DEVELOPMENT OF AN IOT-INTEGRATED APP FOR MONITORING HYDROPONIC FARMING SYSTEMS

CHAN JIA JUN

UNIVERSITI TUNKU ABDUL RAHMAN

DEVELOPMENT OF AN IOT-INTEGRATED APP FOR MONITORING HYDROPONIC FARMING SYSTEMS

CHAN JIA JUN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :

Name : Chan Jia Jun

ID No. : 2002845

Date : 2/10/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DEVELOPMENT OF AN IOT-INTEGRATED APP FOR MONITORING HYDROPONIC FARMING SYSTEMS"** was prepared by **CHAN JIA JUN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature :

Supervisor : See Yuen Chark

Date : 2/10/2024

Signature :

Co-Supervisor : Ts.Dr.Sugumaran Nallusamy

Date : 2/10/2024

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone for providing me with the resources and support necessary to complete this project. I would like to thank my supervisor, Dr. See Yuen Chark, for his guidance and support throughout the project process. His expertise and feedback have been invaluable in completing this project.

In addition, I would like to thank my friend, Chua Shi Jian, for his support on data collection and integration of hydroponic farm system. His contributions have been essential to the success of this project. Moreover, I would also like to acknowledge my other friends that support on this project, providing valuable feedback and insights that helped improve the quality of this project.

Lastly, I am deeply grateful to my family and friends for their unwavering support and encouragement throughout this final year project journey. Their love and support have been a constant source of motivation.

ABSTRACT

As urbanization accelerates in Malaysia, the demand for fresh produce continues to rise, yet the availability of agricultural land is diminishing. Hydroponics, a soil-less cultivation method, presents a viable solution to this challenge. However, managing hydroponic systems can be time-intensive, particularly for urban dwellers with busy lifestyles. This project proposes an IoT-integrated hydroponic farm monitoring mobile application designed to address these challenges. The development of the application focused on a feature-driven approach using React Native, NodeJS, and Flask frameworks. The application enables users to remotely monitor and control environmental parameters within hydroponic farms while receiving real-time notifications about farm status, thereby enhancing overall farm management efficiency. A key aspect of the project was ensuring the application's usability, with features such as environmental parameter trend predictions to facilitate automated hydroponic farm management. This innovation has significant implications for farm management and plant growth, allowing users to manage their time more effectively and increase productivity. Moreover, the application supports plant management by allowing users to record observations and plan tasks with reminder notifications, streamlining the process of tracking plant growth without relying on physical records. Overall, this project contributes to advancing urban agriculture in Malaysia by providing a convenient and efficient mobile solution for managing hydroponic farms, thereby promoting sustainable and efficient urban farming practices.

TABLE OF CONTENTS

DECLARATION			**i**
APPROVAL FOR SUBMISSION			**ii**
ACKNOWLEDGEMENTS			**iv**
ABSTRACT			**v**
TABLE OF CONTENTS			**vi**
LIST OF TABLES			**xi**
LIST OF FIGURES			**xiv**
LIST OF SYMBOLS / ABBREVIATIONS			**xvii**
CHAPTER			
1	**INTRODUCTION**		**1**
	1.1	General Introduction	1
	1.2	Project Background	2
	1.3	Problem Statement	3
		1.3.1 Time Constraints and Monitoring Challenges in Urban Hydroponic Farming	3
		1.3.2 Skills Gap and Technological Adoption in Malaysian Hydroponic Agriculture	4
	1.4	Aim and Objectives	4
	1.5	Scope and Limitations of Study	5
	1.6	Proposed Solution	5
	1.7	Proposed Methodology	6
2	**LITERATURE REVIEW**		**7**
	2.1	Introduction	7
	2.2	Features and Capabilities of Hydroponic Farming System Mobile Applications	7
		2.2.1 User Interface Design	10
	2.3	Machine Learning Model	12
		2.3.1 Long-Short-Term-Memory (LSTM)	12
		2.3.2 Isolation Forest	13
		2.3.3 One-Class Support Vector Machine (One-Class SVM)	14

		2.3.4 Anomaly Detection Model in Agricultural Industry	15
		2.3.5 Justification on Machine Learning Model Selection	16
	2.4	Summary	17
3	**METHODOLOGY AND WORK PLAN**		**19**
	3.1	Introduction	19
	3.2	System Development Methodology	19
		3.2.1 Waterfall Methodology	19
		3.2.2 Unified Process (UP)	20
		3.2.3 Lean Development	21
		3.2.4 Feature Driven Development	23
		3.2.5 Comparison among SDLC methodologies	24
		3.2.6 Activities in Each Phase	25
	3.3	Machine Learning	27
		3.3.1 Datasets Selection and Preparation	27
		3.3.2 Model Training	29
	3.4	Work Plan	30
		3.4.1 Work Breakdown Structure	31
		3.4.2 Gantt Chart	35
	3.5	Development Tools	37
		3.5.1 JavaScript	37
		3.5.2 React Native	37
		3.5.3 Visual Studio Code	37
		3.5.4 Android Studio	37
		3.5.5 Google Collab	38
		3.5.6 Git 38	
		3.5.7 Firebase	38
		3.5.8 Node,js	38
		3.5.9 Flask	38
		3.5.10 TensorFlow	39
		3.5.11 NumPy	39
		3.5.12 Scikit-learn	39
	3.6	Conclusion	39

4 PROJECT SPECIFICATION 40

4.1 Introduction 40

4.2 Requirement Specification 40

4.2.1 Functional Requirements 40

4.2.2 Non-Functional Requirements 42

4.3 Use Case Diagram 42

4.4 Use Case Description 43

4.4.1 Login 43

4.4.2 Register Account 44

4.4.3 Edit Account Credentials 45

4.4.4 Add Plant 46

4.4.5 Edit Plant Details 47

4.4.6 View Plant Details 48

4.4.7 Monitor Real-Time Environmental Parameters 49

4.4.8 Edit Parameters 50

4.4.9 View Notification 51

4.4.10 Configure Notifications Settings 52

4.4.11 Review Data and Insight 53

4.4.12 Detect Anomalies 54

5 SYSTEM DESIGN 55

5.1 Introduction 55

5.2 System Architecture Design 55

5.3 Database Design 58

5.3.1 Entity Relationship Diagram 58

5.3.2 Data Dictionary 60

5.4 User Interface Design 66

5.4.1 User Authentication Pages 66

5.4.2 Monitor Panel Pages 67

5.4.3 Control Panel Screen 69

5.4.4 Data and Insight Screen 69

5.4.5 Plant Management Pages 70

5.4.6 Notification Screen 72

5.4.7 User Profile Pages 72

6 IMPLEMENTATION 75

6.1 Introduction 75

6.2 Frontend Implementation 75

6.2.1 Firebase Authentication 75

6.2.2 Monitor Panel 76

6.2.3 User Profile 77

6.2.4 Control Panel 77

6.2.5 Data and Insight 78

6.2.6 Plant Management 78

6.2.7 Notification 78

6.3 Backend Implementation 79

6.3.1 Application Server 79

6.3.2 Model Server 79

6.3.3 API Functions 80

6.4 Model Training 85

6.4.1 Data Preparation 85

6.4.2 Model Definition 86

6.4.3 Training Loop 87

6.4.4 Model Evaluation 88

6.5 Summary 88

7 SYSTEM TESTING 90

7.1 Introduction 90

7.2 Test Plan 90

7.2.1 Objectives 90

7.2.2 Test Scope 90

7.2.3 Test Basis 91

7.2.4 Test Items 91

7.2.5 Test Strategy 92

7.2.6 Test Criteria 93

7.3 Functionality Test 93

7.3.1 Mobile Application Functionality Test 94

7.3.2 App Server Functionality Test 107

7.3.3 Model Server Functionality Test 119

7.4 Performance Test 121

		7.4.1 Response Time	121
		7.4.2 App Start Time	122
		7.4.3 Response Success Rate	123
		7.4.4 Frozen Frames Percentage	124
	7.5	Anomaly Detection Evaluation	126
	7.6	Summary	132
8	**CONCLUSION AND FUTURE IMPROVEMENT**		**133**
	8.1	Conclusion	133
	8.2	Limitations	134
	8.3	Recommendation for Future Improvements	135
REFERENCES			**136**

LIST OF TABLES

Table 2.1 Comparison of Feature of Hydroponic Farming System Mobile Applications 9

Table 2.2 Advantages and Disadvantages of LSTM 13

Table 2.3 Advantages and Disadvantages of Isolation Forest 14

Table 2.4 Differences between support vectors machine (SVM) and One-Class SVM 15

Table 2.5 Results of the LSTM-CNN model (Alazani et al., 2023) 16

Table 3.1: Advantages and Disadvantages of Waterfall methodology (Gallagher, Dunleavy and Reeves, 2019) 20

Table 3.2 Advantages and Disadvantages of Unified Process (GeeksforGeeks, 2024) 21

Table 3.3 Advantages and Disadvantages of Lean Development 22

Table 3.4 Advantages and Disadvantages of Feature Driven Development (www.productplan.com, n.d.) 23

Table 3.5 Comparison of Software Development Life Cycle methodologies 25

Table 3.6 Environmental Parameters 27

Table 3.7 Datasets Features 28

Table 5.1 Description of Database Tables 60

Table 5.2 Users Entity Data Dictionary 60

Table 5.3 Farms Entity Data Dictionary 62

Table 5.4 Sectors Entity Data Dictionary 62

Table 5.5 Plants Entity Data Dictionary 63

Table 5.6 Devices Entity Data Dictionary 64

Table 5.7 Anomalies Entity Data Dictionary 65

Table 6.1 API List of Multiple Backend Services 80

Table 6.2	LSTM Layer configuration	86
Table 7.1	Functional Services to be tested	91
Table 7.2	Testing Levels, Types and Tools	92
Table 7.3	Test Case MA-1 User registration	94
Table 7.4	Test Case MA-2 User Login	95
Table 7.5	Test Case MA-3 Change Email	96
Table 7.6	Test Case MA-4 Change Password	97
Table 7.7	Test Case MA-5 Retrieve farm and sector data	98
Table 7.8	Test Case MA-6 Update Sector Settings	99
Table 7.9	Test Case MA-7 Add Plant	100
Table 7.10	Test Case MA-8 Add Plant Record	101
Table 7.11	Test Case MA-9 Add Plant Task	102
Table 7.12	Test Case MA-10 Edit Plant Detail	103
Table 7.13	Test Case MA-11 Data Insight Display	104
Table 7.14	Test Case MA-12 Data Export	105
Table 7.15	Test Case MA-13 Notification Retrieval	106
Table 7.16	Test Case MA-14 Notification Delete	106
Table 7.17	Test Case AS-1 Cron Job for Checking and Saving Notifications	107
Table 7.18	Test Case AS-2 Sector Status Update Cron Job	108
Table 7.19	Test Case AS-3 User Registration	109
Table 7.20	Test Case AS-4 Update Email	110
Table 7.21	Test Case AS-5 Check and Update Message Token	111
Table 7.22	Test Case AS-6 Update Notification Settings	112
Table 7.23	Test Case AS-7 Get Sector Latest Data	113
Table 7.24	Test Case AS-8 Update Parameter Settings	113

Table 7.25	Test Case AS-9 Add Sector	114
Table 7.26	Test Case AS-10 Update Parameter Data	115
Table 7.27	Test Case AS-11 Post Trigger Result	116
Table 7.28	Test Case AS-12 Register Device	117
Table 7.29	Test Case AS-13 Add Plant	118
Table 7.30	Test Case MS-1 Receive Data for Anomaly Detection	119
Table 7.31	Test Case MS-2 Predict Trigger Status	120

LIST OF FIGURES

Figure 1.1	Malaysia: Urbanization from 2012 to 2022 by World Bank (World Bank, 2018)	2
Figure 1.2	Survey Result (Cost) (Kyu, 2023)	3
Figure 1.3	Layout of the Proposed System	6
Figure 2.1	User Interface of Smart Suan Pak Nam (Peuchpanngarm et al., 2016)	10
Figure 2.2	User Interface of Smart Suan Pak Nam (Peuchpanngarm et al., 2016)	11
Figure 2.3	User Interface of VertiFarmControl (Kaur et al., 2022)	11
Figure 2.4	User Interface of VertiFarmControl (Kaur et al., 2022)	12
Figure 2.5	LSTM Architecture (GeeksforGeeks, 2019)	13
Figure 2.6	Performance Metrics for Autoencoder Model (Adkisson et al., 2021)	16
Figure 3.1:	Unified Process (GeeksforGeeks, 2024).	21
Figure 3.2	Lean Software Development (trident, 2021)	22
Figure 3.3	Feature Driven Development (www.productplan.com, n.d.)23	
Figure 3.4	Data Training Needs (Baheti, 2021)	29
Figure 3.5	Work Breakdown Structure	33
Figure 3.6	Work Breakdown Structure (continued)	34
Figure 3.7	Gantt Chart	35
Figure 3.8	Gantt Chart (continued)	36
Figure 4.1	Use Case Diagram	42
Figure 5.1	System Architecture Design	56
Figure 5.2	Entity Relationship Diagram	59
Figure 5.3	Login, Register and Forgot Password Screen	66

Figure 5.4	Monitor Panel Screen	67
Figure 5.5	Edit Farm Screen	68
Figure 5.6	Edit Sector Screen	68
Figure 5.7	Control Panel Screen	69
Figure 5.8	Data and Insight Screen	70
Figure 5.9	Plant Screen, Add Plant Screen, and Plant Detail Screen	71
Figure 5.10	Add observations and tasks, and Edit Plant Detail Screen	71
Figure 5.11	Notification Screen	72
Figure 5.12	User Profile Screen	73
Figure 5.13	User Screen with Edit User Credentials Function	74
Figure 6.1	Firebase Authentication Providers	76
Figure 6.2	Model Training Result	87
Figure 6.3	Model Evaluation Result	88
Figure 7.1	19 Hours Response Time Trend from IP 13.229.207.3	122
Figure 7.2	19 Hours App Start Time	123
Figure 7.3	19 Hours Response Success Rate from IP 13.229.207.3	124
Figure 7.4	17 Hours Frozen Frame Percentage for MainActicity instance	125
Figure 7.5	18 Hours Frozen Frame Percentage for dashboardScreen instance	125
Figure 7.6	17 Hours Frozen Frame Percentage for reportScreen instance	126
Figure 7.7	LSTM Prediction Based 2 Weeks Anomaly Detection Result	127
Figure 7.8	LSTM Autoencoder Based 2 Weeks Anomaly Detection Result	127
Figure 7.9	Isolation Forest and One Class Support Vector Machine 2 Weeks Anomaly Detection Result	128

Figure 7.10 LSTM Prediction Based 4 Weeks Anomaly Detection Result 129

Figure 7.11 LSTM Autoencoder Based 4 Weeks Anomaly Detection Result 129

Figure 7.12 Isolation Forest 4 Weeks Anomaly Detection Result 130

Figure 7.13 One Class Support Vector Machine 4 Weeks Anomaly Detection Result 131

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
ARIMA	Autoregressive Integrated Moving Average
CNN	Convolutional Neural Network
CO2	Carbon Dioxide
EC	Electrical Conductivity
FDD	Feature Driven Development
GPU	Graphical Processing Unit
HRDF	Human Resources Development Fund
HTTP	HyperText Transfer Protocol
ICT	Information and Communications Technology
IDE	Integrated Development Environment
iOS	iPhone Operating System
IoT	Internet of Things
LSTM	Long-Short-Term-Memory
MQTT	Message Queuing Telemetry Transport
pH	Potential of Hydrogen
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SVM	Support Vector Machine
TDS	Total Dissolved Solids
UI	User Interface

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Malaysia, a developing nation with a significant agricultural base, is gradually embracing digitization to cope with increasing international competition. Urbanization, a key component of national development, has been steadily rising. As of 2022, Malaysia's urbanization rate reached 78.2%, according to data from the World Bank (2023), as depicted in Figure 1.1. The government's Fourth National Physical Plan aims to further increase the urbanization rate to 85% by 2040 (Department of Statistics Malaysia, 2022). However, urbanization has led to higher population densities in urban areas while diminishing rural populations, which exacerbates the challenge of ensuring an adequate supply of vegetables and fruits in urban markets (Muhammad & Rabu, 2015). To address this issue, urban farming techniques have become increasingly necessary. Among various methods such as vertical farming and aeroponics, hydroponics stands out as the most recognized and practiced in Malaysia (Muhammad & Rabu, 2015).

Hydroponics, a method of growing plants without soil but using nutrient-rich water, offers a viable solution to the challenges posed by limited urban space (Encyclopedia Britannica, n.d.). This technique, known for its efficiency and sustainability, is particularly well-suited to Malaysia's urban communities. It provides an alternative approach to traditional farming, helping to overcome issues related to pest control and enabling indoor cultivation. Nevertheless, the time constraints faced by urban dwellers make it difficult to consistently monitor and manage these systems. Hence, the development of software or mobile applications that allow real-time monitoring of crop status would greatly benefit busy urban farmers.

According to pioneering research by Lakshmanan, Djama, et al. (2020), integrating Internet of Things (IoT) technologies with hydroponic systems heralds a new era of remote farm management. This integration facilitates real-time environmental monitoring via mobile applications, simplifying agricultural oversight and broadening access to farming regardless of an individual's location or expertise (Lakshmanan et al., 2020).

