UNIVERSITY OF THE WESTERN CAPE

MODULE: IFS 325

GROUP ASSIGNMENT

PROJECT DESCRIPTION DOCUMENTATION

GROUP 3 –

CROP DEVELOPMENT VISION SYSTEM

SUBMITTED BY **GROUP 5**:

RAEESAH DARBAR: 4356374

MUBASHIER OMAR: 4348127

MUAATH SALIE: 4369122

DYLAN-THOMAS PUGH: 43553847

NTOKOZO THOKOZANI MHLAMBI: 4337274

MAAJIDA JAKOET: 4227672

SUBMITTED TO: RUCHEN WYNGAARD

DATE OF SUBMISSION: 05 NOVEMBER 2025

Contents

1. Project Description

This project focuses on the design and development of FarmEye, an AI-powered Crop Development Vision System: a low-cost, low-energy Internet of Things (IoT) solution for the Agricultural Research Council (ARC).

FarmEye leverages computer vision, machine learning, and edge computing to monitor crop health, detect pests and diseases, and analyze growth stages in greenhouse environments. It integrates Raspberry Pi edge processing, Firebase cloud storage, and Oracle Apex database management with a Flutter-based frontend for real-time, accessible insights.

Development follows Agile methodology, emphasizing modularity, scalability, and data-driven decision support for smart agriculture.

1.1 The Problem

ARC currently relies on manual monitoring for assessing crop health, pest infestations, and growth progress within its greenhouse facilities.
This process is labour-intensive, time-consuming, and subject to human error, leading to:

- Delayed detection of diseases and pests results in potential crop loss. Inefficient use of water, pesticides, and fertilizers.
- Difficulty maintaining consistency and scalability in monitoring across large operations.
- Furthermore, commercial smart farming solutions are often expensive, limiting their adoption by research and agricultural organizations.

1.2 The Solution

FarmEye provides a fully automated, real-time crop monitoring pipeline that integrates AI vision models, IoT devices, and cloud technologies.

Key features include:

- IoT-based image capture using a Raspberry Pi with an attached camera module.
- Machine learning models are trained to identify crop growth stages, diseases, and pests.
- Edge inference for on-device processing and real-time decision-making.
- Cloud synchronization via Firebase Storage for image archiving and Oracle Apex for structured data storage.
- MQTT-based communication for transmitting predictions to the Data Management Team's central system.

- A Flutter mobile/web frontend for farm managers to review live and historical crop analytics, receive AI-driven recommendations, and manage interventions remotely.

This system reduces the reliance on manual inspections, enhances responsiveness, and facilitates data-driven precision agriculture.

1.3 Project Objectives

1. Design and implement a low-cost IoT vision device capable of capturing high-quality crop images in greenhouse conditions.
2. Train and integrate machine learning models to accurately classify growth stages and detect diseases or pest infestations.
3. Deploy an MQTT communication protocol for real-time message transmission between the IoT device and the data platform.
4. Develop a user-friendly Flutter frontend application for farm managers to monitor, analyse, and respond to crop conditions.
5. Integrate Firebase for secure image storage and Oracle Apex for centralized data management.
6. Follow Agile methodologies for iterative development, testing, and deployment.

1.4 Scope of Work

The scope covers the end-to-end system lifecycle, from hardware design to deployment and testing:

System Lifecycle	
Hardware Setup:	Configure a Raspberry Pi 5 with a camera module for automated image capture and live video streaming.
Software Development:	Build a Python-based local inference application that runs trained CNN models for pest, disease, and growth stage detection.
AI/ML Model Training:	Train deep learning models using TensorFlow/Keras on custom datasets for apples, grapes, tomatoes, lettuce, and basil.
Cloud and Communication Integration:	Configure Firebase Storage for image uploads and MQTT for real-time message exchange with the Data Management platform.

Frontend Development:	Create a Flutter-based dashboard that retrieves data from Firebase and Oracle Apex, displaying historical results and live detections.
Testing and Integration:	Perform local and remote testing to validate accuracy, connectivity, and responsiveness across the full system.
Documentation:	Develop comprehensive deployment, system architecture, and user documentation.

