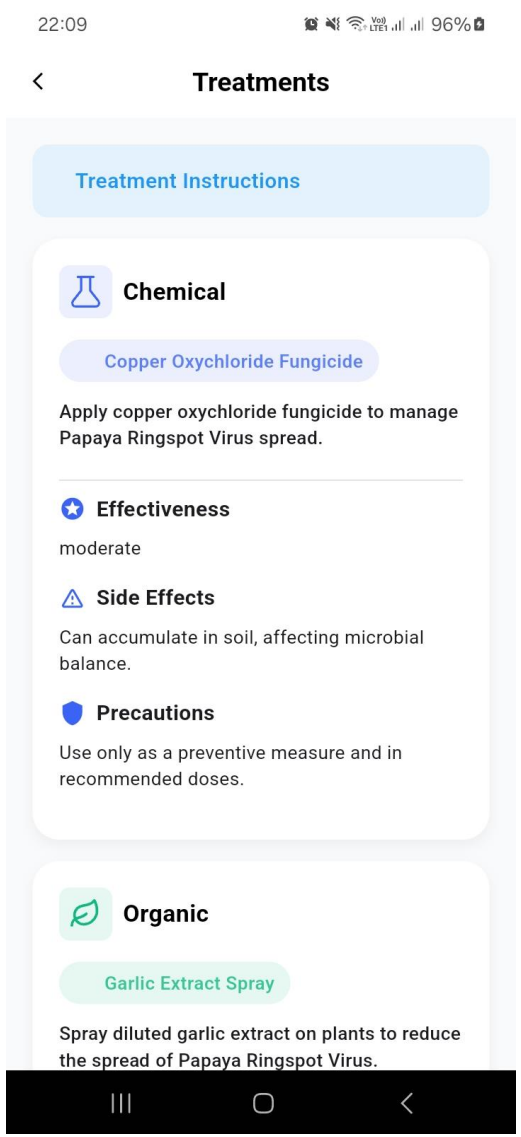


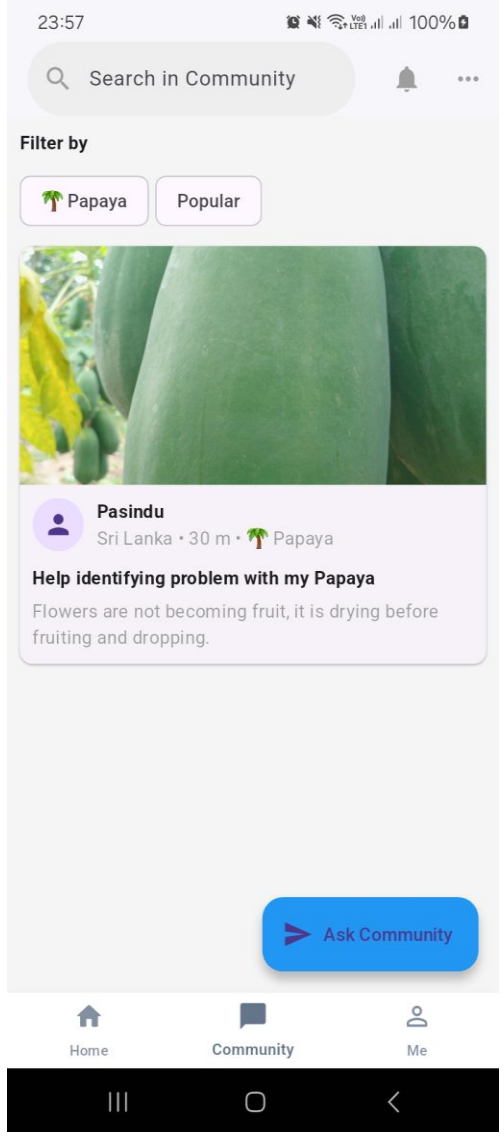
Mobile Application UIs

Evidence of completion









IT21161810 | Mangchanayaka M.V.V

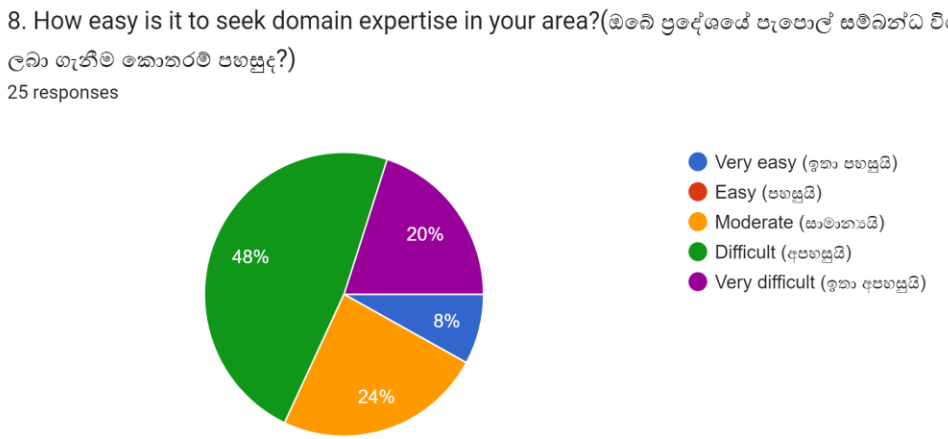
Software Engineering

Research Problem

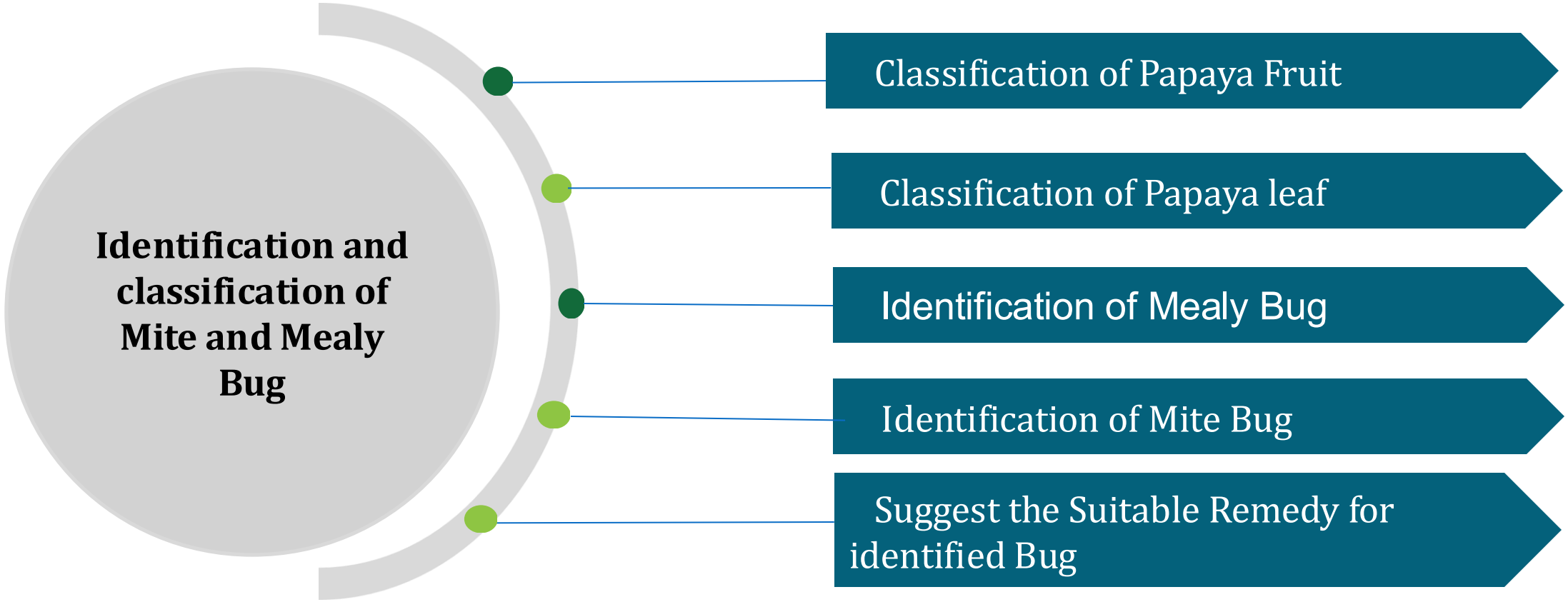
- Mite and mealy bug infestations severely damage papaya crops financial losses for farmers [2]
- Early detection and effective management of these pests are essential
- Current systems do not provide personalized and specific remedy suggestions for pest infestations
- Lack of expert knowledge identifying pest diseases
- Existing systems may not be optimized for real-time pest identifications [3]

How to identify Mite Bug?