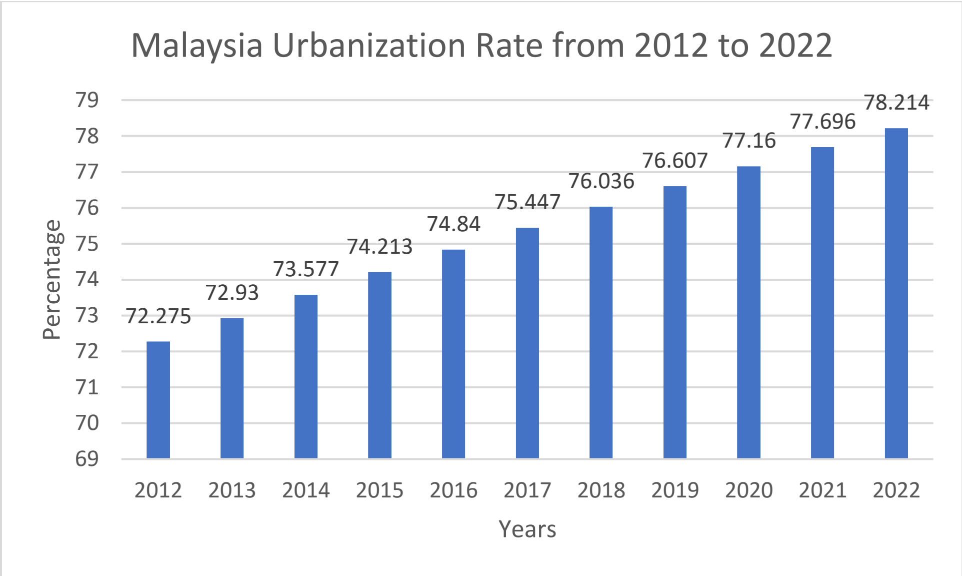

Figure 1.1 Malaysia: Urbanization from 2012 to 2022 by World Bank (World Bank, 2018)

1.2 Project Background

To gauge the need for hydroponic farming system mobile applications in Malaysia, Kyu et al. (2023) conducted a mixed-methods study using online questionnaires and virtual interviews. The study concluded that the majority of respondents believe that automation and mobile applications can significantly reduce labor costs associated with plant care. Additionally, most respondents favored monitoring capabilities, as these features would enhance operational efficiency. Figure 1.2 illustrates the survey results regarding labor cost reduction when a smart hydroponic system mobile application is implemented. This survey underscores the importance of developing IoT-integrated hydroponic farming systems for individuals interested in hydroponic farming.

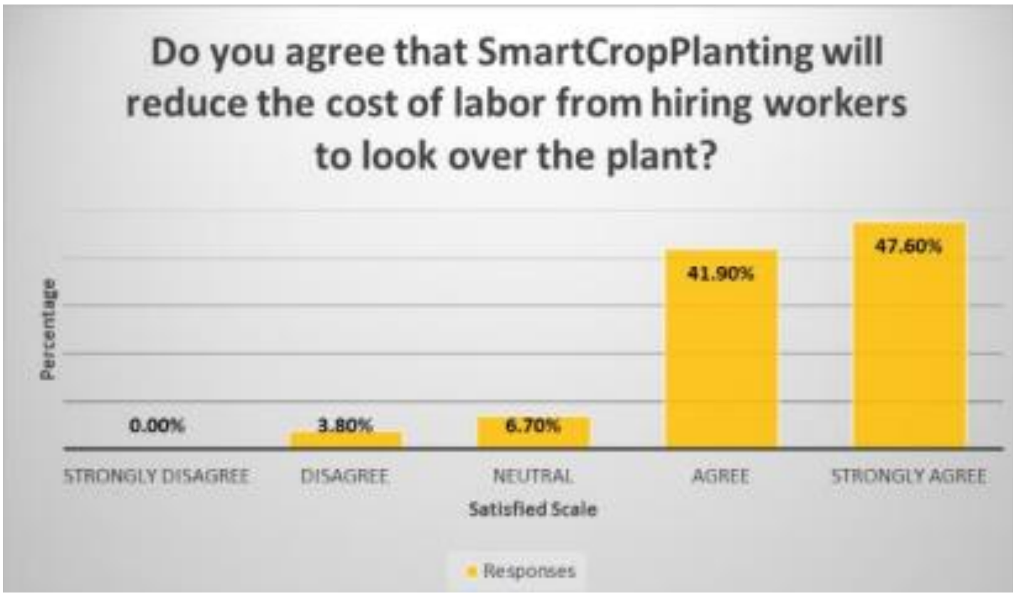

Figure 1.2 Survey Result (Cost) (Kyu, 2023)

Another preliminary survey by Hamdan et al. (2021) focused on the perceptions of low-income households in Selangor regarding various aspects of hydroponic farming, including needs, knowledge, cost, pricing, and benefits. The results indicated a preference for hydroponic systems priced below RM100, particularly for growing food plants such as Green Chili, Mint, and Soup Leaf, as these could help improve daily cooking and overall quality of life. This survey confirms the acceptance and preference for IoT-integrated hydroponic farming systems among Malaysians.

1.3 Problem Statement

1.3.1 Time Constraints and Monitoring Challenges in Urban Hydroponic Farming

Urban dwellers often face significant time constraints that hinder their ability to monitor hydroponic farming systems effectively. Consequently, there is a pressing need for tools that provide convenient, anytime access to crop status information. The development of software or applications that enable remote monitoring of hydroponic systems is crucial for addressing the concerns of busy urban farmers. Given that 98.7% of Malaysians use smartphones and 96.8% have internet access, there is a solid foundation for digital solutions in agriculture (Department of Statistics, Malaysia, 2022). The widespread use of social networks and digital content suggests a population well-acquainted with mobile technologies, indicating a high potential for the acceptance of hydroponic farming applications. Such an app would not only align with existing digital habits but also fulfil the need for accessible, real-time farm management tools.

1.3.2 Skills Gap and Technological Adoption in Malaysian Hydroponic Agriculture

The integration of Internet of Things (IoT) and Information Communication Technology (ICT) into hydroponic farming systems can significantly improve management efficiency. The slow adoption of advanced technologies within Malaysia's agriculture sector highlights a critical opportunity for innovation. A report by the Human Resource Development Fund (2019) identifies a deficiency in training and skill development among agricultural workers. This project aims to bridge this gap by employing IoT and ICT to simplify hydroponic farm management, thereby reducing the reliance on skilled labor and increasing technological integration in agriculture. IoT-enabled monitoring systems can record raw data, while the integration of IoT and ICT allows for real-time data collection, such as temperature and humidity. This data can be used to automatically adjust the indoor environment, creating ideal conditions for plant growth.

1.4 Aim and Objectives

The primary goal of this project is to develop a software solution that simplifies the management of hydroponic farming systems by providing real-time data and remote monitoring capabilities. This solution will help farmers optimize their operations and increase productivity. The objectives of the project are as follows:

- To develop mobile application capable of remotely monitoring and controlling hydroponic farming systems.
- To create a mobile application that alerts users to critical issues and tasks that need to be performed in hydroponic farming systems.
- To employ machine learning with IoT and ICT technologies for detecting normal and abnormal environmental patterns in hydroponic farming, enabling automated adjustments and simplifying management.

1.5 Scope and Limitations of Study

This project focuses on the development of a mobile application tailored to the needs of hydroponic farmers in Malaysia. The application will feature real-time remote monitoring and control of hydroponic farming systems. It will also leverage IoT devices and a cloud database to store vast amounts of data generated by IoT devices, using machine learning to optimize environmental settings. Additionally, an anomaly detection algorithm will be employed to detect irregular data in real-time.

However, the project will concentrate solely on software development and testing, excluding physical hardware implementation. Due to resource constraints, the project may not encompass all variations in hydroponic farming practices, and the effectiveness of the proposed solutions may vary based on specific environmental and operational factors. The hydroponic farming systems will upload data collected from IoT devices to a cloud database, from which the mobile application will retrieve data for display.

1.6 Proposed Solution

The proposed solution involves developing an Android-based mobile application for monitoring and controlling hydroponic farming systems. This application will be capable of real-time environmental monitoring. Figure 1.3 presents the layout of the proposed system. Data will be collected from sensors and transmitted to IoT device, which will then upload the data to a cloud database. The application will retrieve this environmental data from the server and display it to the user. Users can set preferred environmental parameters, such as temperature and humidity. Optimization will be achieved by controlling the fogger and nutrient solution dispenser through a IoT device. Additionally, the anomaly detection model will be deployed on the server.

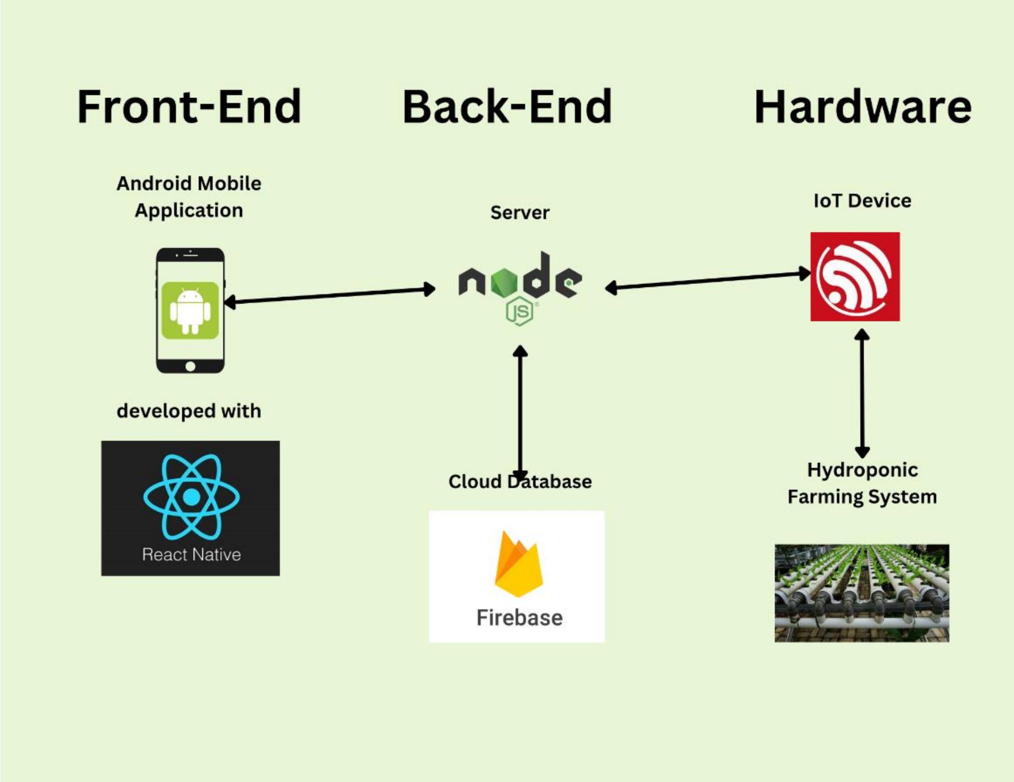

Figure 1.3 Layout of the Proposed System

1.7 Proposed Methodology

The Agile methodology is proposed for this project. Given the need for quick deployment to identify discrepancies between the actual system and requirements, Agile is well-suited for this project. It allows for the accommodation of new requirements discovered during deployment. The detailed project methodology will be discussed in Chapter 3.2.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter analyse the current state of Internet of Things (IoT) integrated hydroponic farming system mobile application and machine learning model. It aims to identify their features, capabilities, and limitation with focus of their ability to regulate environment parameters for plant growth. Additionally, the review of machine learning model aims to identify capabilities on providing support to mobile application feature for better monitoring experiences such as anomaly detection. By synthesizing findings from existing literature, this review aims to bridge the gap between technological potential and the practical needs for urban hydroponic farming, as outlined in Chapter 1.

2.2 Features and Capabilities of Hydroponic Farming System Mobile Applications

An exploration of existing research reveals advancements in IoT-integrated hydroponic applications with studies highlighting autonomous nutrient regulation, environment monitoring, and AI driven functionalities like plant disease classification and nutrient level prediction. For instance, Kularbphettong et al. (2019) developed a hydroponic farming system that capable of autonomously regulating nutrient levels using Message Queuing Telemetry Transport (MQTT) technology. Peuchpanngarm et al. (2016) incorporated various sensors for environment control and a Raspberry Pi2 controller, along with gardening planning and monitoring features into their application. This study has enlightened the importance of the scheduling of farming in our future implementation for users to provide more interactive farming experience. Furthermore, Khare et al. (2023) introduced plant disease classification and nutrient level prediction using deep learning models. Ramakrishnam Raju et al. (2022) also deployed deep learning convolutional neural network (DLCNN) to predict the nutrient level and plant disease. This indicates that Artificial Intelligence (AI) is the trend of providing more responsive and smart farming methods. Similarly, Kaur et al. (2022) focused on monitoring and controlling multiple environmental parameters using Arduino Mega, accompanied by a comprehensive mobile application which consists of climate component that display the environment parameters, nutrition component that display the status of nutrient solution and the image component that capture the condition of the farm. Moreover, Shin et al. (2024) developed a low cost IoT hydroponic setup with the capability of easily replicated, featuring the monitoring of environmental parameters and customization of these environmental parameters for different types of plants. Rahimi et al. (2022) developed a Multi Factor Authentication (MFA) functionality for an indoor hydroponic system mobile application to enhance the security when using the cloud database IoT platform.

From all the studies listed, several strengths and limitations have been identified in existing IoT-integrated hydroponic farming system mobile applications. The strengths include the planning components from Smart Suan Pak Nam by Peuchpanngarm et al. (2016) which it able to let users to setup their planting plan by specifying the number of units for their target, then the application generate a blueprint for actual gardening and record the harvest data for users next planting planning; the customization of environmental parameters based on types of plant by Shin et al (2024); the implementation of plant disease classification, and nutrient level prediction deep learning models by Khare et al. (2023); and the ability to view the hydroponic farming system via web camera by Kaur et al. (2022).While existing applications demonstrate significant strengths, such as planning components and advanced functionality for disease classification and nutrient prediction, they also exhibit limitations, including anomaly detection for overall status of hydroponic system and lack of comprehensive control algorithms. Addressing these limitations presents an opportunity for future development and innovation in this field.

While addressing the main features of the application, the monitoring feature of the hydroponic farming system was the essential and crucial feature that play as foundation of the application. Table 2.1 list out the environment parameters of every application reviewed. From the table, air temperature, humidity, potential of Hydrogen (pH) value, and light intensity were the parameters frequently measured in the hydroponic farm system while water temperature, total dissolved solids (TDS), and electrical conductivity (EC) are measured less frequently. TDS is the total amount of organic and inorganic substances contain in the water that are not dissolved as gases such as salts, metals, minerals, and ions (knowledge.hannainst.com, n.d.). EC is the concentration of conductive ion present which featuring greater salinity or dissolved solids (www.westlab.com, n.d.). The main differences between TDS and EC are the measurement on the water quality. TDS is measured on the dissolved substances while EC focuses on measured the substances' ability to conduct electricity. Thus, the parameters will be monitored are depends on the measure range requirements which decide what to cover in monitoring the hydroponic farm.

Table 2.1 Comparison of Feature of Hydroponic Farming System Mobile Applications

	Kularbphettong et al. (2019)	Peuchpanngarm et al. (2016)	Khare et al. (2023)	Kaur et al. (2022)	Shin et al. (2024)
Water Temperature	✘	✓	✓	✘	✘
Air Temperature	✓	✓	✓	✓	✘
Humidity	✓	✓	✓	✓	✘
pH Value	✓	✓	✓	✓	✓
Light Intensity	✓	✓	✓	✓	✘
Total Dissolved Solids (TDS)	✘	✘	✓	✓	✘
Electrical Conductivity (EC)	✘	✓	✘	✘	✓

2.2.1 User Interface Design

Effective user interface design is paramount in ensuring the accessibility and usability of hydroponic system mobile application. The Smart Suan Pak Nam application, as referenced by Peuchpanngarm et al. (2016) and VertiFarmControl application by Kaur et al. (2022), reveals that clean, intuitive layouts, and easy navigation significantly enhance user experience. These interfaces facilitate efficient farm management by providing clear insights into farming conditions, suggesting that a user-centric design approach is essential for successful application development. Figure 2.1 and figure 2.2 shows the user interface of the Smart Suan Pak Nam application. Figure 2.3 and figure 2.4 shows the user interface of VertiFarmControl application.