1.5 System Architecture

1.5.1 Overview

The Crop Development Vision System follows a multi-layered IoT architecture, integrating hardware, software, cloud services, and user interface components to deliver real-time crop monitoring and analysis.
It consists of three primary layers:

1. Edge Layer (IoT Device) – Responsible for image capture and on-device AI inference.
2. Cloud Layer (Data Management and Storage) – Handles image uploads, data synchronization, and model result storage.
3. Application Layer (Frontend Dashboard) – Provides users with visual access to analytics, alerts, and historical data.

This layered design ensures scalability, modularity, and resilience, allowing each component to operate independently while maintaining seamless data flow across the system.

1.5.2 Architecture Diagram

The architecture diagram outlines how system components interact, from image capture on the Raspberry Pi to cloud storage in Firebase and Oracle Apex, and data visualization in the Flutter frontend.

It highlights the data flow, AI processing, and communication through the MQTT protocol.

(See Appendix A – System Architecture Diagram.)

1.5.3 Data Flow

Phase	Description
Capture Phase →	The IoT device continuously captures crop images in greenhouse environments.
Inference Phase →	Images are processed locally using trained models, generating predictions.
Transmission Phase →	Results and image data are transmitted through MQTT to cloud storage and the data management system.
Storage Phase →	Firebase archives image files, while Oracle Apex records structured metadata and prediction results.
Visualization Phase →	The Flutter frontend retrieves and displays this data for farm managers and ARC scientists.
Feedback Phase →	Users can view historical data, compare growth trends, and receive AI-driven recommendations, closing the decision-making loop.

1.6 Project Deliverables

Deliverable Type	Item	Target Audience
Functional System	Fully deployed, working IoT Crop Development Vision System	ARC Operations
Hardware/Firmware	Configured Raspberry Pi with AI models and camera integration	Technical Team, ARC Engineers
Software	Trained CNN models for disease, pest, and growth stage classification	Research & Development Team
Software	Flutter-based mobile/web dashboard connected to Firebase and Oracle Apex	Farm Managers, End-Users
Code Base	Complete, version-controlled source code hosted on GitHub and iKamva	Technical Team, Lecturer
Documentation	System Architecture, Deployment & User Manuals	Farm Managers, End-Users
Project Management	Final project presentation and demonstration	Lecturer, Stakeholders

1.7 Stakeholders

Stakeholder	Role / Contribution
Agricultural Research Council (ARC)	Project client and primary end user
ARC Farm Managers & Greenhouse Operators	Daily users of the system for monitoring and management
ARC Crop Scientists & Researchers	Consumers of crop diagnostics and image datasets
Data Management Team	Responsible for cloud data handling and MQTT integration
Project Supervisor (Ruchen Wyngaard)	Oversight, evaluation, and project guidance
Group 5 Development Team	Responsible for full system design, development, and testing

1.8 Business Value

The Crop Development Vision System delivers measurable business and operational benefits to ARC:

Value Area	Benefit
Increased Crop Yield:	Early and precise detection of diseases and pests minimizes damage and enhances productivity.
Resource Optimization:	Data-driven irrigation, fertilizer, and pesticide use improve cost-efficiency and sustainability.
Operational Efficiency:	Automated crop monitoring reduces manual labour and improves the consistency of inspection.
Affordable Scalability:	Offers a low-cost alternative to commercial systems, making smart agriculture accessible.
Research Advancement:	Contributes labelled datasets and AI insights to ARC's research for future innovation in crop science.

2. Project Management

2.1 Methodology

The project follows an Agile methodology, focusing on iterative development, testing, and feedback. Each sprint covers AI model improvements, hardware integration, and system validation, ensuring flexibility and progressive refinement throughout the project lifecycle.

2.2 Work Breakdown Structure (WBS)

The system development process is divided into distinct modules and sub-tasks, representing all deliverables and dependencies.