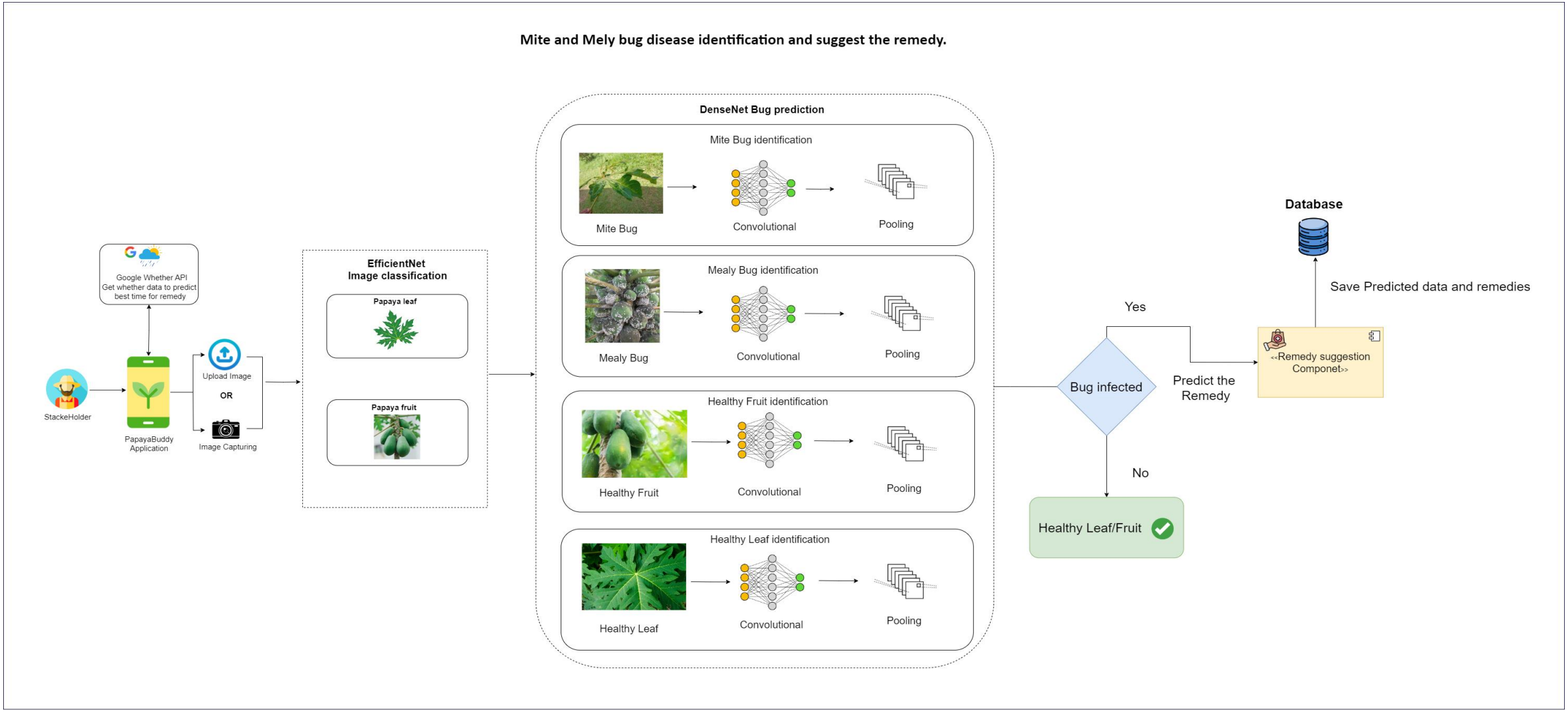
How to identify Mealy Bug?



Specific and Sub Objectives



Methodology System Diagram



Evidence of Completion

- Dashboard UI
- Dense Net CNN for identify Pest infection
- Pest Image Picker
- Data Preprocessing
- Node js API Controllers
- Fast API Server to expose CNN model as a endpoint
- Disease View & Treatment View
- Efficient Net for healthy leaf & fruit identification

Data Collection

Class Name	Data set
Healthy leaves	310 Images
Healthy Fruit	310 Images
Mealy Bug	300 Images
Mite Bug	300 Images

Healthy Leaf

Healthy Fruit

Mealy Bug

Mite Bug

CNN Model Summary

	DenseNet	EfficientNet
Modal Version	DenseNet121(Functional)	EfficientNetv2-B0(Functional)
Parameters	Total : 70,41,604 Trainable params : 41,00	Total : 60,83,538 Trainable params : 60,22,930
No of Classes	4 Healthy Fruit, Healthy Leaf, Mealy Bug, Mite Bug	2 Fruit, Leaf
Dataset Size	600 images	620 images
No of Epochs	30	30
Batch Size	32	16
Input Size	(224 x 224)	(224 x 224)
Used Environment	Jupyter notebook	Jupyter notebook

Data Augmentation & Preprocessing

- Preprocessed to a size of 224 x 224, making it best suited for model training.
- Tensorflow's ImageDataGenerator API for Data Augmentation

```python
# Data Generators with Augmentation
train_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.efficientnet.preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(preprocessing_function=tf.keras.applications.efficientnet.preprocess_input)
test_datagen = ImageDataGenerator(preprocessing_function=tf.keras.applications.efficientnet.preprocess_input)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical'
)
```

Data Preprocessing & Augmentation in EfficientNet

```python
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    'densenet_dataset/splitted_data/train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

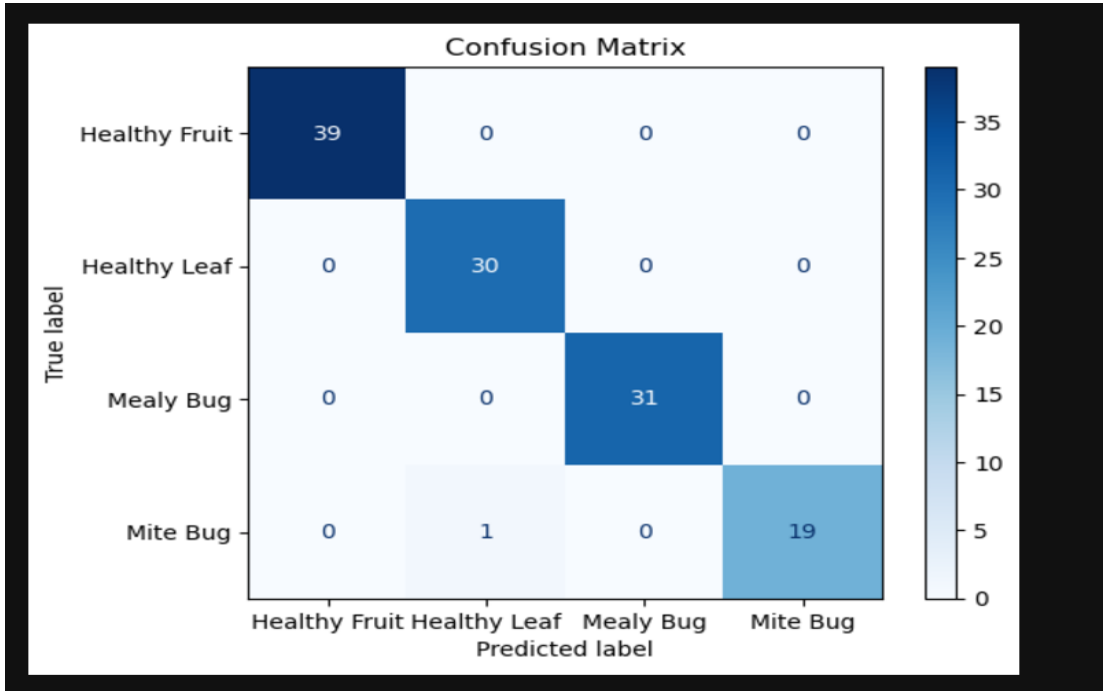
Data Preprocessing & Augmentation in DenseNet

Data Visualization for Performance Analysis

- Confusion Matrix for DenseNet

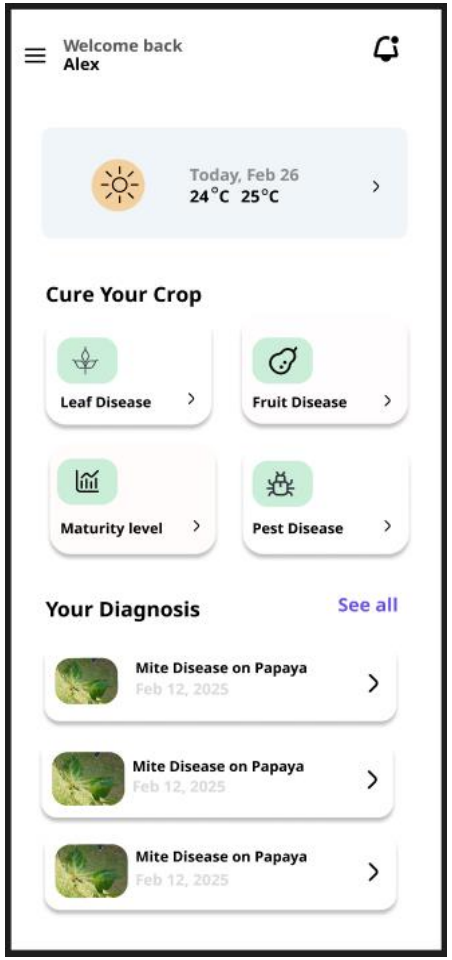
```
4/4 [==============================] - 80s 12s/step - loss: 0.0459 - accuracy: 0.9917
Test accuracy: 0.99
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 996ms/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 756ms/step
Confusion Matrix:
[[39  0  0  0]
 [ 0 30  0  0]
 [ 0  0 31  0]
 [ 0  1  0 19]]