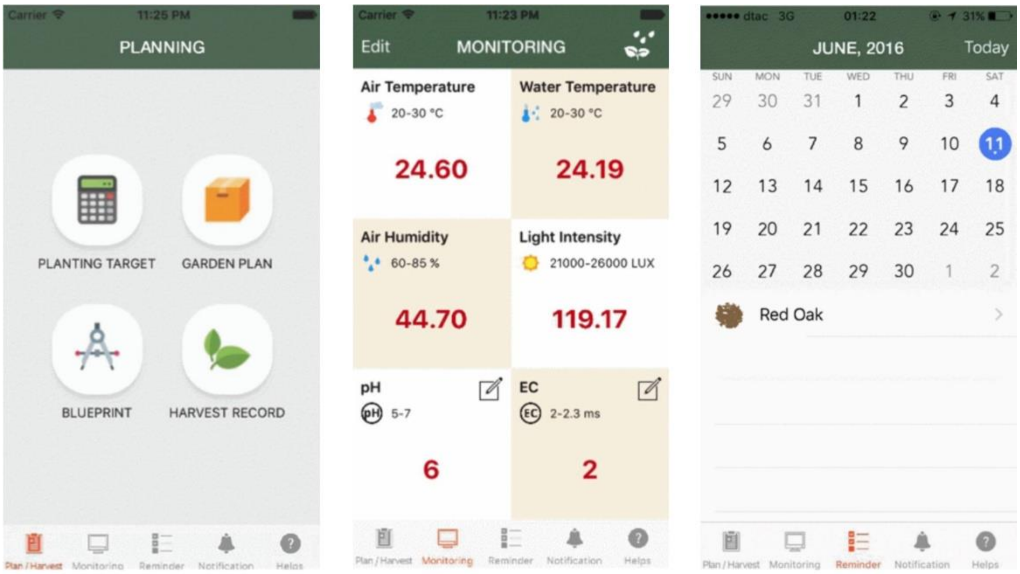

Figure 2.1 User Interface of Smart Suan Pak Nam (Peuchpanngarm et al., 2016)

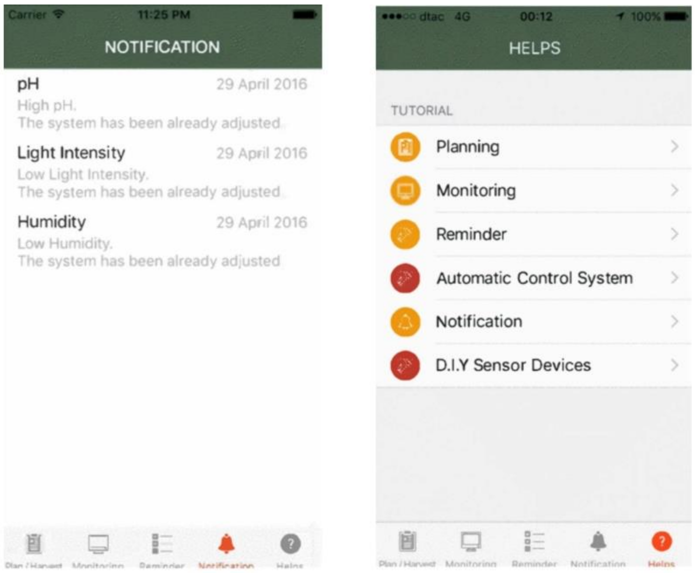

Figure 2.2 User Interface of Smart Suan Pak Nam (Peuchpanngarm et al., 2016)

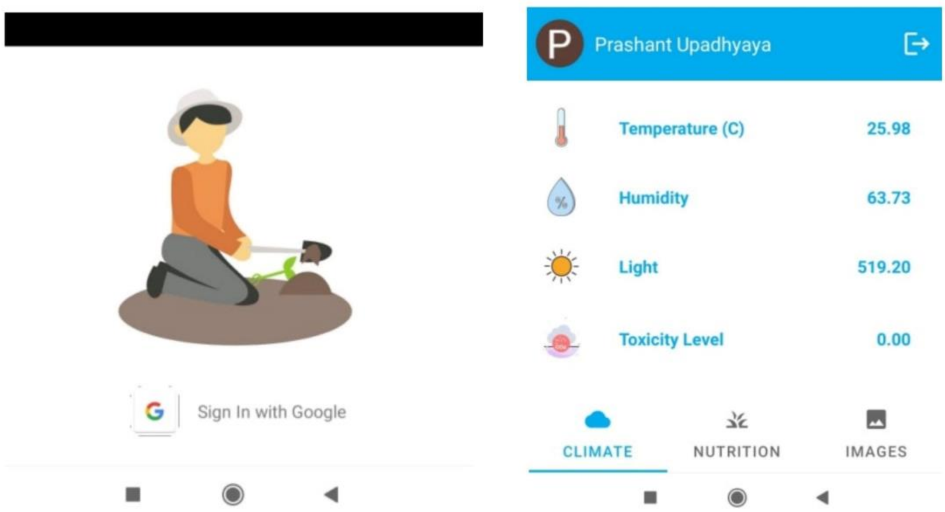

Figure 2.3 User Interface of VertiFarmControl (Kaur et al., 2022)

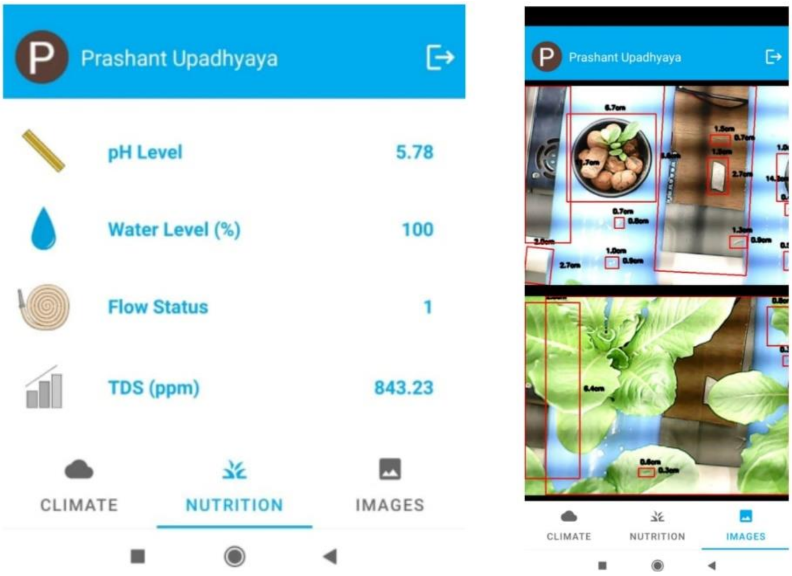

Figure 2.4 User Interface of VertiFarmControl (Kaur et al., 2022)

2.3 Machine Learning Model

This section explores on the machine learning model for providing real time data insight such as Long-Short-Term-Memory (LSTM), Isolation Forest, and One-Class Support Vector Machine (One-Class SVM). This section aims to identify AI models that providing anomaly detection feature to mobile application. This section also examines each model characteristics, abilities, and suitability for anomaly detection purpose. Furthermore, this section reviews several existing research about anomaly detection application in agriculture industry.

2.3.1 Long-Short-Term-Memory (LSTM)

According to Chugh (2019), Long-Short-Term-Memory is an improved version of recurrent neural networks (RNN) addressing the difficult to learn long-term dependencies. LSTM capable to learn long-term dependencies by introducing a memory cell to hold the information for an extended period. The memory cell consists of three gates which are the input gate for information insertion, the forget gate for information removal and the output gate for information output (GeeksforGeeks, 2019). Figure 2.5 shows the architecture of LSTM which include the forget gate, the input gate, and the output gate respectively. Table 2.2 discussed about the advantages and disadvantages of LSTM. The characteristics of LSTM enable it well-suited for tasks such as language translation, speech recognition, time series forecasting and anomaly detection (GeeksforGeeks, 2019).

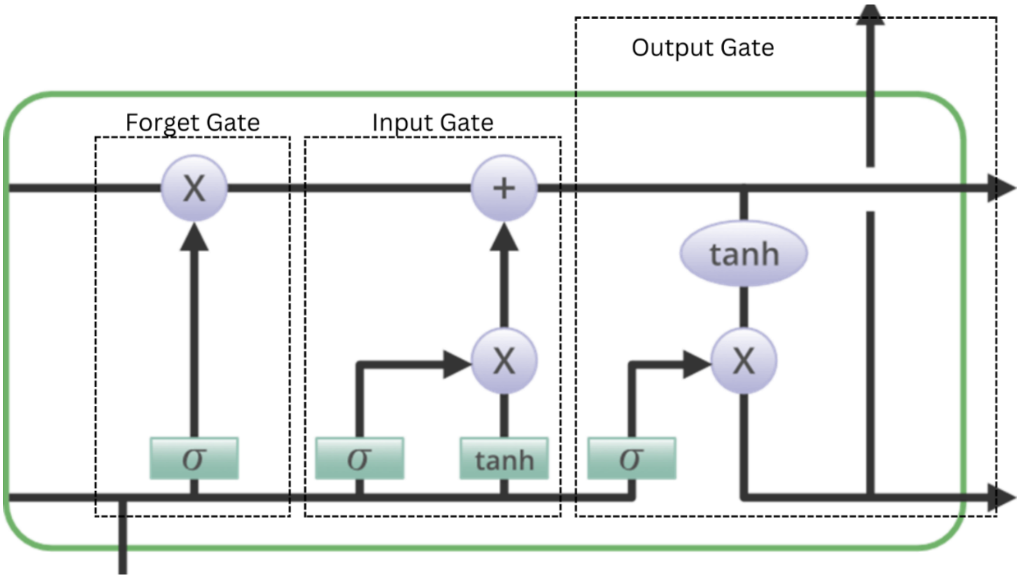

Figure 2.5 LSTM Architecture (GeeksforGeeks, 2019)

Table 2.2 Advantages and Disadvantages of LSTM

Advantages of LSTM	Disadvantages of LSTM
Capture long-term dependencies	Computationally more expensive
Selectively recalls or forget information	Training more time consuming
Capture important context even there is significant time gap between related events	Hard to parallelize work of processing the sentences

2.3.2 Isolation Forest

Isolation Forest is a model that introduces random partitioning the data recursively to isolate the anomalies instances instead of distance or density computations like traditional method (Liu et al., 2008). According to Liu et al. (2008), anomalies are "few and different," making them more prone to isolation than normal points. In isolation trees (iTrees), anomalies typically have shorter path lengths than normal instances. To perform anomaly detection, isolation forest model will plot out an ensemble of iTrees based on random sub-samples data, then exploiting shorter path lengths for anomalies due to the susceptibility. Through averaging the path lengths across ensemble of isolation trees, a scoring formula based on tree analysis will be used to obtain anomaly scores. Lastly, the final anomaly detection is ranked the instances based on anomaly scores. Table 2.3 listed out the advantages and disadvantages of isolation forest.

Table 2.3 Advantages and Disadvantages of Isolation Forest

Advantages of Isolation Forest	Disadvantages of Isolation Forest
Isolates anomalies instead profiling normal instances	May not perform well if anomalies not “few and different” from normal instances
Linear time complexity and low memory requirement	Requires tuning of sub-samples and number of trees for optimal performance
Alleviates the effects of swamping and masking by using small sub-samples data	May not be as effective as other methods for low-dimensional data with few irrelevant attributes.
Handle high-dimensional data	Does not provide a direct interpretation of the anomaly scores

2.3.3 One-Class Support Vector Machine (One-Class SVM)

One Class Support Vector Machine is a model designed to outlier, anomaly, or novelty detection but not for performing binary or multiclass classification tasks like other traditional machine learning model (GeeksforGeeks, 2024). The model key working principles are outlier boundary, margin maximization and high sensitivity. One-Class SVM define boundary around normal instances in the feature space to encapsulate the normal data points, then maximize the margin around the normal instances to separate the normal and anomaly data points. Furthermore, One-Class SVM consist of a hyperparameter, “nu” to represent upper boundary on the fraction of margin errors with support vectors, influences the model’s sensitivity to anomalies. Table 2.4 discussed the differences between support vectors machine (SVM) and One-Class SVM.

Table 2.4 Differences between support vectors machine (SVM) and One-Class SVM

Aspects	Support Vector Machine	One-Class Support Vector Machine
Single-Class Training	Requires labeled data from both classes for training	Operates with only the majority class during training
Imbalance Handling	Can't handle imbalance nature of datasets.	Inherently addresses class imbalance, prevalent in outlier detection tasks. By concentrating on the majority class during training.
Outlier Detection Focus	Only aims to find a hyperplane that best separates multiple classes.	Excels in scenarios where the goal is to uncover instances that deviate from the norm like in fraud detection or fault monitoring.

2.3.4 Anomaly Detection Model in Agricultural Industry

The purpose of Anomaly Detection is to identify rare events or observations that raise suspicious by being statistically different from the rest of the observation (GeeksforGeeks, 2019) The anomaly can be categorized to three types: point anomaly, contextual anomaly, and collective anomaly, difference are the point of view to the data which are tuple in a dataset, context of observations, and set of data instances respectively. The anomaly detection can be done in both supervised and unsupervised depend on the datasets and requirement. For example, Adkissson et al. (2021) proposed an anomaly detection model using unsupervised Autoencoder machine learning model to detect data discrepancies on environments condition for smart farming. Figure 2.6 shows the result of the autoencoder in anomaly detection. Bandar Alanazi and Ibrahim Alrashdi (2023) proposed Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) deep learning model anomaly detection to protect the smart agriculture system from network edge threats such as Distributed Denial of Service (DDoS) attacks by detect anomaly data transmitted from sensors device. Table 2.5 shows the result of the CNN-LSTM model in anomaly detection. Furthermore, Abdallah et al. (2021) explored the deployment of Machine Learning (ML) in digital agriculture using Autoregressive Integrated Moving Average (ARIMA) and LSTM models for predicting time series of sensor data then perform anomaly detection, found out that LSTM has better prediction performance on unseen dataset compared to ARIMA model. Moreover, Catalano et al. (2022) proposed an anomaly detection system for overcome infrastructure threats based on Multivariate Linear Regression (MLR) and LSTM algorithms, found out that LSTM results are closer to the actual observed data.

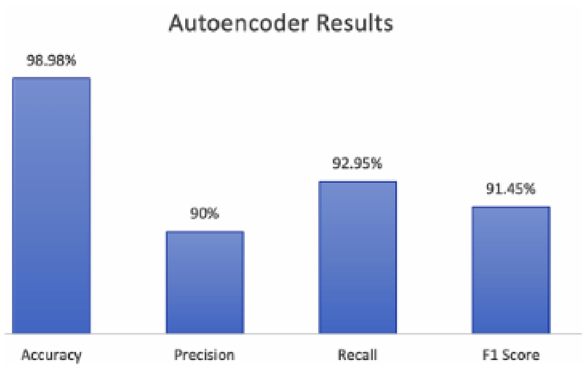

Figure 2.6 Performance Metrics for Autoencoder Model (Adkisson et al., 2021)

Table 2.5 Results of the LSTM-CNN model (Alazani et al., 2023)

Algorithms	Dataset	accuracy	Precision	Recall	F1 score
CNN-LSTM	IoT-23	0.9822	0.9538	0.9041	0.9215
	CICDDoS2019	0.9947	0.9541	0.9021	0.9191
CNN	IoT-23	0.8345	0.8821	0.9026	0.8846
	CICDDoS2019	0.8136	0.7861	0.8276	0.8585
LSTM	IoT-23	0.7988	0.8536	0.7642	0.8123
	CICDDoS2019	0.8349	0.8646	0.8262	0.7724

2.3.5 Justification on Machine Learning Model Selection

To provide precise and efficient anomaly detection in the hydroponic farming system, the machine learning model selection is essential. The Long Short-Term Memory (LSTM) neural network was chosen for this project after a number of models were evaluated because of its capacity to capture the long-term linkages and temporal dependencies present in time-series data. As opposed to conventional anomaly detection models like One-Class Support Vector Machine (SVM) and Isolation Forest, which lack of mechanisms to deal with sequential data, LSTM is specially made to be able to learn from the environmental factors' historical context. This feature allows the model to detect anomalies based on patterns and trends that develop over time as well as the specific values of sensor readings. Moreover, the gating mechanisms and memory cells of LSTM enable it to retain information over extended sequences, which makes it very useful for identifying minute variations in sensor data that can point to any problems with the hydroponic system. By utilizing recent prediction errors as a basis, the dynamic thresholding mechanism keeps anomaly detection flexible enough to adjust to shifting environmental circumstances, therefore decreasing false positives and enhancing detection precision. This flexibility is essential in a real-time monitoring application because system disruptions might cause sensor data to show different patterns. Since the objective of this research is to provide robust and reliable anomaly detection for optimizing the hydroponic farming environment, LSTM was selected as the most suitable model.

2.4 Summary

Several existing studies have explored various features and capabilities of a hydroponic farming system mobile application, such as autonomous nutrient regulation, environmental monitoring, gardening planning, plant disease classification and nutrient level prediction using deep learning models. Furthermore, user interface design demonstrated in the previous studies have underscores the importance of clean, intuitive, easy navigation, and consistent visual aesthetics in ensuring the user experience and accessibility. Despite the strengths, there were limitations could be found, such as anomaly detection and lack of comprehensive control algorithms.

In addition, there are various machine learning models to perform anomaly detection for real time farming datasets such as LSTM, Isolation Forest, and One-Class SVM. After reviewing several existing journals related to anomaly detection machine learning model in agriculture industry, LSTM model is mentioned relatively more numbers than other machine learning models, it features high accuracy on predicting the environment parameters, and reliable in learning new data collected from the IoT devices.

In synthesizing the research, it's evident that while current applications excel in certain areas, they lack in customizable control and anomaly detection (Khare et al., 2023; Shin et al., 2024). The integration of such algorithms, alongside LSTM's data interpretative strength, could significantly propel the functionality of the proposed mobile application. This chapter lays the groundwork for developing a solution that not only addresses the identified gaps but also leverages advanced IoT and ICT solutions, echoing the objectives set out in Chapter 1

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter will discuss the suitable system development methodology for this project by reviewing various methods from online resources and the decision of methodology will be elaborated. The machine learning will discuss the datasets used, the training details and procedure. The development flow of the project will be listed detailed in the Work Plan Section. In the Work Plan Section, Work Breakdown Structure and Gantt Chart are provided to define the tasks details and schedule. Finally, development tools for this project will be elaborated in this chapter.

3.2 System Development Methodology

To select the most appropriate development methodology, review on existing methodology need to be included. The methodologies will be discussed are waterfall, unified process, lean development, and feature driven development. Each methodology advantages and disadvantages will be compared to each other to decide whether which methodology are suitable for this project. This section also elaborates the activities performed in each development phases.

3.2.1 Waterfall Methodology

The Waterfall methodology is a linear and sequential approach to project management which based on fixed requirements, flows, testing and output (Gallagher, Dunleavy and Reeves, 2019). This methodology does not require much communication between stakeholders but only approval from stakeholders to continue to next stage. The lack of communication lets the project consists of limitations and problem in delivering a good quality software to stakeholders. This also increases the development cost due to the long duration of the project, and unnecessary functionality in the software. Below table 3.1 listed out the advantages and disadvantages of Waterfall methodology. In contrast, this methodology is suitable for project that have constant project scope and minimal changes to requirements while also have sufficient budget and time.

Table 3.1: Advantages and Disadvantages of Waterfall methodology (Gallagher, Dunleavy and Reeves, 2019)

Advantages	Disadvantages
Static project scope	Hard to add new requirements
Minimal changes to system	Dependencies on relatively unstable products
Easy to plan the tasks	Difficult to estimate total time project complete
Reduce impact from the leave of key members	Large contingency during development

3.2.2 Unified Process (UP)

The Unified Process methodology is an incremental and iterative approach to project management that emphasizes teamwork, producing usable software increments, and adapting to changes. (GeeksforGeeks, 2024). Figure 3.1 shows the flow of Unified Process. This is based on Unified Modeling Language (UML) and is a use case driven development. Its focus on architecture design enables it more suitable for complex project. This methodology requires clear guidelines and workflows to enhance the feedback and communication from stakeholders which also ensure the quality of the project outcomes. However, it needs to have solid understanding on the principles which increase the learning curves of this methodology. Table 3.2 discusses the advantages and disadvantages of Unified Process.