(See Appendix B – Work Breakdown Structure.)

2.3 Roles & Responsibilities

The detailed roles and responsibilities for each team member are outlined below, ensuring clear accountability and equitable contribution throughout the project.

Name	Role	Responsibilities
Muaath Salie 439122	AI Model Specialist & Data Engineer	• Develop, train, and optimize growth stage and disease detection models. • Manage dataset preprocessing, validation, and model improvement. • Co-author Deployment Documentation. • Integrate trained models onto the Raspberry Pi. • Create and maintain the Gantt Chart for project scheduling and progress tracking.
Maajida Jakoet 4227672	Data Preparation & Model Training Assistant	• Develop and refine growth stage classification models. • Support dataset structuring, labelling, and quality assurance. • Create and maintain the Work Breakdown Structure (WBS) document.
Dylan-Thomas Pugh 4353847	Frontend Developer & UI Designer	• Design and develop the Flutter frontend (draft and prototype). • Ensure integration between the frontend and Firebase database.

		• Implement a responsive dashboard for monitoring system data. • Collaborate with Ntokozo Mhlambi on frontend functionality. • Contribute to the Project Description Documentation.
Mubashier Omar 4348127	Lead Developer & Systems Integrator	• Configure and install all required software on the Raspberry Pi. • Train and integrate all AI models onto the device. • Develop and test disease detection models. • Establish a connection to the Topic 5 Team via MQTT. • Lead Deployment Documentation. • Manage and update the Trello board for task tracking. • Complete and deploy the Crop Vision System end-to-end.
Ntokozo Thokozani Mhlambi 4337274	Frontend Integration & Requirements Analyst	• Develop the Flutter app prototype and ensure full connectivity to Firebase. • Finalize and test all frontend functionalities. • Define Functional and Non-Functional Requirements. • Support real-time data display and visualization in the Flutter app.
Raeesah Darbar 4356374	Coordinator, AI Developer & Documentation Lead	• Train and refine disease detection and growth stage models. • Contribute to model evaluation and integration with the Raspberry Pi. • Oversee project documentation and progress tracking. • Compile and refine all written reports and deliverables. • Ensure captured images are uploaded and stored correctly on Firebase. • Manage Roles & Responsibilities, Project Description, and User Documentation.

2.4 Project Timeline (Gantt Chart)

A structured Gantt chart outlines the phases for research, model training, development, testing, and deployment, ensuring alignment with the final submission deadline of **05 November 2025**.

(See Appendix C – Gantt Chart.)