Classification Report:
              precision    recall  f1-score   support

Healthy Fruit       1.00      1.00      1.00        39
 Healthy Leaf       0.97      1.00      0.98        30
    Mealy Bug       1.00      1.00      1.00        31
     Mite Bug       1.00      0.95      0.97        20

     accuracy                           0.99       120
    macro avg       0.99      0.99      0.99       120
 weighted avg       0.99      0.99      0.99       120
```

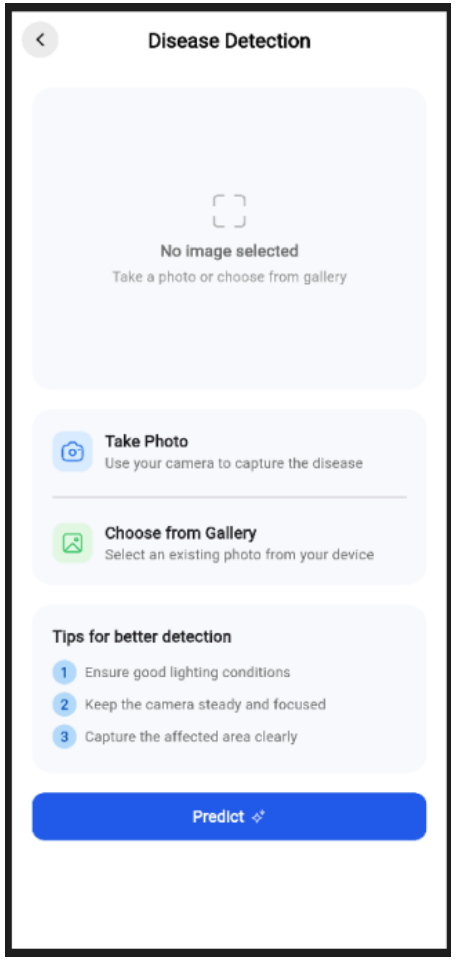
Predicted label across all true labels

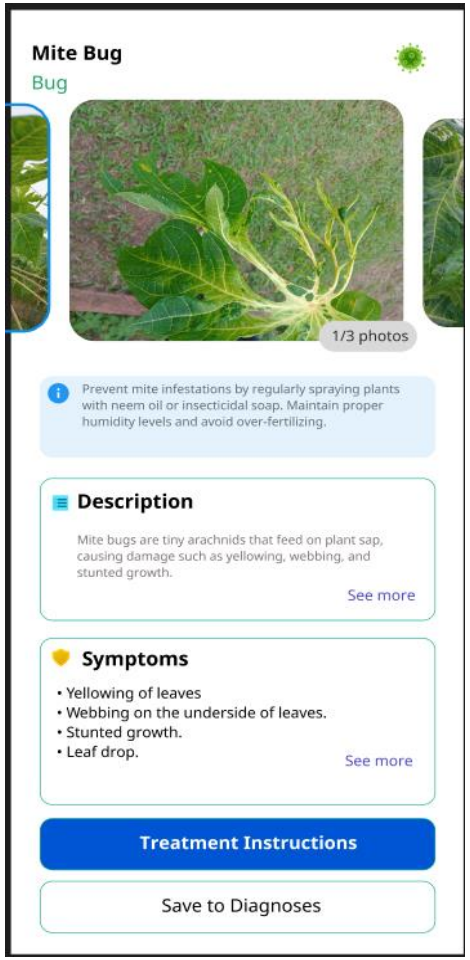


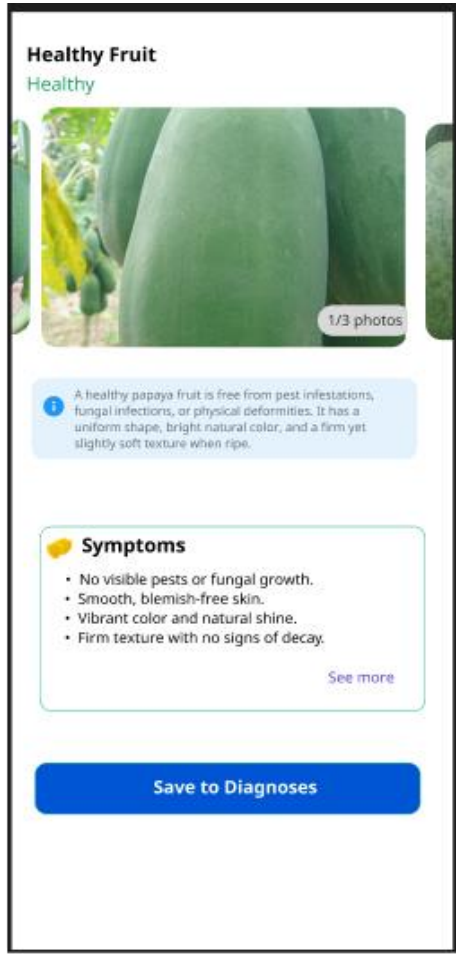
Plotted Predicted label across all true labels

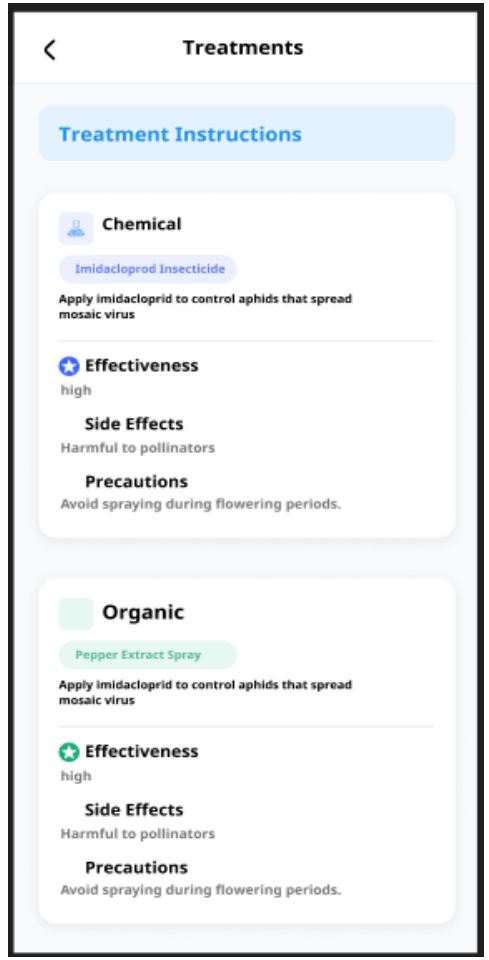
Mobile Application Mockups



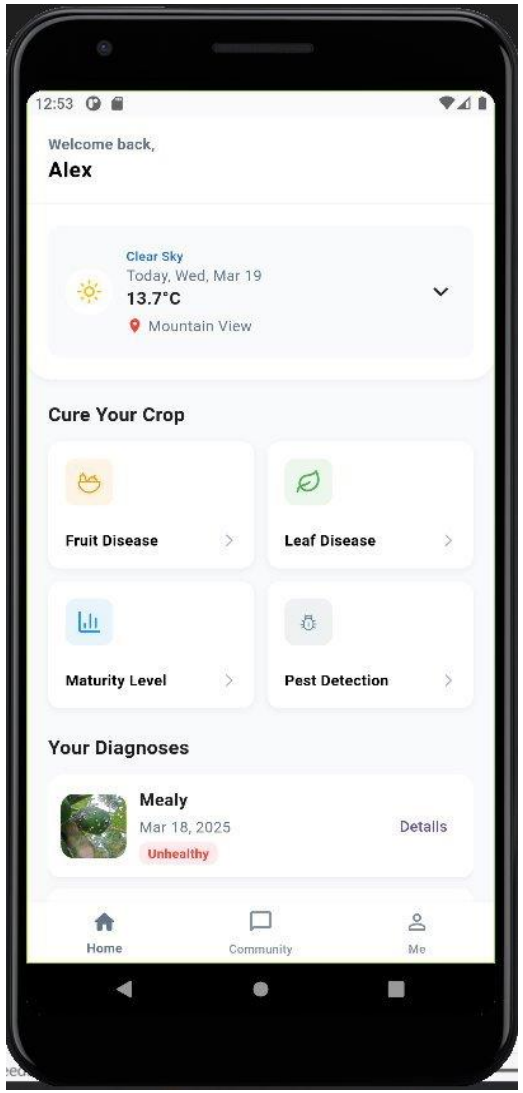


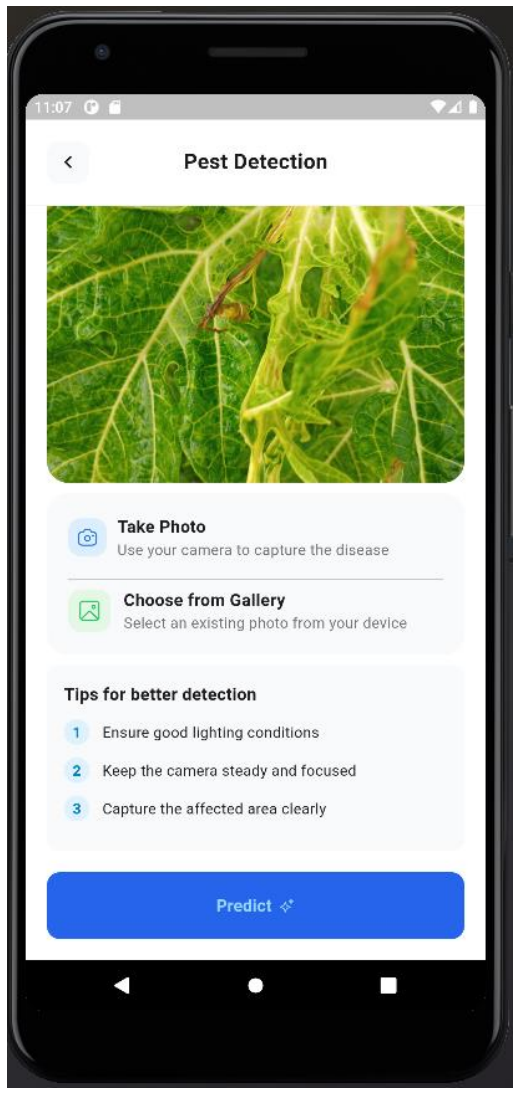