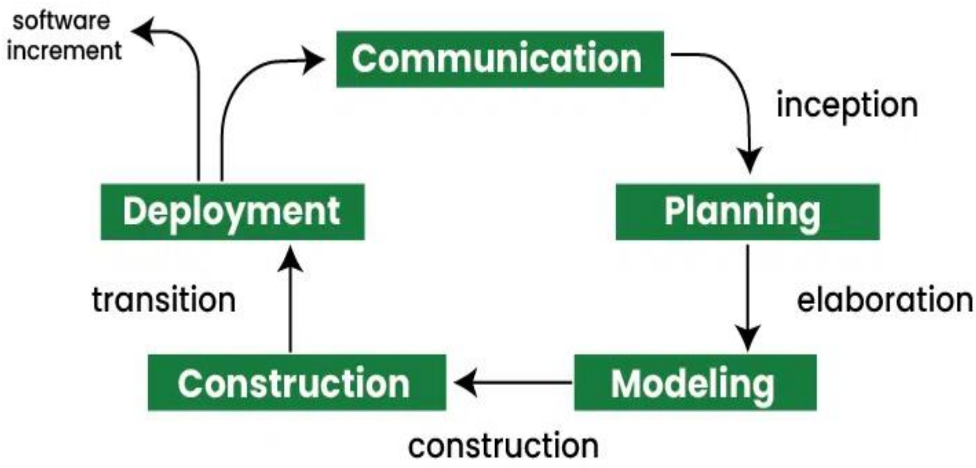

Figure 3.1: Unified Process (GeeksforGeeks, 2024).

Table 3.2 Advantages and Disadvantages of Unified Process (GeeksforGeeks, 2024)

Advantages	Disadvantages
Iterative development	Complexity
Risk Management	Overhead in documentation and formalized processes
Quality Assurance	Longer Learning Curve
Stakeholder Collaboration	Scope Management
Flexibility and Adaption	Adoption Resistance from stakeholders or team members

3.2.3 Lean Development

The Lean Development is a continuous improvement approach to project management focusing on efficiency and waste reduction on time and resources in software development (Developer.com, 2022). This methodology also focuses on the collaboration which the teams consist of cross-functional members to achieve the goals of the project. The characteristics of this methodology are short cycles development, focus on customer value, collaboration, minimize waste and learn from errors actively. Figure 3.2 shows the flow of Lean Development. This methodology suitable for stakeholders that prefer fast development with minimum resources consumed, and improvement based on validation from customer feedback. Table 3.3. listed out the advantages and disadvantages of Lean Development.

Figure 3.2 Lean Software Development (trident, 2021)

Table 3.3 Advantages and Disadvantages of Lean Development

Advantages	Disadvantages
Increase Efficiency by focus on essential tasks	Lead to "ship it now, fix it later" mindset which may cause low quality
Improved Quality	Relies heavily on customer feedback
Increase Customer Satisfaction	Change on team's work habits
Improve morale by streamlining development process	

3.2.4 Feature Driven Development

The Feature Driven Development is an iterative approach in project management with mixture of different Agile approach practices, it more focused on the exact features of a software to develop. This methodology relies heavily on customer input, as the software features are defined by the customer (Laoyan, 2022). This methodology has four main values: Individuals over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiations, and responding to change over following a plan (Laoyan, 2022). Figure 3.3 visualize the flow of Feature Driven Development. This methodology suitable for project that need to iterate rapidly based on feature require by customers and for the project leader that have clear vision on the software development. Table 3.4 listed out the advantages and disadvantages of Feature Driven Development.

Figure 3.3 Feature Driven Development (www.productplan.com, n.d.)

Table 3.4 Advantages and Disadvantages of Feature Driven Development (www.productplan.com, n.d.)

Advantages	Disadvantages
Simple five-step process enable for rapid development	Does not work efficiently with small projects
Allows larger teams to move product forwards with continuous success	Less written documentation
Leverages pre-defined development standards	High dependant on lead developers or programmers

3.2.5 Comparison among SDLC methodologies

To achieve the goals of the project, comparison among methodologies is performed to determine the most appropriate software development approach. Table 3.5 compares the four methodologies reviewed in this chapter. The criteria used to determine the most appropriate software development are the focus, flexibility, result delivery, risk factors, and customer feedback. Based on the comparison, Feature Driven Development (FDD) is more suitable for this project than other methodologies. This is due to the development of hardware components and mobile applications for hydroponic farming system are developed by separate teams. As a result, a clear and structured process is needed for developing the application to ensure the deliverables of software features. Although the flexibility of this approach might not higher than Lean Development and Unified Process but the adaptability to changing project requirements also enable this project respond to new requirements in short period. The adaptability is important because the app relies on real-time data retrieve from the sensors, while it depends on the app to provide control and monitoring functionality which need adjust based on the situation. By employing FDD's iterative approach with frequent delivery cycles, the mobile application can be developed to meet all required features while maximizing the use of data from the hydroponic system sensors. Moreover, FDD's emphasis on delivering working features incrementally aligns well with the need for continuous communication and feedback between the hardware sensor and mobile app development teams. In summary, with the feature driven approach, adaptability, and the constant feedback from stakeholders, ensure the project objectives and scope to be achieved.

Table 3.5 Comparison of Software Development Life Cycle methodologies

Methodology / criteria	Waterfall	Unified Process (UP)	Lean Development	Feature Driven Development
Focus	Fixed requirements, flows, testing, output	Creating working software increments, collaborating, adapting	Efficiency and waste reduction	Exact features development based on customer input
Flexibility	Limited	Excellent	Excellent	Excellent
Result Delivery	Delayed	Medium	Frequent	Medium
Risk Factor	High	Low	Low	Low
Customer Feedback	Low	High	High	High

3.2.6 Activities in Each Phase

The first phase for the project is initiation planning, which includes identifying the problem statement, defining the aim, objectives, and scope of the project. The aim and objectives are defining based on the problem statement while the problem statement is identified based on the online research via journal, article, and public data. This phase also determines the scope and the limitations of this project to ensure the range of the deliverables. After these preparations, this phase will come out a simplify proposed solution and methodology for this project.

The next phase will be the performing the requirements collection and complete the software design of the project. During this phase, requirements define, use case design, software design including the user interface and system architecture will be performed. The requirements define is based on the module identified and analysed in the early phase. For the use case design will be including the use case diagram with use case description to better visualize the interactions between users and system. The user interface design will be based on the use case design to ensure providing a more intuitive and professional experience to the users. For the tools will be use in this phase are Enterprise

Architect (EA) for drawing the use case diagram, while Figma for drawing the user interface design for this project. After these have been completed, the requirements and design will be reviewed for verification to ensure the good quality of the project.

After the review of the requirements and software design, there will be correction based on the feedback collected. New iteration will be initiated based on updated requirements and design which is the module development. The first module is the monitoring and controlling module which represents the data presentation and parameters control of the hydroponic system. This module includes the setup of the database for the hydroponic system and the mobile application to collect the data retrieved from the sensors. The outcome for this module is the app able to display the data fetch from the database and database respond to the request from the app. Functional testing will be performed continuously during the development of the module. After the completion of the main module, other modules will be initiated like reminders and notifications, listing of plants details, and user authentication and management. The reminders and notification will remind and push notification to the users on the tasks, anomaly alerts and daily reminders about the hydroponic system. The plant module handles the farm plant management. The user authentication and management are for the security of the application and maintain the session.

Next iteration is about the anomaly detection machine learning model for better detection on the anomaly data via the database. This model is to enhance the monitoring of the environment parameters of hydroponic system by making predictions based on real time data. After fine-tuning of the model and testing, it will be integrated into the main module to support features such as reminder module trigger and preparing deployment to the server.

The last iteration is the integration testing and performance testing. The integration testing will be run for all modules to test on the interaction between modules. While performance testing will focus on evaluating the system efficiency and stability. After all the testing performed and bug fixed, the final product will present to the user indicates that the development is complete and able to deploy to production environment.

3.3 Machine Learning

This project will leverage machine learning to enhance the mobile application's capability to monitor hydroponic system effectively. The Long Short-Term Memory (LSTM) model is selected, as justified in Chapter 2.3.1 and Chapter 2.3.4, for its proficiency in handling time-series data crucial for predicting environmental parameters in hydroponic farming.

3.3.1 Datasets Selection and Preparation

The LSTM model will predict key environmental parameters outlined in Table 3.6, drawing on methodologies from Khare et al. (2023). Comprehensive training will also incorporate sensor value and timestamps to improve model accuracy. The selection of features for training is detailed in Table 3.6.

Table 3.6 Environmental Parameters

No.	Environmental Parameters
1.	Surrounding Temperature
2.	Surrounding Humidity
3.	Light Intensity
4.	pH Level
5.	Total Dissolved Solids (TDS)
6.	Solution Temperature
7.	Low pH Trigger
8.	High pH Trigger
9.	Low TDS Trigger
10.	High TDS Trigger
11.	Fogger Trigger
12.	Fogger Temperature
13.	Fogger Humidity

Table 3.7 Datasets Features

No.	Datasets Features
1.	Environmental Parameters
2.	Timestamps

Data will be sourced from the hydroponic system developed by Chua Shi Jian, an Electrical Electronic System (3E) student responsible for the design and operation, the data will continuously be uploading to the Firebase database after deployment.

Data preparation involves multiple stages to ensure quality and consistency:

Data Validation: Ensuring data conforms to type, range, and presence requirements (Bhandari, 2021).

Data Screening: Identifying and removing inconsistent, missing, or outlier data using manual and statistical methods (Bhandari, 2021).

Data Cleaning: Eliminating duplicates and correcting invalid data entries.

Data Normalization: Utilizing MinMaxScaler or StandardScaler techniques to standardize data values, facilitating more effective training (Brownlee, 2016). MinMaxScaler normalize data by rescaling values between range of 0 and 1 as shown in Equation 3.1 while StandardScaler as shown in Equation 3.2, normalize data by subtracting the mean value.

Post-Preparation: Data is split into training and validation sets with an 80/20 ratio, as visualized in Figure 3.4, to optimize learning outcomes and model validation.

$$y = \frac{(x - min)}{(\max - \min)} \tag{3.1}$$

where

min = Minimum observable value

max = Maximum observable value

$$y = \frac{(x-\mu)}{\sigma} \tag{3.2}$$

where

μ = mean of observable value

σ = standard deviations of observable value

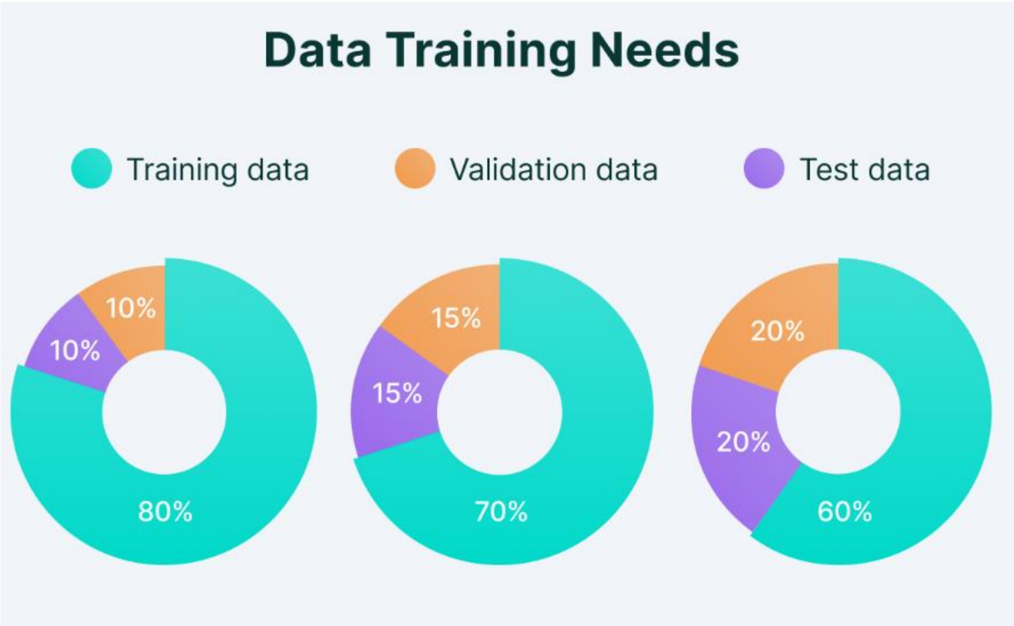

Figure 3.4 Data Training Needs (Baheti, 2021)

3.3.2 Model Training

The chosen LSTM model will be trained using Backpropagation Through Time (BPTT) to address long-term dependencies and sequence-related challenges in machine learning. TensorFlow will serve as our primary tool for model training.

The model architecture will consist of several layers, with specific units per layer, activation functions, and dropout layers to mitigate overfitting. Tuning will be conducted to determine the optimal architecture configuration.

The optimization algorithm will be needed to adjust the weights and biases of the model. It can minimize the error between the predicted and actual values which ensure the accuracy of model. The optimization algorithm can be used to train LSTM models are Bayesian optimization, Sine Cosine Algorithm, Harmony Search and Gray Wolf Optimizers (Rashid et al., 2018). The choice of optimization algorithms will depend on the datasets features and requirements.

Model Performance will be assessed using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), providing insight into average prediction deviations and variability. The MAE derived by calculating the average difference between the predicted value and the actual value, as shown in Equation 3.3.

$$MAE = \frac{1}{n}\Sigma|y - \widehat{y}| \tag{3.3}$$

where
n = Total number of data points
y = Actual output value
$\hat{y} =$ Predicted output value

The RMSE is determined by calculating the square root of difference between square of predicted value and actual value shows in Equation 3.4. The finalized model will be deployed to a server for real-time predictions and to monitor anomalies in the collected data.

$$RMSE = \sqrt{\sum \frac{(\hat{y}-y)^2}{n}} \tag{3.4}$$

where
n = Total number of data points
y = Actual output value
$\hat{y} =$ Predicted output value

3.4 Work Plan

This section includes the Work Breakdown Structure and Gantt Chart. The Work Breakdown Structure will list out the tasks of each phase. While the Gantt Chart will provide each tasks' duration.