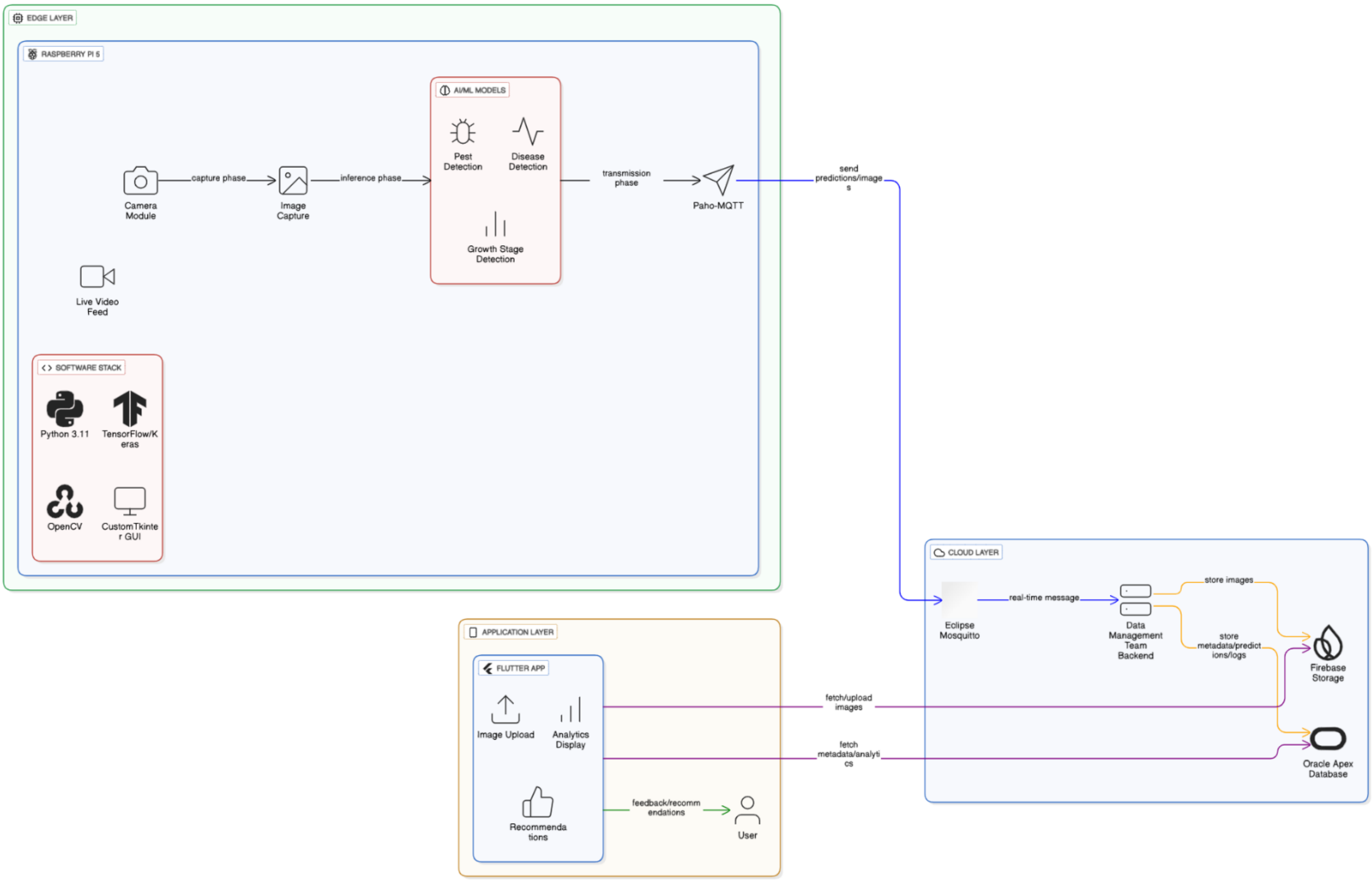
2.5 Success Criteria

Project success will be evaluated based on the following measurable outcomes:

- Functional IoT device performing reliable image capture and inference.
- AI models are achieving at least 85% accuracy across detection categories.
- Successful data transmission to Firebase and Oracle Apex.
- Full integration with the Data Management Team's platform.
- Flutter frontend successfully displaying live and historical data.
- Comprehensive documentation and adherence to deadlines.

Appendix

Appendix A: System Architecture Diagram



Appendix B: Work Breakdown Structure

For better View – Link for our WBS Google Sheet :

https://docs.google.com/spreadsheets/d/1gfpphJEXfGhi9TBPXssJwIzNd4qn94qw/edit?usp=sharing&ouid=105601601280337457575&rtpof=true&sd=true