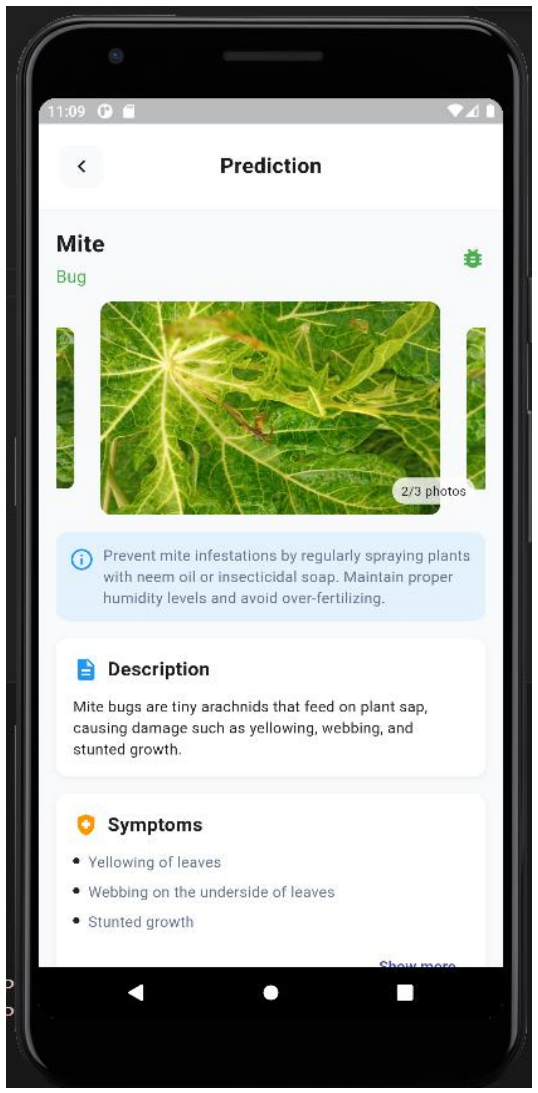


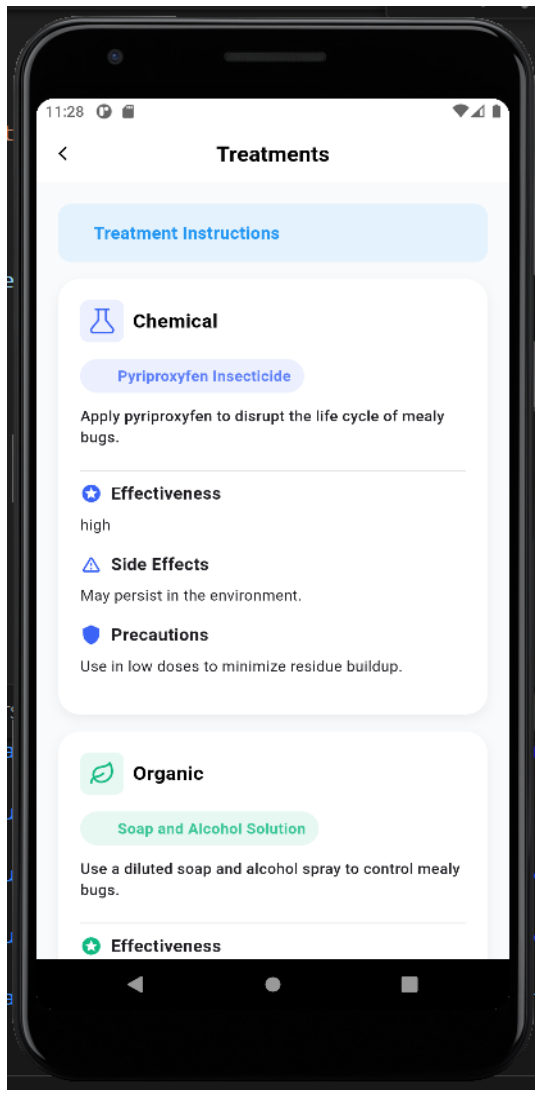


Mobile Application UIs









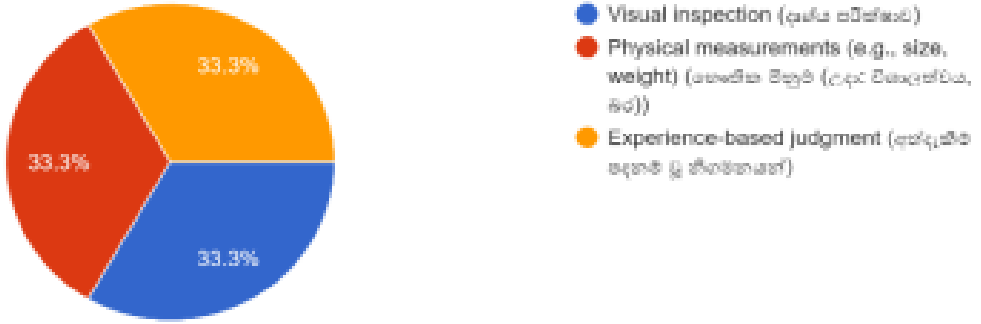
IT21386022 | Senevirathne U.W.H. N

Software Engineering

Research Problem

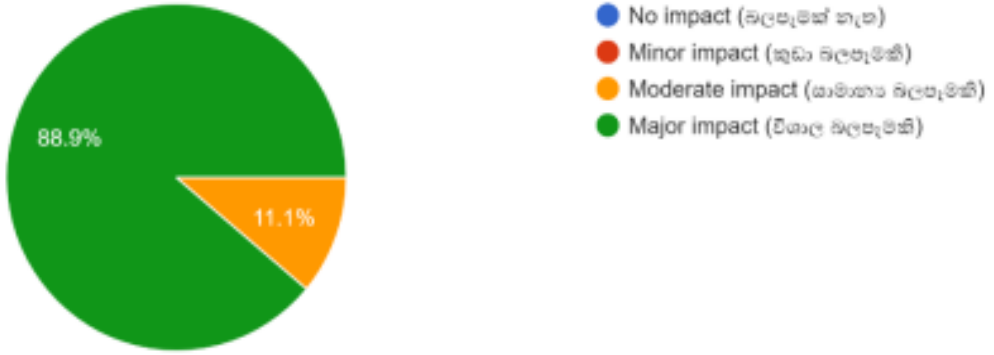
23.How do you currently determine the optimal time for harvesting papaya fruit?(.ඔබ දැනට පැපොල් ගෙඩි අස්වැන්න සඳහා සුදුසුම කාලය නිර්ණය කරන්නේ කෙසේද?)

9 responses



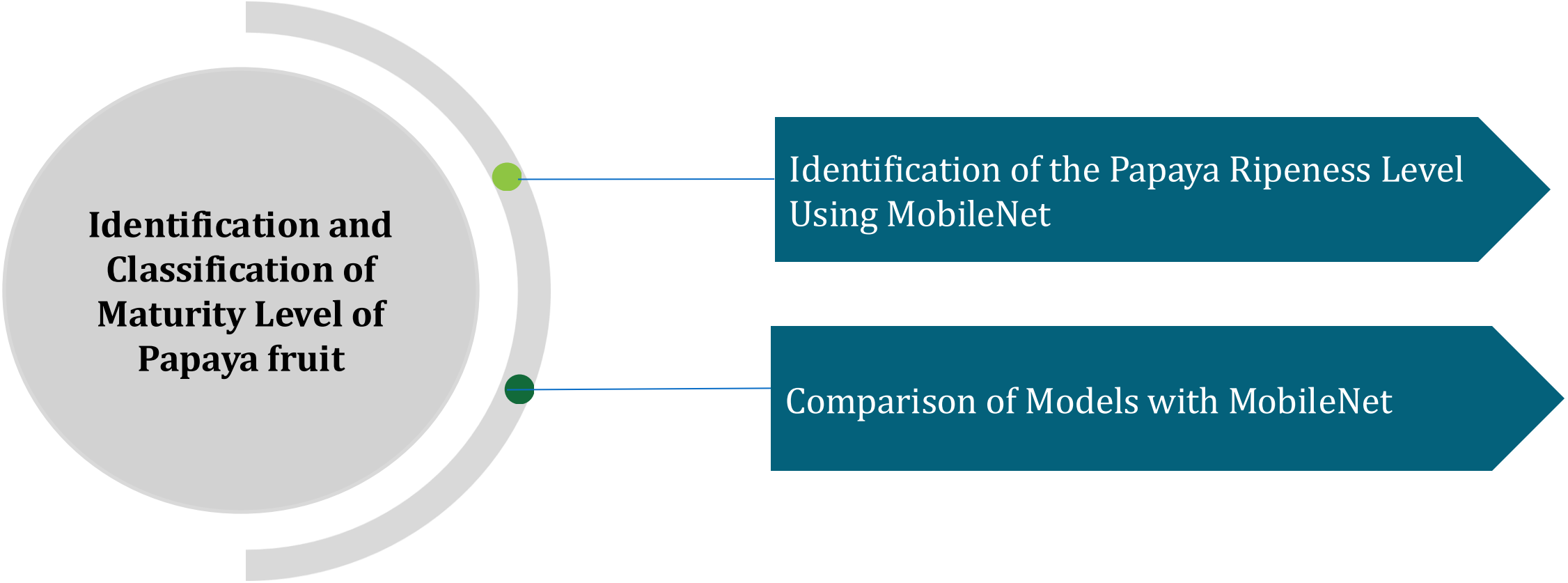
How do you determine the optimal time for harvesting papaya fruit?

25.How significant is the impact of harvesting papayas at the wrong maturity level on your overall yield? (වැරදි පරිණත මට්ටමකින් ගස්ලබු අස්වැන්න ඔබේ සමස්ත අස්වැන්නට කෙතරම් බලපානවද?)



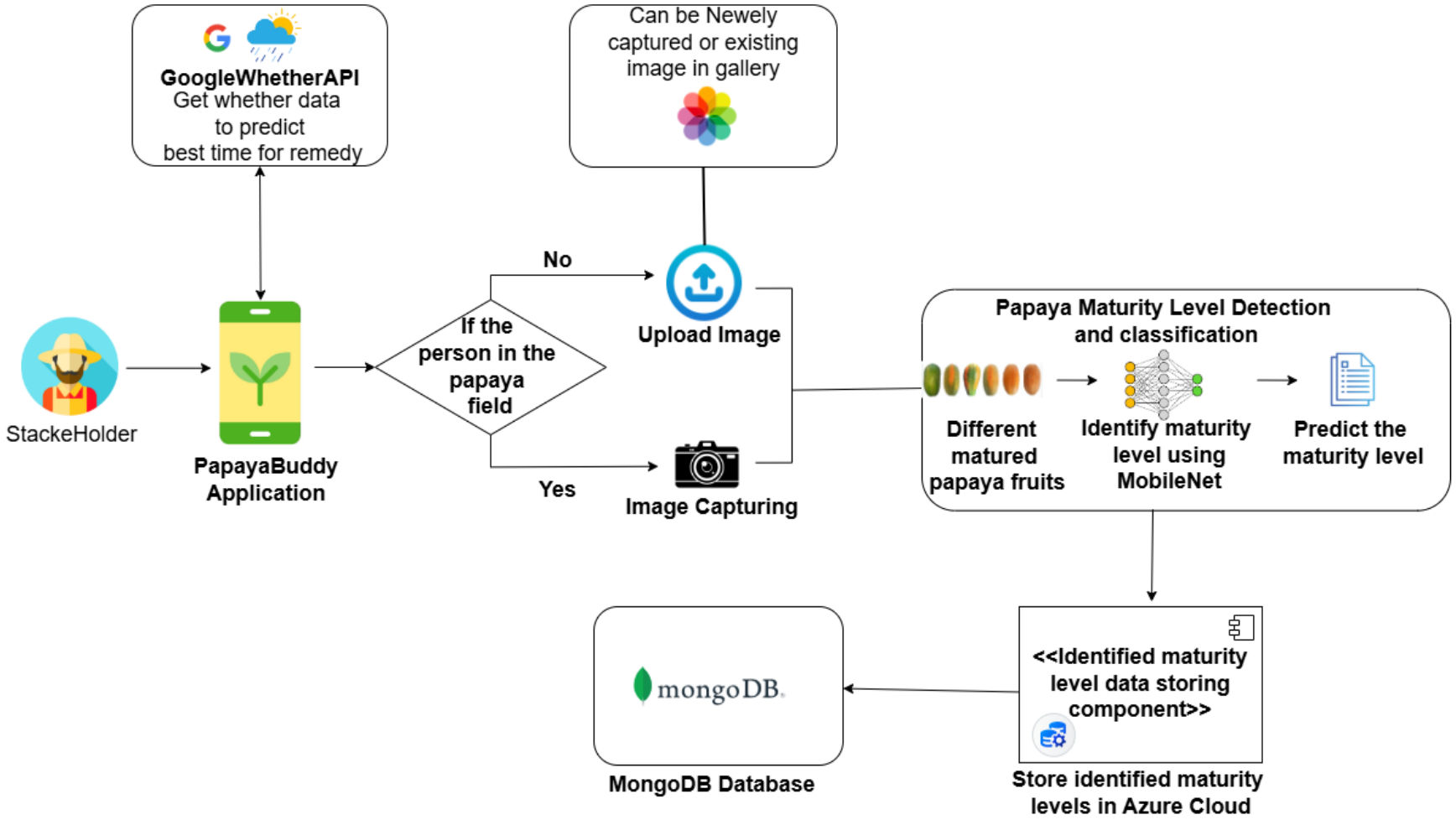