3.4.1 Work Breakdown Structure

IoT-Integrated Hydroponic Farming System Mobile Application

1.0 Initial Planning

1.1 Project Planning

1.1.1 Project Background Research

1.1.2 Define Problem Statement

1.1.3 Define Project Objectives

1.1.4 Define Scope and Limitation

1.1.5 Define Solution and Approach

1.2 Literature Review

1.2.1 Review on Features and Capabilities of Hydroponic Farming System Mobile Applications

1.2.2 Review on Mobile Applications User Interface Design

1.2.3 Review on Proportional-Integral-Derivative (PID) Control Algorithm

1.2.4 Review on Machine Learning Model

1.3 Methodology and Work Plan

1.3.1 Compare and Select SDLC methodology.

1.3.2 Develop Work Plan

1.3.3 Determine Development Tools

2.0 Execution

2.1 First Iteration

2.1.1 Requirement Collection and Analysis

2.1.2 Design User Interface

2.1.3 Design System Architecture

2.1.4 Develop Use Case Diagram

2.1.5 Evaluation and Feedback

2.2 Second Iteration

2.2.1 Develop Monitor and Control Module

2.2.2 Develop Reminder and Notification Module

2.2.3 Develop Plant Listing and Detail Module

2.2.4 Develop User Authentication and Account Module

2.2.5 Evaluation and Feedback

2.3 Third Iteration

2.3.1 Preprocessing Datasets

2.3.2 Training Model

2.3.3 Fine-tuning and Testing Model

2.3.4 Integrate Machine Learning Model with Hydroponic Farming System

2.3.5 Evaluation and Feedback

2.4 Fourth Iteration

2.4.1 Develop Testing Plan

2.4.2 Develop Test Case

2.4.3 Integration Testing

2.4.4 Performance Testing

2.4.5 Evaluation and Feedback

3.0 Closure

3.1 System Deployment

3.2 Report and Documentation

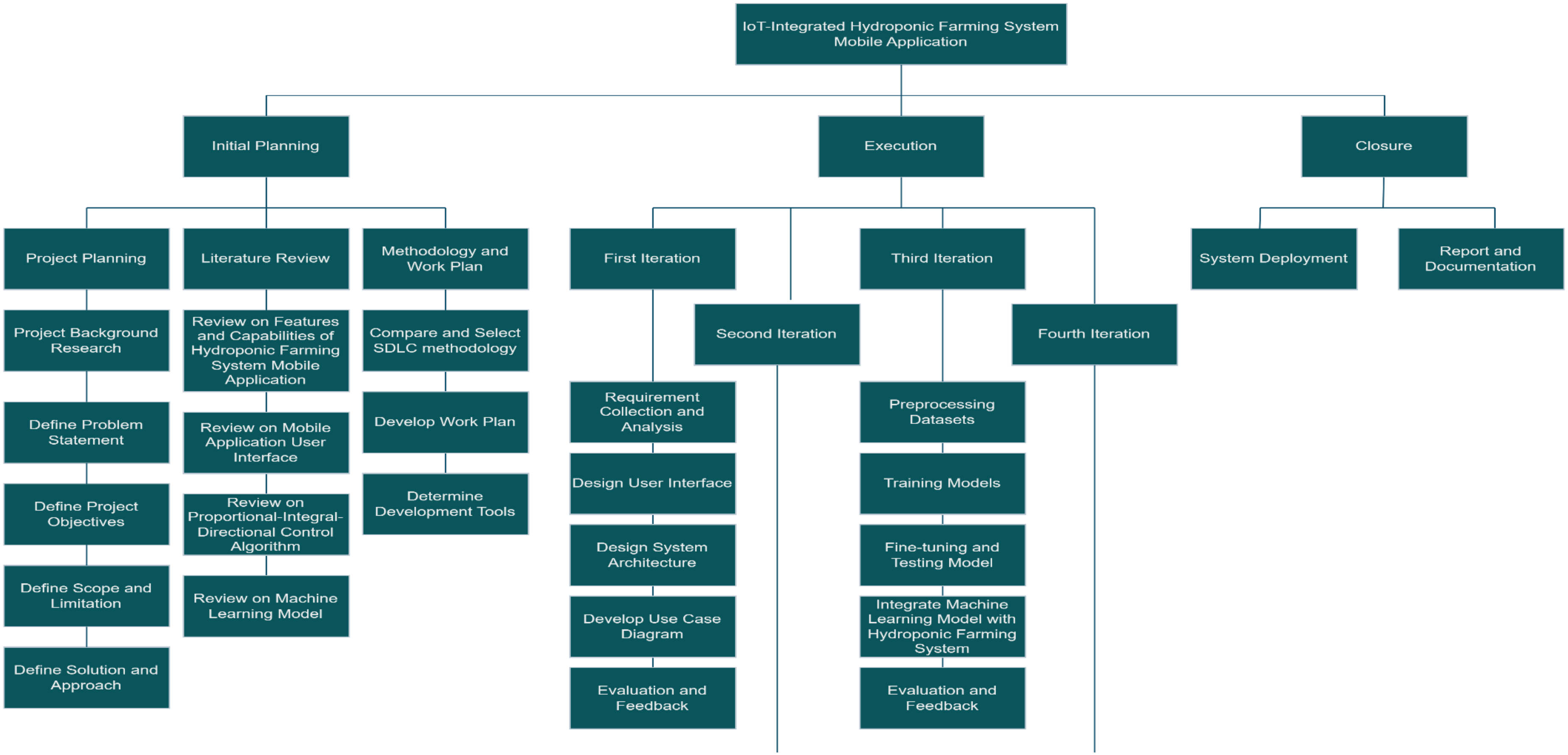

Figure 3.5 Work Breakdown Structure

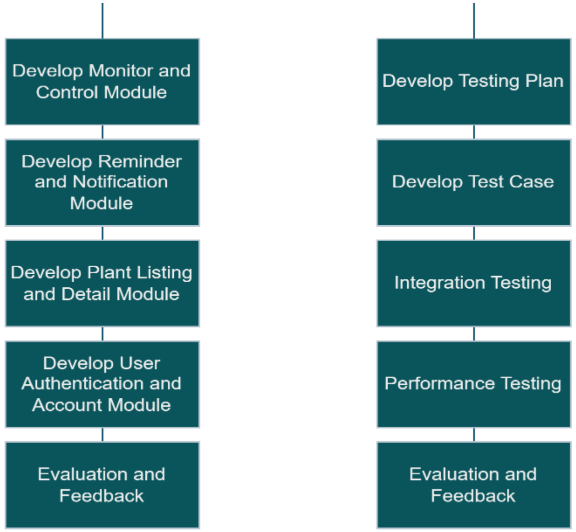

Figure 3.6 Work Breakdown Structure (continued)

3.4.2 Gantt Chart

No	Task Name	Duration (days)	Start Date	End Date
1	Initial Planning	67	29/1/2024	5/4/2024
1.1	Project Planning	31	29/1/2024	29/2/2024
1.1.1	Project Background Research	18	29/1/2024	16/2/2024
1.1.2	Define Problem Statement	4	16/2/2024	20/2/2024
1.1.3	Define Project Objectives	2	20/2/2024	22/2/2024
1.1.4	Define Scope and Limitation	4	22/2/2024	26/2/2024
1.1.5	Define Solution and Approach	3	26/2/2024	29/2/2024
1.2	Literature Review	18	29/2/2024	18/3/2024
1.2.1	Review on Features and Capabilities of Hydroponic Farming System Mobile Applications	3	29/2/2024	3/3/2024
1.2.2.	Review on Mobile Applications User Interface Design	5	3/3/2024	8/3/2024
1.2.3	Review on Proportional-Integral-Directional (PID) Control Algorithm	5	8/3/2024	13/3/2024
1.2.4	Review on Machine Learning Model	5	13/3/2024	18/3/2024
1.3	Methodology and Work Plan	18	18/3/2024	5/4/2024
1.3.1	Compare and Select SDLC methodology	5	18/3/2024	23/3/2024
1.3.2	Develop Work Plan	6	23/3/2024	29/3/2024
1.3.3	Determine Development Tools	7	29/3/2024	5/4/2024
2	Execution	32	5/4/2024	7/5/2024
2.1	First Iteration	38	5/4/2024	13/5/2024
2.1.1	Requirement Collection and Analysis	7	5/4/2024	12/4/2024
2.1.2	Design User Interface	7	12/4/2024	19/4/2024
2.1.3	Design System Architecture	6	19/4/2024	25/4/2024
2.1.4	Develop Use Case Diagram	6	25/4/2024	1/5/2024
2.1.5	Evaluation and Feedback	6	1/5/2024	7/5/2024

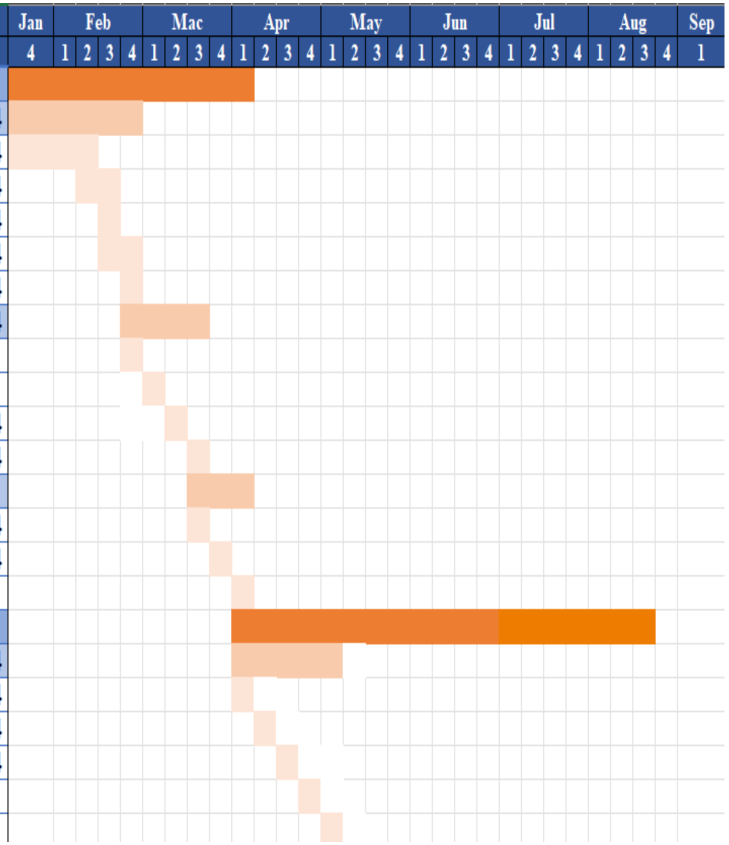

Figure 3.7 Gantt Chart

2.2	Second Iteration	38	7/5/2024	14/6/2024
2.2.1	Develop Monitor and Control Module	9	7/5/2024	16/5/2024
2.2.2	Develop Reminder and Notification Module	9	16/5/2024	25/5/2024
2.2.3	Develop Plant Listing and Detail Module	9	25/5/2024	3/6/2024
2.2.4	Develop User Authentication and Account Module	6	3/6/2024	9/6/2024
2.2.5	Evaluation and Feedback	5	9/6/2024	14/6/2024
2.3	Third Iteration	35	14/6/2024	19/7/2024
2.3.1	Preprocessing Datasets	6	14/6/2024	20/6/2024
2.3.2	Training Models	4	20/6/2024	24/6/2024
2.3.3	Fine-tuning and Testing Model	13	24/6/2024	7/7/2024
2.3.4	Integrate Machine Learning Model with Hydroponic Farming System	7	7/7/2024	14/7/2024
2.3.5	Evaluation and Feedback	5	14/7/2024	19/7/2024
2.4	Fourth Iteration	34	19/7/2024	22/8/2024
2.4.1	Develop Testing Plan	5	19/7/2024	24/7/2024
2.4.2	Develop Test Case	5	24/7/2024	29/7/2024
2.4.3	Integration Testing	12	29/7/2024	10/8/2024
2.4.4	Performance Testing	7	10/8/2024	17/8/2024
2.4.5	Evaluation and Feedback	5	17/8/2024	22/8/2024
3	Closure	14	22/8/2024	5/9/2024
3.1	System Deployment	7	22/8/2024	29/8/2024
3.2	Report and Documentation	7	29/8/2024	5/9/2024

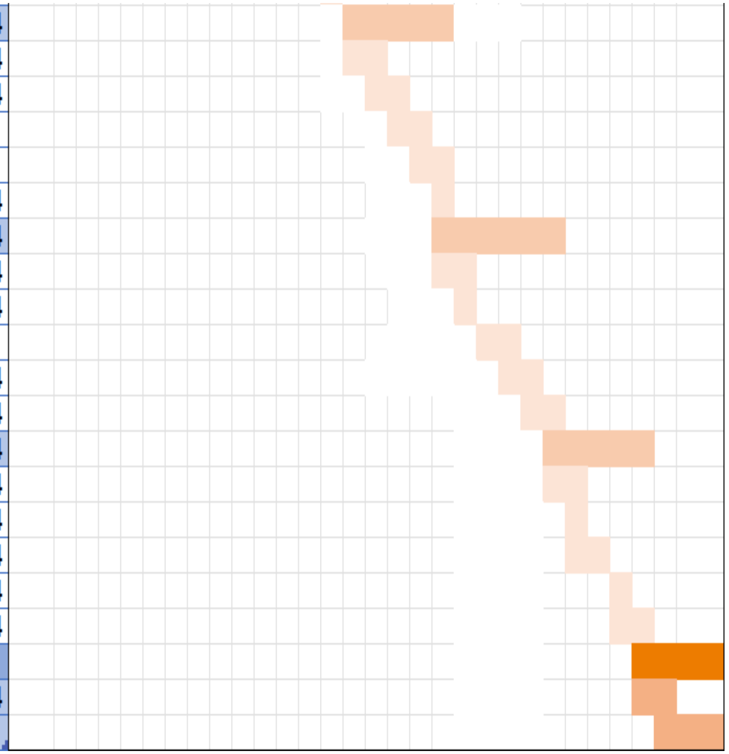

Figure 3.8 Gantt Chart (continued)

3.5 Development Tools

This section introduces the tools will be used for developing this project including programming languages, frameworks, integrated development environments (IDE), version control systems, and database. This project mainly using Android Studio and Visual Studio Code to develop the application while Google Collab for developing the anomaly detection model.

3.5.1 JavaScript

This programming language are used as the default language of React Native framework. This language has many developed and basics libraries can be used in this project like user interface (UI) libraries. It is a language that suitable for this project frontend and backend development.

3.5.2 React Native

This framework is a JavaScript framework for developing cross-platform mobile applications. It allows the project can be deployed in multiple platforms such as Android and iPhone Operating System (iOS). React Native can import large numbers of third-party libraries and community support for efficient development.

3.5.3 Visual Studio Code

This IDE is a lightweight and powerful tool to develop different kinds of web and mobile project. Its capabilities of importing extension and module enhance the productivity for the project. Thus, it is suitable for the project using JavaScript and React Native.

3.5.4 Android Studio

This IDE is the official tool for developing Android applications. It offers efficient tool for building and testing the Android app, the emulators a virtual device to simulate the environment of smartphones. It also ensures the compatibility and performance of the Android app for easier developing and debugging.

3.5.5 Google Collab

This online tool is a cloud-based platform for developing the machine learning model that will deploy in this project. It provides free access to Graphics Processing Unit (GPU) resources which reduce the costs and time to train the model. It reduces the difficulty of developing and testing the machine learning model for this project.

3.5.6 Git

Git is a version control system that used for tracking changes in source code during project development. It can sync with GitHub for backing up the source code to avoid accident loses. It enables the project can merge the changes or revert to previous versions if occur incompatible of dependencies or other issues.

3.5.7 Firebase

Firebase is a No Structured Query Language (NoSQL) database which increase the efficiency of retrieving time-stamp data. It also provides real-time database for real-time synchronization between mobile applications and server. It will simplify the backend development and management of this project, increase the productivity of the development.

3.5.8 Node,js

Node.js is a JavaScript runtime environment can be used for developing server-side scripting. It is an event-driven architecture, featuring of asynchronous input and output, and single-threaded design enabled high efficiency development. It also provides frameworks that simplify the development process of mobile applications.

3.5.9 Flask

Flask is a lightweight web framework for Python, designed to make web applications and API easy to build. Its characteristics of microframework and simplicity let it easy to use and understand which using less code to write and suitable for developers. This framework provides URL mapping to functions which allow handling of different HTTP requests such as GET and POST.

3.5.10 TensorFlow

TensorFlow is an open-source machine learning framework developed by Google. This framework is widely used for building and training machine learning and deep learning models. The Keras API is one of the key features for TensorFlow to build deep learning models and promote quick model development including packages such as layers, optimizers, regularizers and others.

3.5.11 NumPy

NumPy is a Python fundamental library for numerical and scientific computing which suitable for processing the raw data. It supports data processing for arrays, matrix, and many mathematical functions such as arithmetic, statistical analysis, and linear algebra. It also integrates well with other scientific libraries like scikit-learn that would also be used in this project.

3.5.12 Scikit-learn

In this project, Scikit-learn, an open-source machine learning framework for Python that offers effective and simple to use data analysis and modeling capabilities was used. It is constructed upon NumPy, SciPy, and matplotlib, which are data science-specific libraries. The transformers for this library will be used for preprocessing and feature extraction such as StandardScaler and MinMaxScaler.

3.6 Conclusion

This chapter decides the Feature Driven Development (FDD) software development methodology as the project development methodology. Additionally, steps to process the datasets and model training were identified and discussed. Moreover, this chapter also discussed the tasks to be done in the work breakdown structure while the time schedule mentioned in the Gantt Chart. The tools for developing the project include JavaScript, React Native, Visual Studio Code, Android Studio, Google Collab, Git and Firebase. In conclusion, by utilizing the FDD methodology with the tools identified, the project could deliver efficient solution to meet the requirements and challenges of the development process.

CHAPTER 4

PROJECT SPECIFICATION

4.1 Introduction

This chapter will introduce the requirements specification for the application including functional and non-functional requirements. The use case diagram and description provide a visualization and explanation of the main functionality of the application. All these information act as a foundation to the application architecture design and development.

4.2 Requirement Specification

Requirement Specification will define what features or functionalities include in this project. The requirements were collected through review of feature and capabilities of hydroponic farming system mobile application and based on the project objectives.

4.2.1 Functional Requirements

Monitoring and Controlling Module

- The application shall allow users to monitor real-time data from sensor in the hydroponic system.
- The application shall display environmental parameters such as temperature, humidity, and pH levels.
- The application shall allow users to adjust parameters range remotely.
- The application shall display the tasks to do that user set for their plants.

Reminders and Notifications Module

- The application shall push reminder of tasks set by user based on user's notification settings.
- The application shall push notification to alert users about critical issues or changes detected by Anomaly Detection Module based on user's notification settings.
- The application shall allow users to view all the notification.

- The application shall allow users to configure notification settings on reminders and alerts.

Plant Management Module

- The application shall allow users to add the plant details including image, status, observation, and measurement.
- The application shall allow users to edit the plant details.
- The application shall display a listing of plants of the farm.
- The application shall allow users to add observation and measurement for record purposes.
- The application shall allow users to add tasks with date that need to be implement on plants.

User Authentication and Management Module

- The application shall allow users to register a new account.
- The application shall allow users to login via email and password.
- The application shall send a verification email after users register an account.
- The application shall allow users to edit their account credentials include email and password.
- The application shall allow users to recover account by resetting password via email.

Anomaly Detection Module

- The model shall perform real-time analysis of data from sensors to detect anomalies in environmental parameters.
- The model shall identify abnormal patterns or deviations from the data collected.
- The application shall allow users to review data and insight of the current farm.

4.2.2 Non-Functional Requirements

- The application shall achieve a response time of less than 5 second for displaying real-time data, measured consistently during operation.
- The application shall maintain a minimum uptime of 99% for continuous hydroponic system operation, tracked through uptime monitoring, and alert on anomalies.
- The user interface of application shall be intuitive, require minimal learning for operation.
- The application shall be able to adapt to different screen sizes of mobile devices.

4.3 Use Case Diagram

Figure 4.1 shows the use case diagram for Internet of Things (IoT) integrated hydroponic farming system mobile application.