WORK BREAKDOWN STRUCTURE

PROJECT TITLE	FARM EYE CROP VISION SYSTEM						COMPANY NAME		
PROJECT MANAGER							DATE		
LEVEL 1									
LEVEL 2	Project Planning and Analysis			System Design and Architecture			Model Development and Testing		
LEVEL 3	Task 1.1	Task 1.2	Task 1.3	Task 2.1	Task 2.2	Task 2.3	Task 3.1	Task 3.2	Task 3.3
	Define project scope	Assign Team roles	Identify system requirements	Design System Architecture	Define Hardware Setup	Define Software Architecture	Data Collection	Data Preprocessing	Train AI Model
LEVEL 4	Subtask 1.1.1.	Subtask 1.2.1	Subtask 1.3.1	Subtask 2.1.1.	Subtask 2.2.1	Subtask 2.3.1	Subtask 3.1.1.	Subtask 3.2.1	Subtask 3.3.1
	Identify project goals and deliverables	Identify each member's strengths and expertise	Define functional and non-functional requirements	Develop system block diagream showing IoT data flow.	Select Raspberry Pi componenets and camera module.	Define backend and frontend components.	Gather sample crop images from PlantVillage or similar datasets.	Resize and normalise images for training.	Use MobieNetV2 or similar CNN architecture.
	Outline system objectives and success criteria	Allocate roles	Determine data sources, tools, and hardware.	Identify how Pi, AI model, and dashboard connect.	Determine optimal camera angles and capture intervals.	Outline APIS for data exchange with Group 5.	Organise images into labeled folders (Healthy, Diseased, Growth,Pest)	Split dataset into training, validation, and test sets.	Configure hyperparameters)batch size, learning rate).
	Establish project boundaries and limitations	Communicate responsibilities clearly	Document software dependencies.	Define system inputs, processes, and outputs.	Plan data storage and power management setup.	Specify data formats (JSON, image storage path).	Augment dataset (flipping, rotation, zooming).	Verify data balance across categories.	Monitor accuracy and loss metrics during training.

PROJECT TITLE			ARC - Agricultural Research Council									
PROJECT MANAGER			Oct-25									
LEVEL 1	Implementing Crop Development Vision System											
LEVEL 2	System Development (Frontend, Backend, Integration)				Hardware setup and Integration			System testing and Validation			Documentation and Presentation		
LEVEL 3	Task 4.1	Task 4.2	Task 4.3	Task 4.4	Task 5.1	Task 5.2	Task 5.3	Task 6.1	Task 6.2	Task 6.3	Task 7.1	Task 7.2	Task 7.3
	Backend API Development	Frontend Dashboard	Data Integration	External System Integration	Raspberry Pi Setup	Connect AI Model	Implement MQTT Communication	Functional Testing	Performance Testing	Usability Testing.	Technical Documentation.	User manual	Progress Report
LEVEL 4	Subtask 4.1.1.	Subtask 4.2.1	Subtask 4.3.1	Subtask 4.4.1	Subtask 5.1.1.	Subtask 5.2.1	Subtask 5.3.1	Subtask 6.1.1.	Subtask 6.2.1	Subtask 6.3.1	Subtask 7.1.1.	Subtask 7.2.1	Subtask 7.3.1
	Develop FastAPI endpoints for image upload.	Design user interface for image upload and results view.	Create schema for storing analyss results.	Establish MQTT topic structure and naming conventions with Group 5	Install Raspberry Pi OS and required libraries.	Upload trained .tflite model to the Pi.	Install and confgure MQTT broker.	Test each feature.	Measure average response time per image.	Collect feedback from potential users.	Document hardware setup and software configurations.	Write simple step-by-step guide for system use.	Summarise completed tasks and milestones.
	Integrate model inference and return prediction results.	Add visualisation features for health trends and alerts.	Implement query functions for viewing past records.	Integrate soil moisture and humidity data endpoints from Group 5.	Configure camera module and verify image capture.	Integrate model inference code to process captured images in real time.	Develop MQTT publisher on Raspberry Pi to send results.	Confirm all buttons and links work.	Evaluate model latency and processing load.	Assess ease of navigation and layout clarity.	Create architecture and date flow diagrams.	Include screenshots of interface and Pi setup.	Include testing results and lessons learned.
	Implement data storage for results (SQLite or JSON).	Ensure responsive design for mobile and desktop.	Test database connections with backend endpoints.	Validate data sychronisation between systems and frontend display.	Test local model inference with sample images before integration.	Validate output predictions locally and optimise inference speed.	Test data transmission with Flutter app subscriber and verify alert updates.	Validate end-to-end workflow.	Optimise code for faster inference.	Make improvements based on input.	Include details on AI model training process.	Highlight troubleshooting tips.	Document improvements from feedback.

Appendix C: *Gantt Chart*