How significant is the impact of harvesting papayas at the wrong maturity level on your overall yield?

Specific and Sub Objectives



System Diagram

Papaya Maturity Level Detection



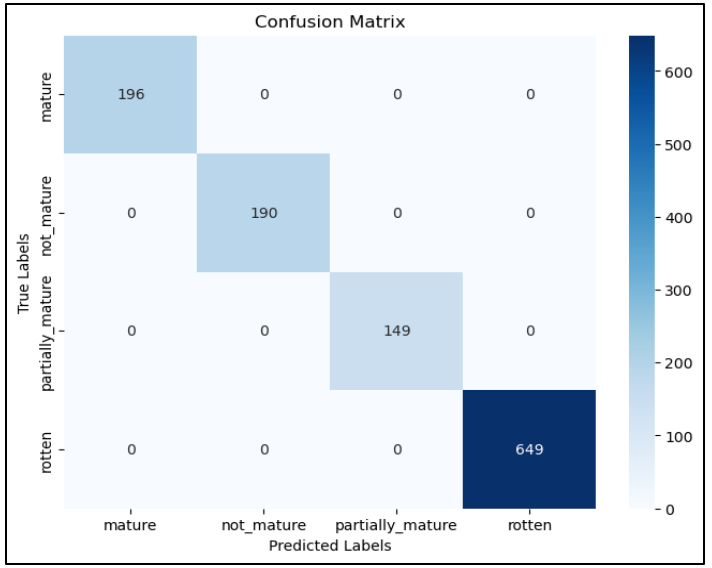
Evidence of completion

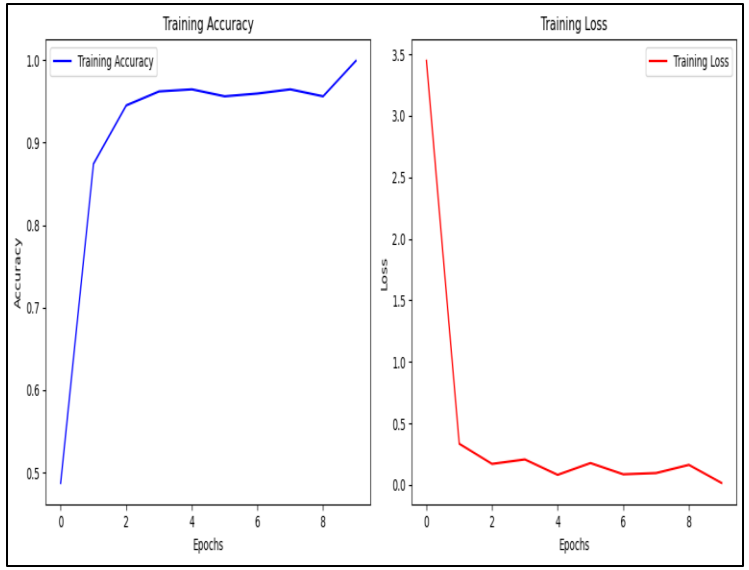
- Maturity Piker UI
- Maturity Stage UI
- Maturity Prediction Node Controllers
- Implement MobileNetV2 Model
- Host the finalized model in the flask server
- Display the results in mobile application

MobileNetV2 Model Evaluation

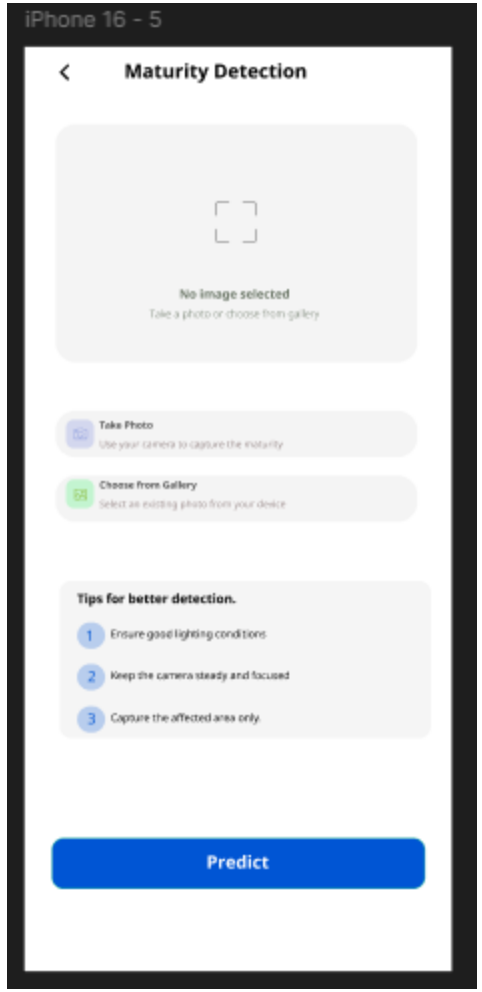
Evidence of Completion

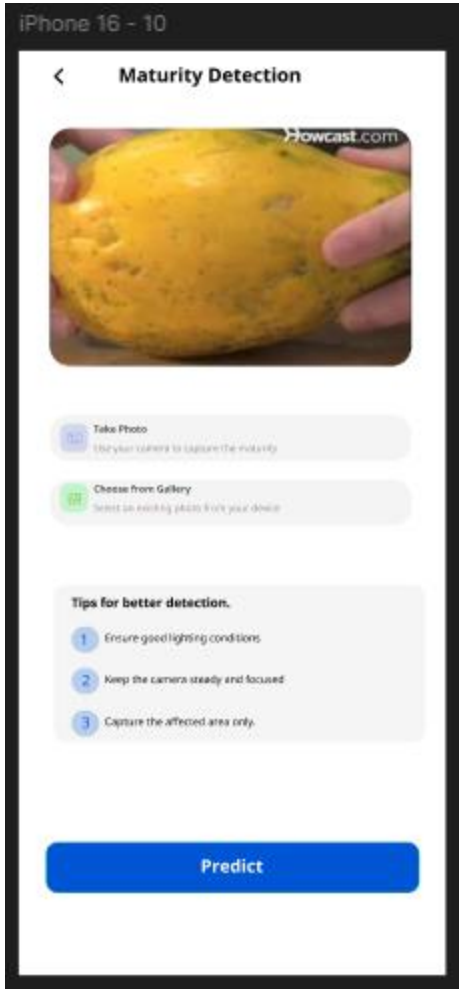
Variables	MobileNetV2
Modal Version	MobileNet2
No of Classes	4
Dataset Size	2000 images
No of Epochs	10
Batch Size	32
Input Size	(224 x 224)
Used Environment	Jupyter notebook

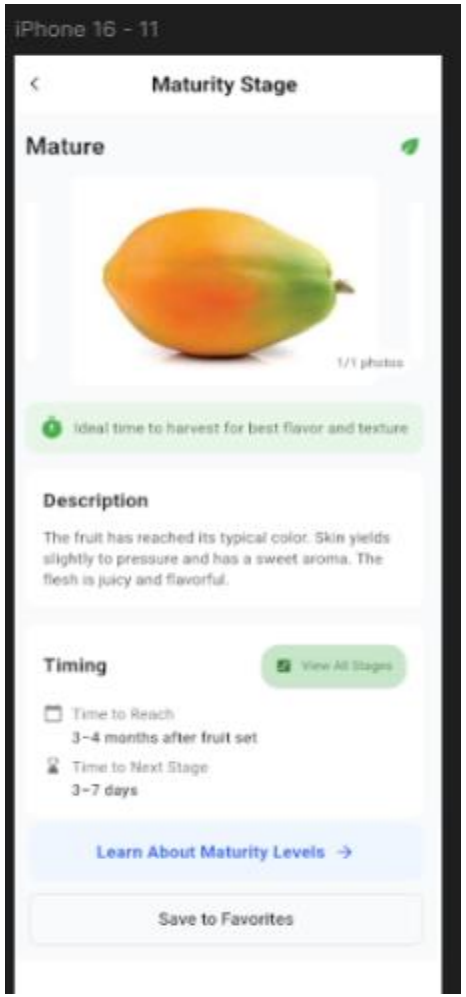