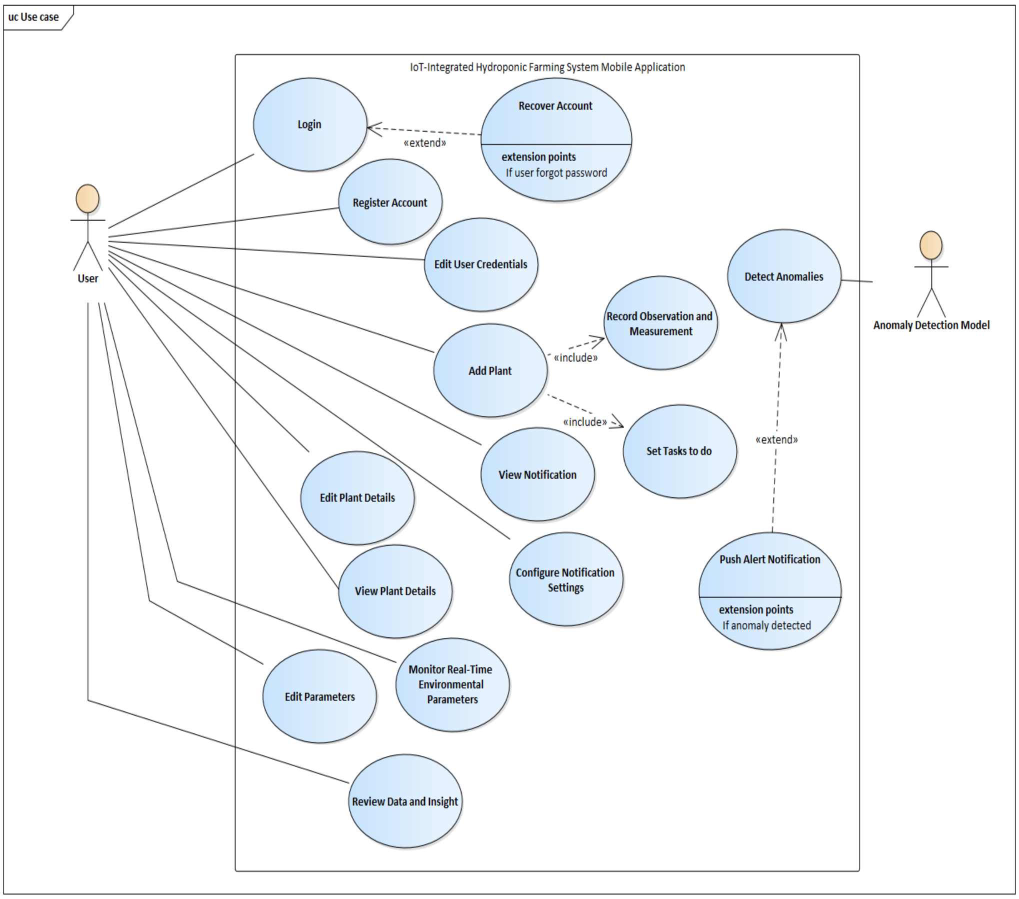

Figure 4.1 Use Case Diagram

4.4 Use Case Description

4.4.1 Login

<table>
<tr><td colspan="2">Use Case Name: Login</td><td>ID: UC01</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Login to account</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a registered user login to the mobile application</td></tr>
<tr><td colspan="4">Trigger: User opens the mobile application</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : Recover Account
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the mobile application.
2. The application displays the login screen.
3. The users forgot their password.
The S-1 Recover Account flow performed.
4. The user enters the valid email and password.
5. The user submits the user credentials.
6. The application verifies the user’s login credentials. If user credentials are invalid, perform Exceptional Flow 6.1.</td></tr>
<tr><td colspan="4">Sub-flows:
S-1 Recover Account
1. The user selects the Recover Account option from the login screen.
2. The application displays the recover screen.
3. The user enters their registered email address and submit.
4. The application generates a password reset link and sends to user’s email address.
5. The application prompts the password reset link sent.
6. Return to main flow step 2.</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
6.1 Invalid User Credentials
1. The application displays an error message on wrong user credentials.
2. The application prompts user to retry the login procedure.</td></tr>
</table>

4.4.2 Register Account

Use Case Name: Register Account	ID: UC02	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Real	
Stakeholders and Interests: User: Register account		
Brief Description: This use case describes that a new user can register a new account.		
Trigger: User opens the mobile application and don't have account.		
Relationships: Association **:** User Include : N/A Extend : N/A Generalization: N/A		
Normal Flow of Events**:** 1. The user selects the Register Account option from the login screen. 2. The application displays the registration screen. 3. The user enters the required information for register account. 4. The user submits the registration information. 5. The application validates the information entered. <u>If the registration information is invalid and duplicated, perform Exceptional Flow 5.1</u>		
Sub-flows**:** N/A		
Alternate/Exceptional Flows: **5.1 Invalid and Duplicated Register Information** 1. The application displays an error message on invalid or duplicated register information. 2. The application prompts user to retry the registration procedure.		

4.4.3 Edit Account Credentials

<table>
<tr><td colspan="2">Use Case Name: Edit User Credentials</td><td>ID: UC04</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Edit user credentials.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a registered user can update their email or password.</td></tr>
<tr><td colspan="4">Trigger: User wants to change user credentials.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens their profile.
2. The application displays option for changing email or password.
3. The user can modify the user credentials by select the option. <u>If the update information is invalid and duplicated, perform Exceptional Flow 3.1</u>
4. The user saves the changes of user credentials.
5. The application prompts update successful message, and updates user credentials.</td></tr>
<tr><td colspan="4">Sub-flows:
N/A</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
5.1 Invalid and Duplicated Update Information
1. The application displays an error message on invalid or duplicated update information.
2. Return to main flow step 2.</td></tr>
</table>

4.4.4 Add Plant

<table>
<tr><td colspan="2">Use Case Name: Add Plant</td><td>ID: UC05</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Add Plant.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a user can add a plant including image, status, observation, and measurement.</td></tr>
<tr><td colspan="4">Trigger: User wants to add a new plant for a sector.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : Record Observation and Measurement, Set Tasks
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the plant screen.
S-1 Add New Observation and Measurement flow performed.
S-2 Add New Task flow performed.
2. The user selects Add Plant option.
3. The user enters the add plant screen and fill in the info.
4. The application saves the new plant. <u>If the information is incomplete or invalid, perform exceptional flow 4.1.</u></td></tr>
<tr><td colspan="4">Sub-flows:
S-1 Add New Observation and Measurement
1. The user enter new observation and measurement to the selected plant.
2. The application save the new info.
3. Return to main flow step 1.
S-2 Add New Task
1. The user enter new task note and date to the selected plant.
2. The application save the new info.
3. Return to main flow step 1.</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
Exceptional Flow 4.1: Incomplete or Invalid information
1. The application displays an error message on incomplete or invalid information.
2. The application prompts user to complete and enter valid information.
3. Return to main flow step 2</td></tr>
</table>

4.4.5 Edit Plant Details

<table>
<tr><td colspan="2">Use Case Name: Edit Plant Details</td><td>ID: UC06</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Edit Plant Details.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a user can edit a plant detail including status, observation and measurement, and tasks.</td></tr>
<tr><td colspan="4">Trigger: User wants to edit a new plant detail.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the plant screen.
2. The application displays list plants for each sector.
3. The user selects a plant that require update.
4. The application displays the selected plant information.
5. The user selects Edit Plant Details option.
6. The user edits the plant details.
7. The application saves the new plant details. <u>If the information is incomplete or invalid, perform exceptional flow 7.1.</u></td></tr>
<tr><td colspan="4">Sub-flows:
N/A</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
Exceptional Flow 7.1: Incomplete or Invalid information
1. The application displays an error message on incomplete or invalid information.
2. The application prompts user to complete and enter valid information.</td></tr>
</table>

4.4.6 View Plant Details

<table>
<tr><td>Use Case Name: View Plant Details</td><td>ID: UC07</td><td colspan="2">Importance Level: High</td></tr>
<tr><td colspan="2">Primary Actor: User</td><td colspan="2">Use Case Type: Detail, Essential</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: View Plant Details.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a user can view a plant detail including image, status, observation and measurement, and tasks.</td></tr>
<tr><td colspan="4">Trigger: User wants to view a new plant detail.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the plant screen.
2. The application displays list of plants for each sector.
3. The user selects a plant to view.
4. The application displays the selected plant information.</td></tr>
<tr><td colspan="4">Sub-flows:
N/A</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
N/A</td></tr>
</table>

4.4.7 Monitor Real-Time Environmental Parameters

<table>
<tr><td colspan="2">Use Case Name: Monitor Real-Time Environmental Parameters</td><td>ID: UC08</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Monitor real-time environmental parameters.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a user can monitor real-time data from sensor in the hydroponic system.</td></tr>
<tr><td colspan="4">Trigger: User wants to monitor hydroponic farm system.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the monitor panel.
2. The application displays the real-time environmental parameters including surrounding temperature, solution temperature, humidity, pH value, nutrient level, light intensity, and TDS level.
3. The application updates the parameters every 10 minutes.
4. The user can manually refresh the monitor panel to retrieve the latest data</td></tr>
<tr><td colspan="4">Sub-flows:
N/A</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
N/A</td></tr>
</table>

4.4.8 Edit Parameters

<table>
<tr><td colspan="2">Use Case Name: Edit Parameters</td><td>ID: UC09</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: User</td><td colspan="3">Use Case Type: Detail, Essential</td></tr>
<tr><td colspan="4">Stakeholders and Interests: User: Edit parameters.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a user can edit parameters remotely</td></tr>
<tr><td colspan="4">Trigger: User wants to adjust parameters for hydroponic farm.</td></tr>
<tr><td colspan="4">Relationships:
Association : User
Include : N/A
Extend : N/A
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The user opens the control screen.
2. The application displays parameters settings for each sector.
3. The user selects edit parameters option.
4. The user can adjust the values of parameters.
5. The user saves the changes on the parameters.
6. The application saves and update the required parameters in database.</td></tr>
<tr><td colspan="4">Sub-flows:
N/A</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
N/A</td></tr>
</table>

4.4.9 View Notification

Use Case Name: View Notification		ID: UC10	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential		
Stakeholders and Interests: User: View notification			
Brief Description: This use case describes that a user can view notification.			
Trigger: User wants to view notification.			
Relationships Association : User Include : N/A Extend : N/A Generalization: N/A			
Normal Flow of Events**:** 1. The user opens the notification screen. 2. The application displays list of notification. 3. The user can delete the notification. 4. The user can select one of the notifications for review. 5. The application displays the details of the notifications.			
Sub-flows: N/A			
Alternate/Exceptional Flows: N/A			

4.4.10 Configure Notifications Settings

Use Case Name: Configure Notification Settings	ID: UC11	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Real	
Stakeholders and Interests: User: Configure Notification Settings		
Brief Description: This use case describes that a user can configure notification settings.		
Trigger: User wants to configure notification settings.		
Relationships Association : User Include : N/A Extend : N/A Generalization: N/A		
Normal Flow of Events: 1. The user opens the user profile. 2. The user selects Notification Settings option. 3. The application displays the current enabled notification settings. 4. The user can enable or disable the push of reminders and alerts. 5. The application saves the notification settings.		
Sub-flows: N/A		
Alternate/Exceptional Flows: N/A		

4.4.11 Review Data and Insight

Use Case Name: Review Data and Insight	ID: UC12	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Real	
Stakeholders and Interests: User: Review data and insight		
Brief Description: This use case describes that a user can review data and insight		
Trigger: User wants to gain insights of hydroponic farm.		
Relationships Association : User Include : N/A Extend : N/A Generalization: N/A		
Normal Flow of Events: 1. The user opens the Insight screen. 2. The application displays the data and insight of the current farm. 3. The insight categorizes data and insight to two periods, daily and monthly. 4. The user can select the date for review. 5. The user can choose to export the specific date of parameter data to csv file.		
Sub-flows: N/A		
Alternate/Exceptional Flows: N/A		

4.4.12 Detect Anomalies

<table>
<tr><td colspan="2">Use Case Name: Detect Anomalies</td><td>ID: UC13</td><td>Importance Level: High</td></tr>
<tr><td>Primary Actor: Anomaly Detection Model</td><td colspan="3">Use Case Type: Detail, Essential</td></tr>
<tr><td colspan="4">Stakeholders and Interests: Anomaly Detection Model: Detect anomalies</td></tr>
<tr><td colspan="4">Brief Description: This use case describes that a model able to identify abnormal patterns or deviations from expected forms.</td></tr>
<tr><td colspan="4">Trigger: When there are anomalies in the data collected</td></tr>
<tr><td colspan="4">Relationships
Association : Anomaly Detection Model
Include : N/A
Extend : Push Alert Notification
Generalization: N/A</td></tr>
<tr><td colspan="4">Normal Flow of Events:
1. The model retrieves the real-time sensor data.
2. The model analyses the data based on trained data.
3. The model identifies abnormal patterns or deviations from the expected forms.
4. The application collected details of the anomaly detected.
5. The application can push alert notification to user.
S-1 Push Alert Notification flow performed.</td></tr>
<tr><td colspan="4">Sub-flows:
S-1 Push Alert Notification
1. The application check the alert notification settings is enable
2. The application push the alert notification.
3. Return to main flow step 1.</td></tr>
<tr><td colspan="4">Alternate/Exceptional Flows:
N/A</td></tr>
</table>

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter provides an in-depth discussion of the system architecture design, database design, and user interface design. The system architecture design outlines the flow of the system process, including the frameworks and connections used. The database design visualizes the structure of the database and presents the data dictionaries. Finally, the user interface design introduces each screen's design and its respective functionality.

5.2 System Architecture Design

The mobile application implements a client-server architecture that is divided into three tiers: presentation tier, application tier, and database tier, representing the frontend, backend, and database layers, respectively. Figure 5.1 visualizes the system architecture of the entire system. This architecture ensures that each layer is independent, thereby enhancing the maintainability and flexibility of the system during implementation and development. Additionally, it improves the system's reliability by implementing rules that limit user access to data based on authority levels.

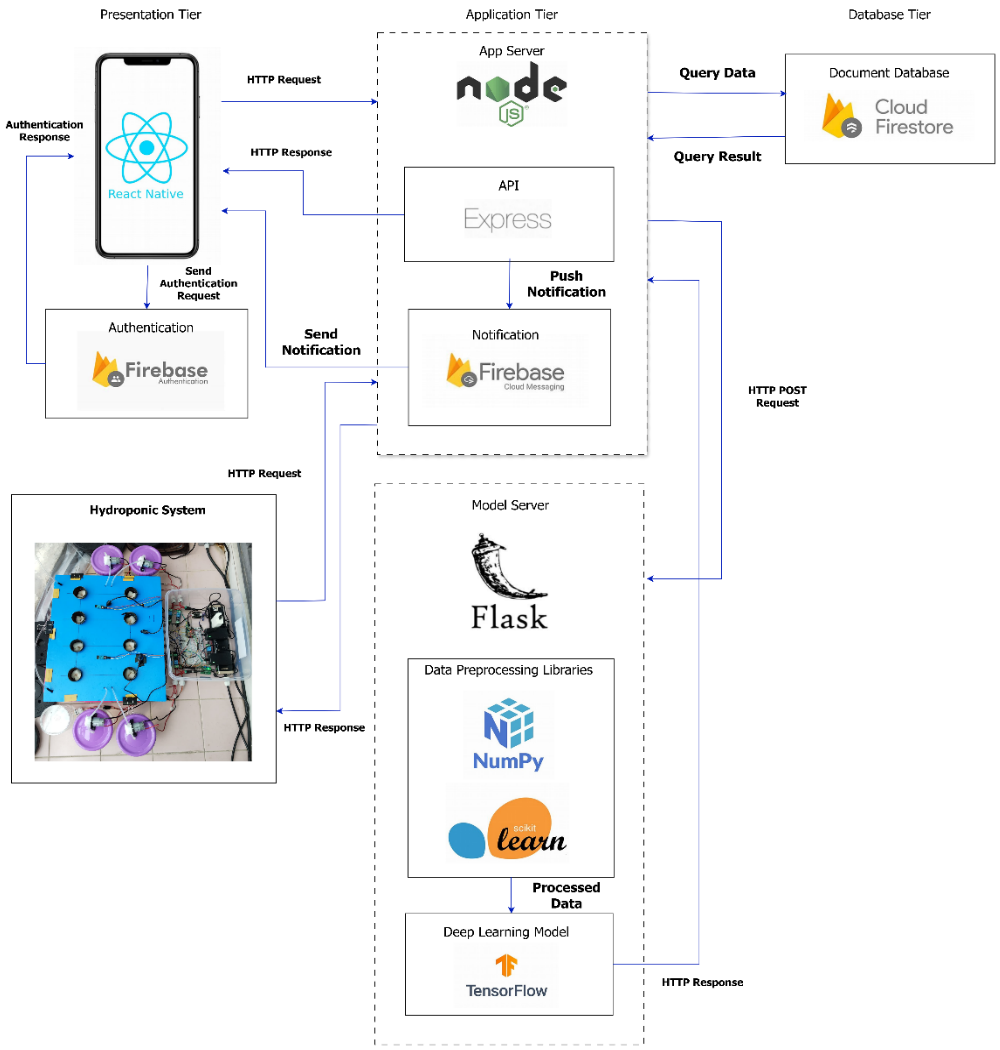

Figure 5.1 System Architecture Design

The presentation tier includes the mobile application, developed using the React Native framework to simplify the development process. The mobile application is responsible for displaying processed data to users and receiving user input. Additionally, the hydroponic system functions as a client within the overall system. To enable communication with the mobile application and data upload, the hydroponic system sends HTTP requests to the server for data uploading and other actions. Furthermore, Firebase Authentication is employed in the presentation tier to handle user registration and sign-in processes, offering comprehensive authentication services, including email verification and password reset, thus reducing the development time for the user authentication module.

The application tier, acting as the middle layer, is responsible for executing business logic, managing data access and processing, and performing tasks such as triggering push notifications when certain conditions are met, as well as scheduling notifications or reminders based on user settings. Node.js was selected to provide the foundation for deploying functionalities and services in the application tier due to its ease of deployment and flexibility. The app server continuously operates to handle HTTP requests directed at the application programming interface (API), which is customized using the Express.js framework. The API processes requests, executes actions such as communicating with the database to retrieve data, and finally returns data via HTTP response. For the model server, the Flask framework was chosen to set up the server for the anomaly detection machine learning model, as the model relies on Python libraries such as NumPy, Pandas, and TensorFlow. To obtain the anomaly detection results, the app server sends HTTP POST requests to the model server with data retrieved from the database. This data is preprocessed before being sent to the model for prediction and anomaly detection, with the results returned via HTTP response.

Moreover, Firebase Cloud Messaging is used to provide notification services for the mobile application due to its efficient message delivery and customization capabilities. The backend server pushes notifications when certain conditions are met, such as user settings or scheduled reminders and tasks. Additionally, the trained anomaly detection model can send request to app-server to push notifications when an anomaly is detected. To ensure real-time monitoring, the trained model constantly retrieves the latest data from the database and identifies any potential anomalies.