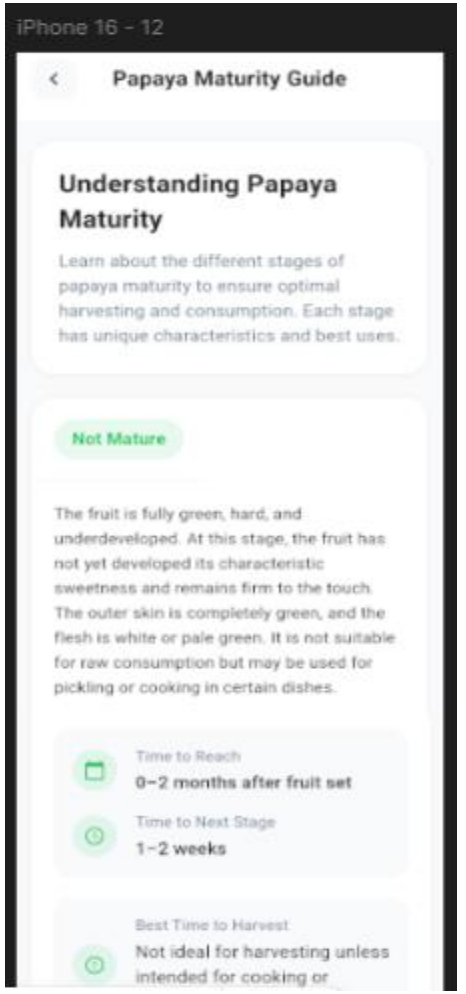


Mobile Application Mockups

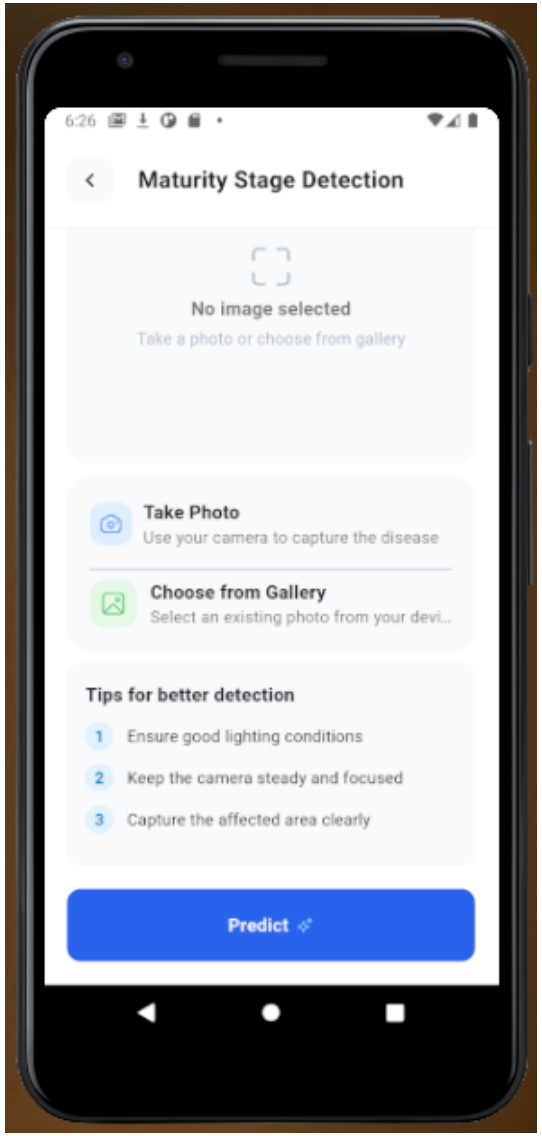


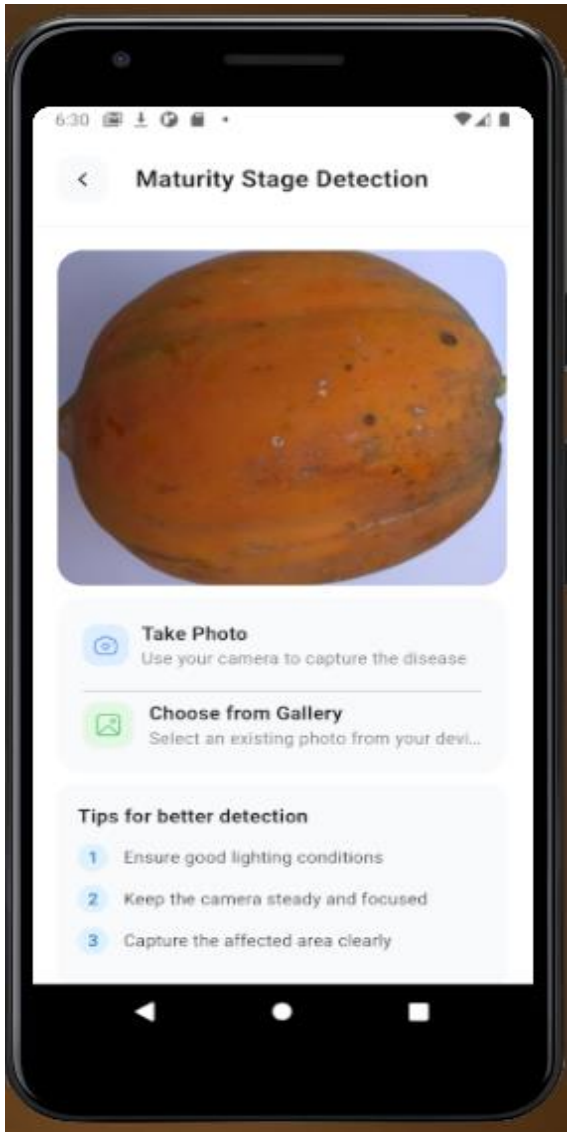


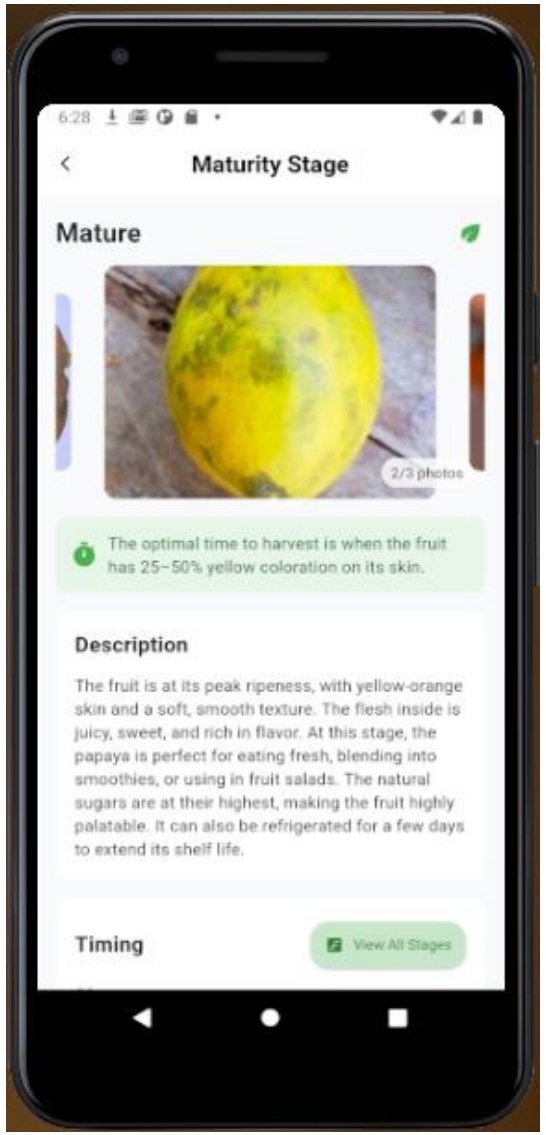


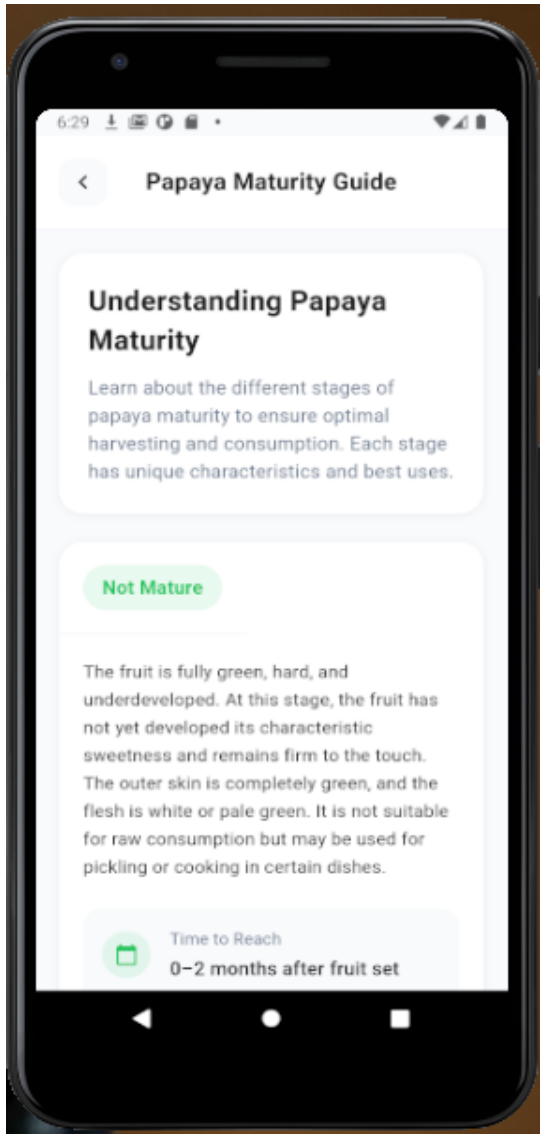


Mobile Application UIs

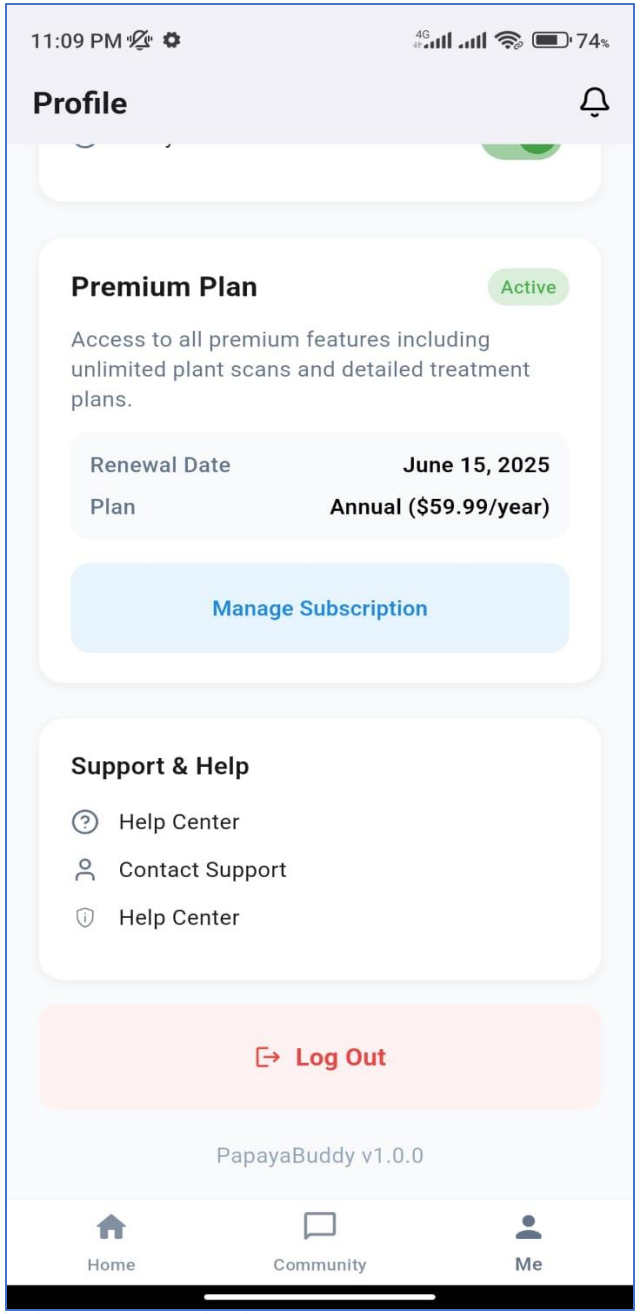








Commercialization



Commodity version

Identification of,

- Cercospora
- Mosaic Virus
- Powdery Mildew
- Ring Spot
- Identification of Bugs
- Fruit Maturity
- Remedy Suggestion

Up to 15 images/ free week

Premium version

- **Commodity version +**
- **With the lowest price**

Up to 100 Images

- Monthly - Price: Rs. 2500.00

Unlimited Images

- Yearly - Price: Rs. 10000.00

Target Audience

- Farmers
- Stakeholders
- Researchers

Market Place

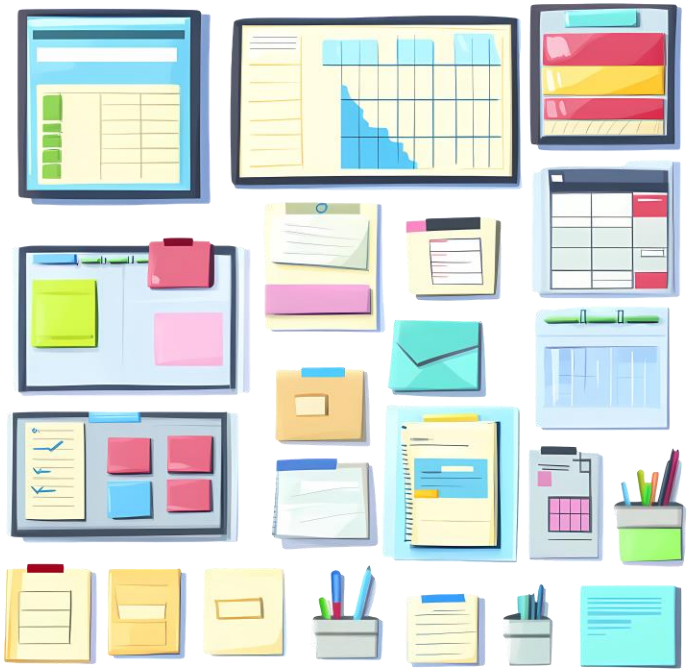