Lastly, the database tier is responsible for data storage and retrieval. This tier includes the database management system (DBMS) that handles data storage, retrieval, and modification. Firebase Cloud Firestore and Cloud Storage are used to store all the data utilized by the system, including user data, farm data, and images. These technologies were chosen because they offer NoSQL database and cloud storage solutions, enabling flexible and scalable data storage for the entire system.

5.3 Database Design

Before building the actual database, a database design process was conducted, which involved defining data elements, data relationships, and normalizing data. This section visualizes the database design using an Entity Relationship Diagram (ERD) and a Data Dictionary, providing an overview of the database structure for the mobile application.

5.3.1 Entity Relationship Diagram

An Entity Relationship Diagram (ERD) is a crucial tool for database design and development, as it visualizes the overall structure of the database and validates the design against the system's requirements. Figure 5.2 illustrates the ERD for the IoT-Integrated System for Monitoring Hydroponic Farming. This ERD contains four entities: Users, Farms, Sectors, Plants, Devices, Anomalies.

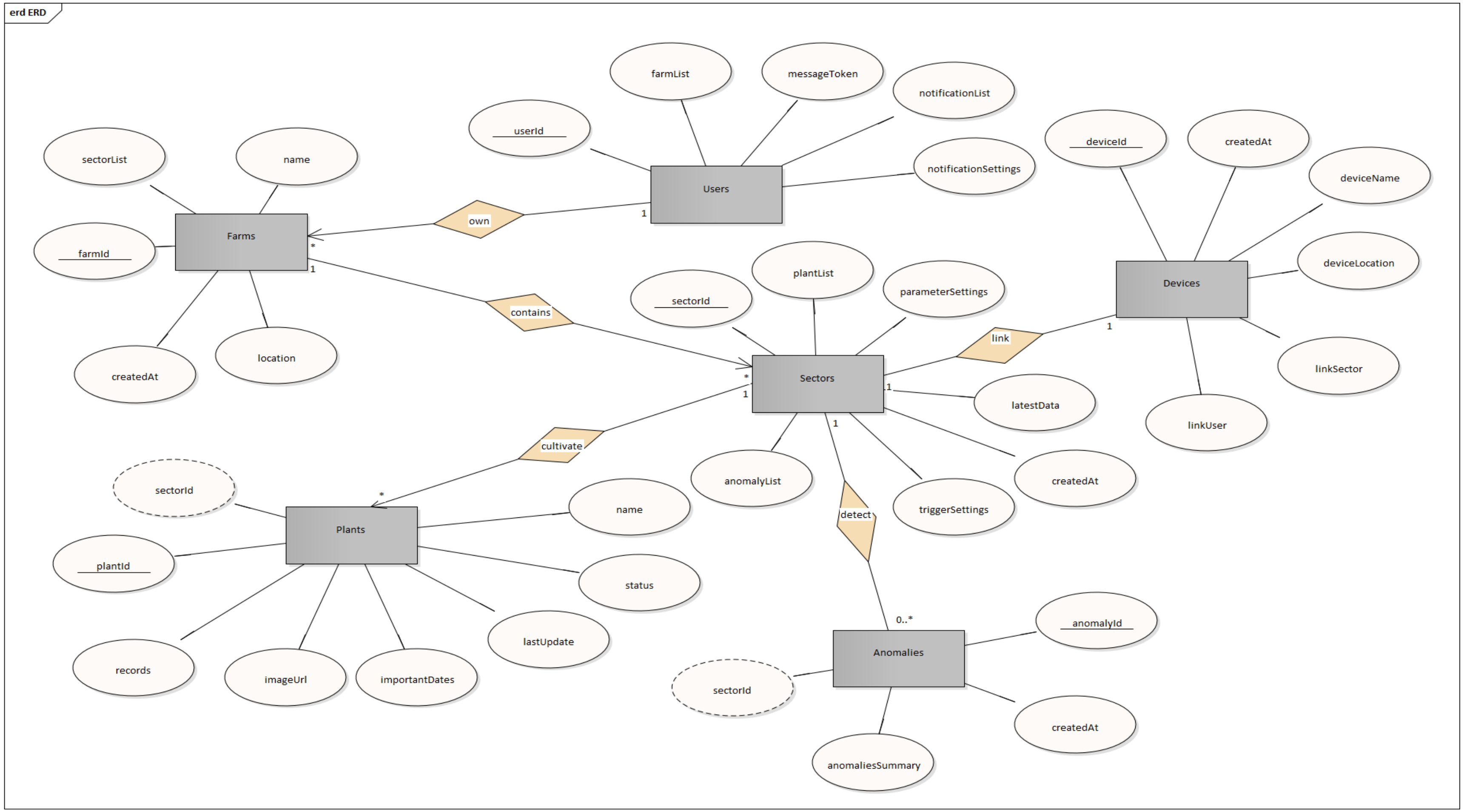

Figure 5.2 Entity Relationship Diagram

5.3.2 Data Dictionary

Table 5.1 Description of Database Tables

Table Name	Description
Users	Contains user's data required for performed certain action
Farms	Contains farm information, one user can have multiple farms.
Sectors	Contains sector information, one farm can have multiple sectors.
Plants	Contains plant information, one sector can have multiple plants.
Devices	Contains device information, one device only can have one sector.
Anomalies	Contains anomaly information, one sector can have zero to many anomalies.

Table 5.2 Users Entity Data Dictionary

Fields	Field's Description	Type	Example
userId [PK]	Unique identifier for the user.	String	VWYk4xnl1UZTx6R4HgMWdP5To4j1
farmList	Array of references (document IDs) to documents within the	String	aJ59zZKqjt8zm0JC2Hq3

	Farms collection, indicating farms the user is associated with.		
messageToken	Token used for sending push notifications to the user's device.	String	cZ30dQrFRm6q3pXktOkjsv:APA91bH1gZ4OjWj7
notificationList	Array of objects containing notification details.	String	This is your daily reminder for you to check on your farm. 2024-07-10 1720586167113 12:36:06 Daily Reminder normal
notificationSettings	Array of user preferences for different notification categories.	Boolean	true true true

Table 5.3 Farms Entity Data Dictionary

Fields	Field's Description	Type	Example
farmId [PK]	Unique identifier for the farm.	String	aJ59zZKqjt8zm0JC2Hq3
createdAt	Date and time the farm was created.	Timestamp	2024-07-11T13:44:08.497Z
location	Location details of the farm.	String	test location
name	Name of the farm.	String	test
sectorList	Array of references (document IDs) to documents within the Sectors collection, representing sectors belonging to the farm.	String	DzTAF5V1yxGSdVT9UfAO

Table 5.4 Sectors Entity Data Dictionary

Fields	Field's Description	Type	Example
sectorId [PK]	Unique identifier for the sector.	String	DzTAF5V1yxGSdVT9UfAO
createdAt	Date and time the sector was created.	Timestamp	2024-07-11T13:44:08.497Z
latestData	Object containing the latest value of the multiple parameter objects	String	surroundingTemperature -timestamp: 2024-07-29T21:06:20 -value: 31.3

plantList	Array of references (document IDs) to documents within the Plants collection, representing plant belonging to the sector.	String	zntrE8qKcmEY8UC2kHKV
parameterSettings	Object containing the upper and lower ranges of the multiple parameter objects	Number	surroundingTemperature 20 40
anomalyList	Array of anomaly document IDs to documents within the anomaly collection, referring the anomaly detected in the sector	String	2UMLVlf2LlmEMOCqZHii
triggerSettings	Object containing the multiple IoT device trigger name and values	Boolean	foggerTrigger: false

Table 5.5 Plants Entity Data Dictionary

Fields	Field's Description	Type	Example
plantId [PK]	Unique identifier for the plant.	String	OyWjQu67vnTzH4fQ1waO
imageUrl	Reference (URL) to the plant's image stored in Firebase Storage.	String	https://storage.googleapis.com/test-aeba2.appspot.com/sectors/

importantDates	Array of objects containing dates and notes for the plant.	String	2024-07-07 Leaf Check
lastUpdate	Date and time the plant document was last updated.	Timestamp	2024-07-11T13:44:08.497Z
name	Name of the plant.	String	lettuce
records	Array of objects containing dates, observation, and measurements.	String	2024-07-11T13:44:08.497Z Leaf in healthy state 3cm
sectorId	Reference (document ID) to a document within the Sectors collection.	String	DzTAF5V1yxGSdVT9UfAO
status	Current state of the plant	String	healthy

Table 5.6 Devices Entity Data Dictionary

Fields	Field's Description	Type	Example
deviceId [PK]	Unique identifier for the device.	String	YVsE3C3e4pwfLs8Rh7PM
createdAt	Date and time the device was created.	String	2024-07-28T00:54:07
deviceName	Name of the device	String	abc

deviceLocation	Location of the device	String	abc location
linkSector	The linked sector ID	String	rXHbTROjARlvr2DCubny
linkUser	The linked user ID	String	jNpmgDej52T9D758EoZyS0HF4Yl2

Table 5.7 Anomalies Entity Data Dictionary

Fields	Field's Description	Type	Example
anomalyId [PK]	Unique identifier for the anomaly.	String	2UMLVlf2LlmEMOCqZHii
createdAt	Date and time the anomaly was detected.	String	2024-07-28T00:54:07
anomalySummary	Object containing detected status and anomaly score	String	Detected: true Anomaly_score: 10.01
sectorId	The anomaly detected at this sector	String	rXHbTROjARlvr2DCubny

5.4 User Interface Design

User interface (UI) design visually represents the mobile application's features and layout, ensuring that user requirements and expectations are met.

5.4.1 User Authentication Pages

The user authentication module comprises three screens: the login screen, the registration screen, and the forgot password screen. Figure 5.3 displays the UI design of the user authentication module. Users are required to enter a valid email address and password to sign in to the application. During registration, users must correctly input their email and password. The system verifies whether the email has already been registered. After verification, a confirmation email is sent to the user's email address. If a user forgets their password, they can submit their registered email address, and a password reset link will be sent to that address.

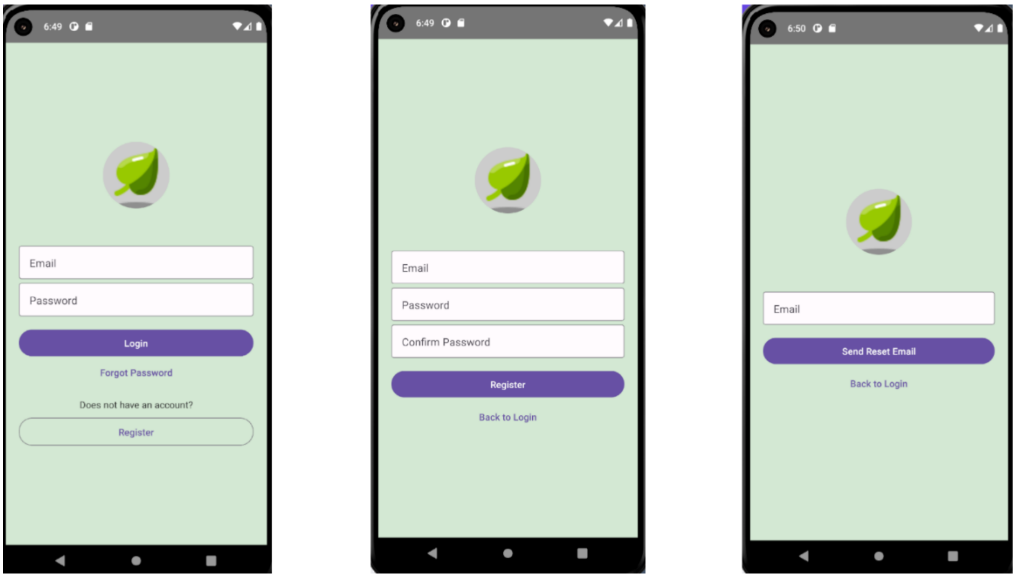

Figure 5.3 Login, Register and Forgot Password Screen

5.4.2 Monitor Panel Pages

Figure 5.4 presents the monitor panel screen of the application. An icon in the upper right corner allows users to navigate to the user profile screen. The monitor panel displays the current farm name along with related sector parameters and events. Users can switch between parameters and tasks by clicking on the upper tab bar, with parameters grouped by sector. On the right side of the screen title, icons provide features such as editing farms and sectors. The bottom navigation bar includes five modules: Monitor Panel, Control Panel, Data and Insight, Plant, and Notification.

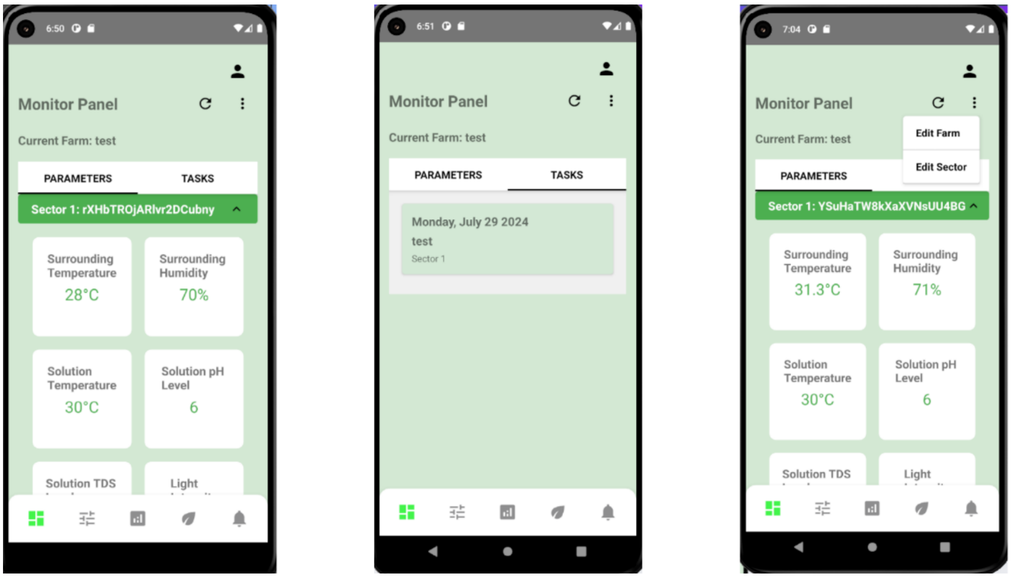

Figure 5.4 Monitor Panel Screen

Figure 5.5 illustrates the edit farms screen with its functions. Users can add or delete farms using the plus and edit icons located in the upper right corner of the screen. To add a farm, users must enter the farm name and location.

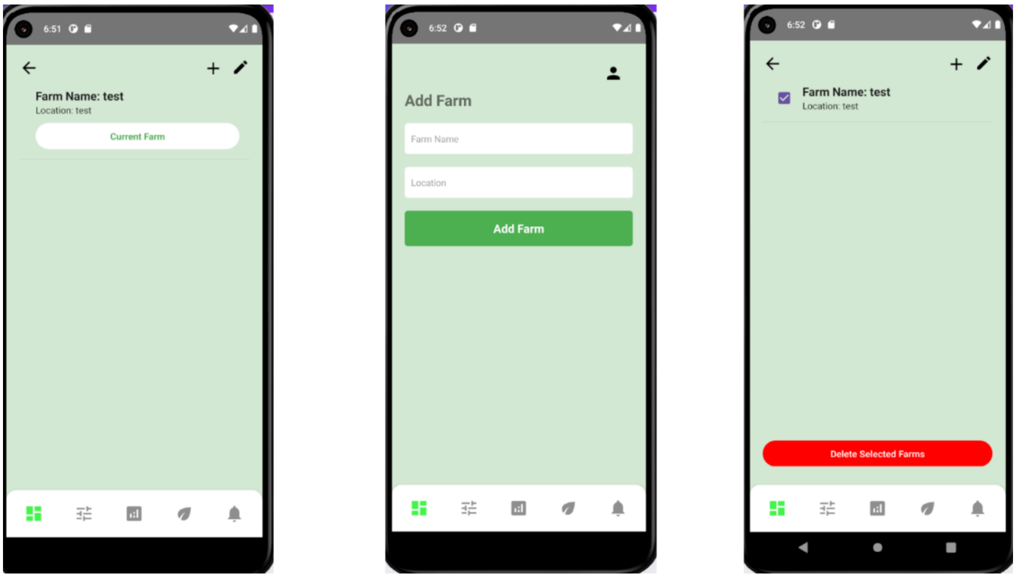

Figure 5.5 Edit Farm Screen

Figure 5.6 depicts the edit sector screen with its functions. Similar to farm management, users can add or delete sectors using the icons in the upper right corner. To add a sector, users need to enter the device information obtained after registering the device on the hydroponic farm system.

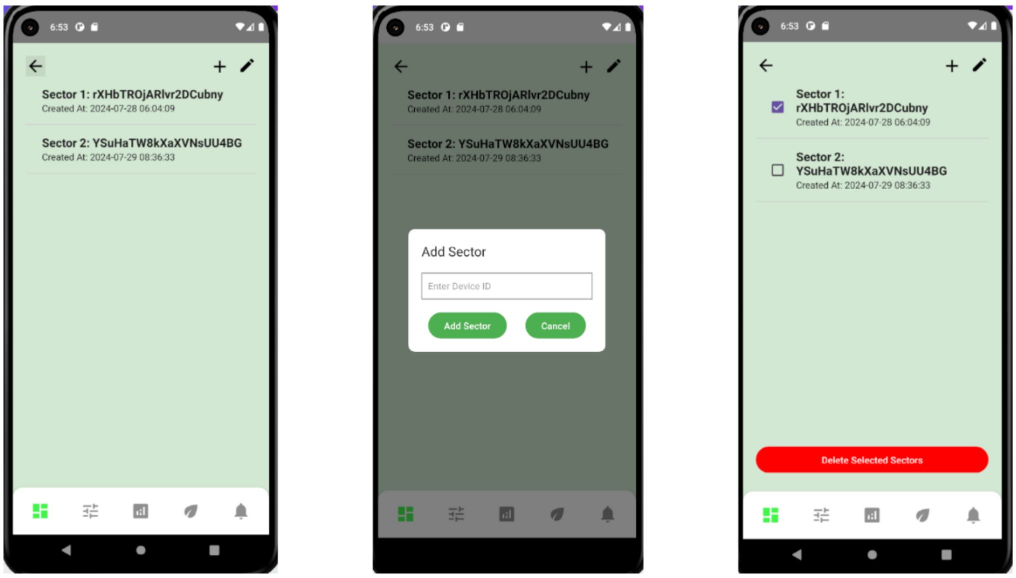

Figure 5.6 Edit Sector Screen

5.4.3 Control Panel Screen

Figure 5.7 shows the control panel screen of the application. This screen displays all the sector parameter settings associated with a farm. By clicking the edit icon next to each parameter, a modal window opens, allowing users to edit the parameter settings. Below the parameter settings, users can manually turn IoT devices, such as pumps for pH and TDS control, on or off using the trigger actions.