- Excellent user experience through simpler User Interface
- No need of advanced technical knowledge

Completion and Future work

Completion of the Components

90%

- Fruit/ Leaf/ Pest disease detection
- Maturity level detection
- User Management
- Remedy Suggestion
- Community Component View
- Integrate Whether API
- Fast API/Flask API integration with Models



Future implementation

10%

- Develop Community Functionality
- Deploy PapayaBuddy app in Play-Store

Key Challenges & Solutions

- Containerizing services with Docker in the Microservices architecture, including inter-service communication, performance optimization, and network configurations.
- Encountered dependency version conflicts with libraries like TensorFlow, NumPy and etc. To mitigate this, we implemented a Microservices architecture with separate models, ensuring isolation and compatibility.
- Monolithic architecture required excessive computing resources, leading to limitations. To overcome this, we adopted a Microservices architecture for better resource efficiency and scalability.
- Data collection required long-distance travel with high costs, and we faced challenges such as bad weather conditions and poor lighting, affecting data quality.
- Poor camera setup impacted image quality, and the costs for Colab and Google Drive storage added to the challenges.
- Implemented a Microservices architecture to resolve dependency conflicts, with Node.js for data handling, FastAPI for model hosting, and Kong as an API gateway for authentication, authorization, load balancing, rate limiting, and secure service routing.

Papaya Buddy

Thank You !