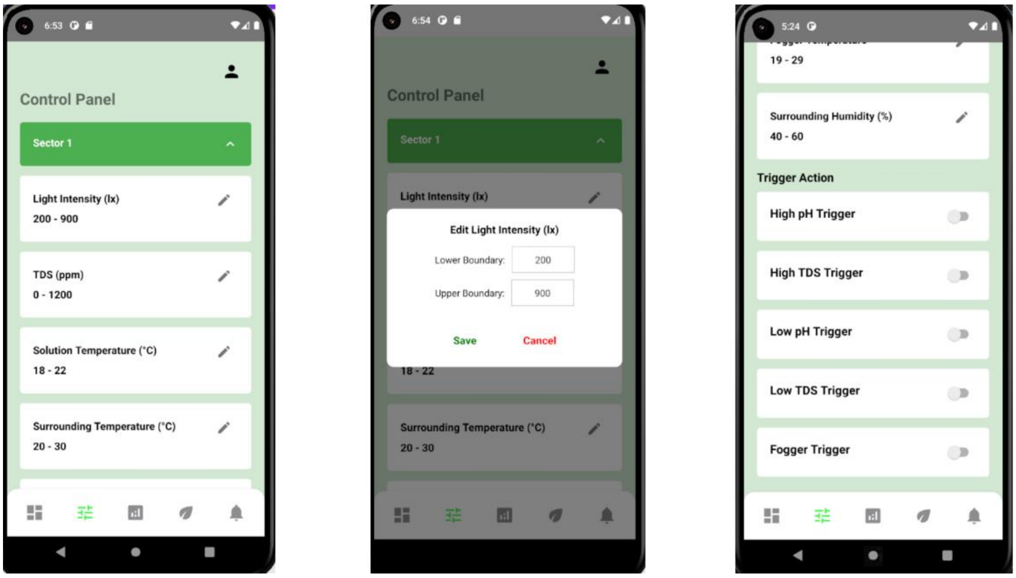

Figure 5.7 Control Panel Screen

5.4.4 Data and Insight Screen

Figure 5.8 presents the data and insight screen of the application. It displays parameter trends and detected anomalies based on the selected sector and timeframe, including daily and monthly data. Users can switch between parameters to update the trend graph accordingly. The trend graph displays actual data points, predicted data points, and anomaly points. Detected anomalies are detailed with the detection time, threshold, and the discrepancy between predicted and actual values, allowing users to analyse abnormal events thoroughly. Additionally, users can export all parameter data for a specific date by clicking the download button.

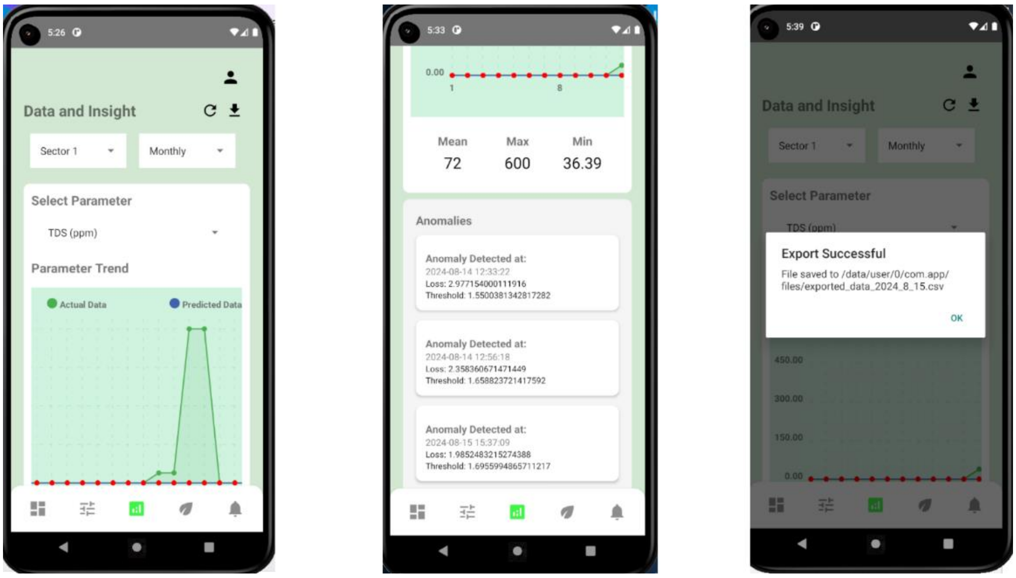

Figure 5.8 Data and Insight Screen

5.4.5 Plant Management Pages

Figure 5.9 shows the Plant Screen, Add Plant Screen, and Plant Detail Screen. The Plant Screen lists all plants within the sectors that have been added. Users can navigate to the Add Plant Screen by clicking the plus icon at the bottom right of the screen. To view plant details, users simply click on the respective plant item. The Plant Detail Screen displays the plant's status, observation records, and tasks to be performed. Users can add new records and tasks by clicking the respective buttons. The edit and delete icons in the upper right corner allow users to modify or remove plant details.

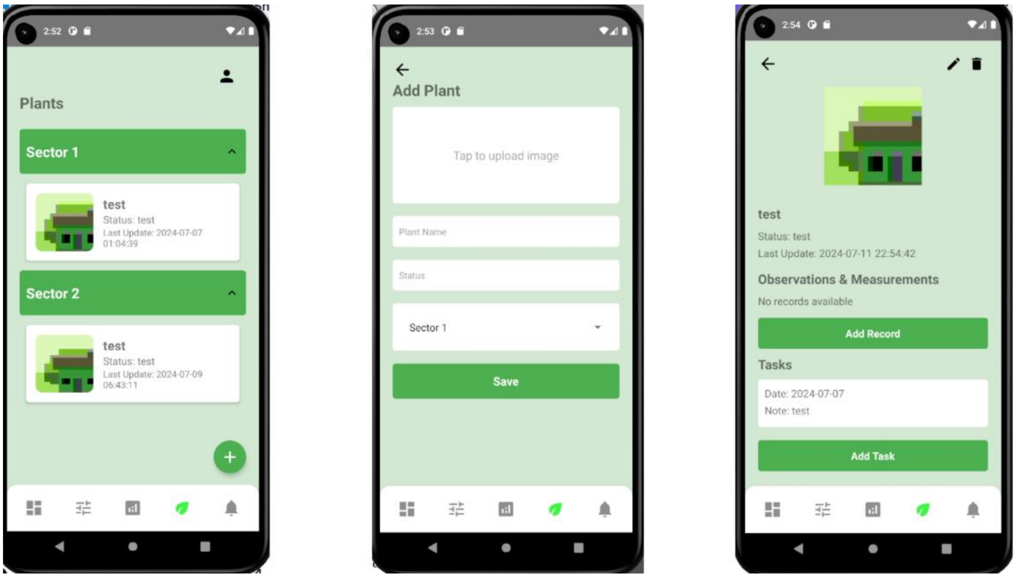

Figure 5.9 Plant Screen, Add Plant Screen, and Plant Detail Screen

Figure 5.10 shows the modals for adding observation records and tasks, as well as the edit plant detail screen, which appears after clicking the edit icon. Tasks can only be deleted, while observation records can be both edited and deleted. Users can also edit the plant's image and status. After editing, users must click the save button in the upper right corner to save changes; otherwise, modifications will not be saved.

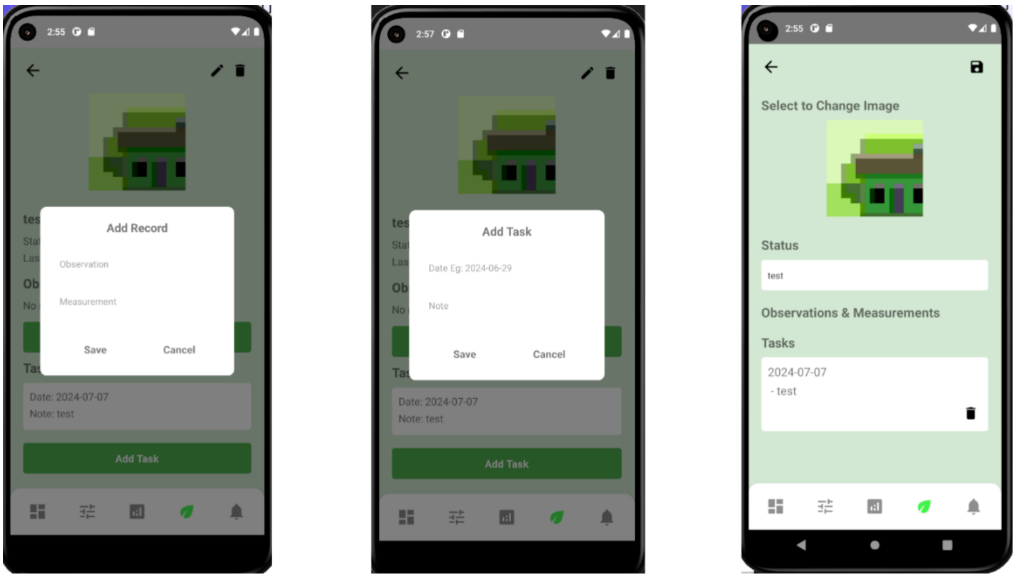

Figure 5.10 Add observations and tasks, and Edit Plant Detail Screen

5.4.6 Notification Screen

Figure 5.11 shows the notification screen of the application. This screen lists all notifications sent to the user. Users can delete notifications by clicking the top-right edit icon and selecting the notifications they wish to remove. A modal will appear showing the notification details when a user clicks on a notification.

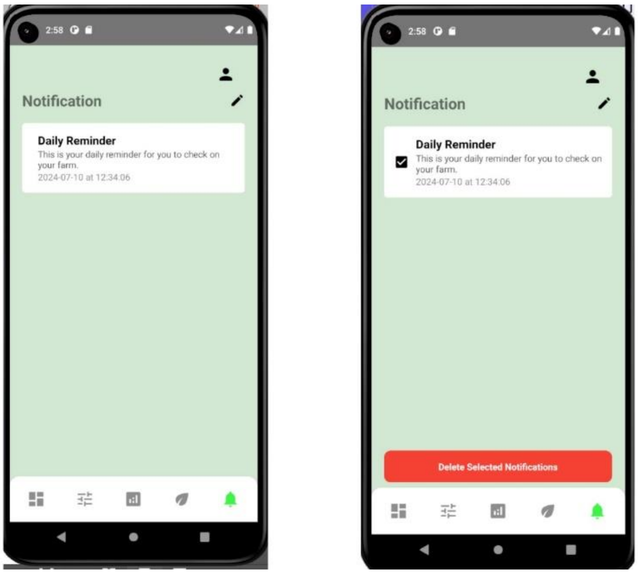

Figure 5.11 Notification Screen

5.4.7 User Profile Pages

Figure 5.12 shows the user profile page of the application, which lists four options: change email, change password, notification settings, and logout.

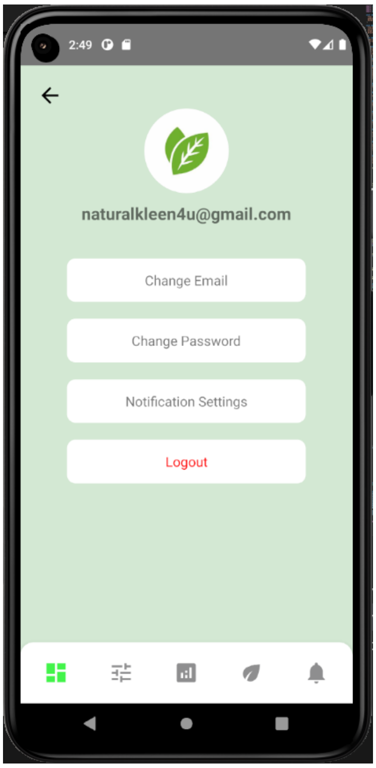

Figure 5.12 User Profile Screen

Figure 5.13 displays the screens for each of these options: change email, change password, and notification settings. To change their email, users must enter the new email address. To change their password, users must enter both their old and new passwords to ensure account security. Both actions will log the user out, requiring them to sign in again with their new credentials. The notification settings screen lists options for push notifications, including task reminders, daily reminders, and anomaly alerts. Users must save their updated settings before returning to the user profile.

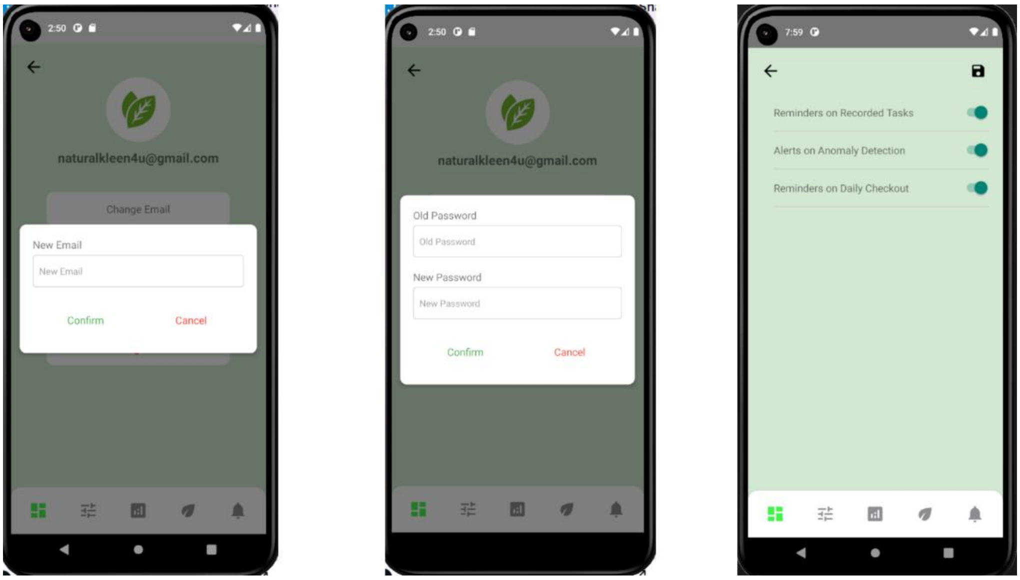

Figure 5.13 User Screen with Edit User Credentials Function

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This chapter discusses the implementation of the frontend, backend, and model training components of the Hydroponic Farm Monitoring Application. The frontend implementation covers the development of user interfaces and the setup of Firebase Authentication. The backend implementation focuses on configuring the application programming interface (API) and deploying the machine learning model. The model training section outlines the steps involved in training the machine learning model.

6.2 Frontend Implementation

The project utilizes the React Native framework as the foundation for mobile application development. The initial setup involves initiating the React Native framework using the command npx react-native init HydroponicApp, which downloads the necessary libraries, packages, and configuration files, including default directories and environment settings.

6.2.1 Firebase Authentication

Firebase Authentication is integrated into the project to manage user authentication and account management. The setup involves registering the application with Firebase and downloading the necessary configuration file, google-services.json. This file includes critical credentials such as the API key, project ID, and application ID, which are required for communication between the application and Firebase. Once the Firebase project is connected, the Firebase Authentication module is integrated into the application, including the setup of email/password sign-in methods as shown in Figure 6.1.

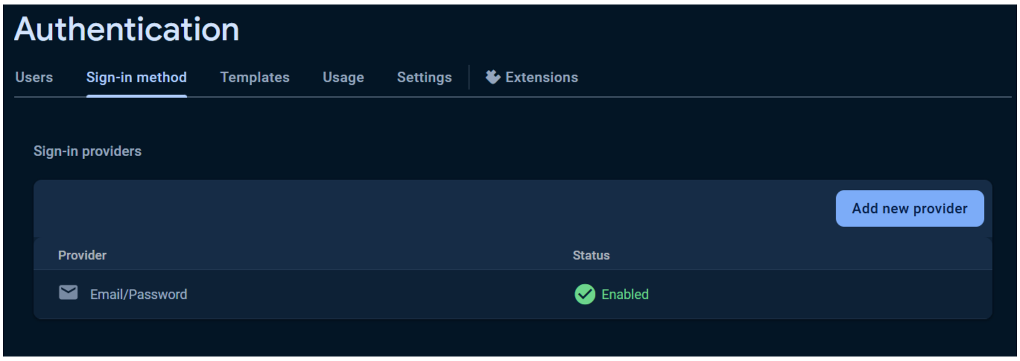

Figure 6.1 Firebase Authentication Providers

The application implements several authentication features. Users can register for a new account by completing a registration form, which captures the user's email and password. After validation, a new user account is created via Firebase Authentication, and a user document is generated to store additional information such as farm lists and notification settings. The login functionality allows users to sign in using their registered email and password. In case users forget their password, the application provides a password reset option, sending a reset link to the registered email. The user interface (UI) for login, registration, and password reset is built using react-native-paper, as referenced in Figure 5.3 of Chapter 5. Forms are managed using Formik, with schema validation provided by Yup to ensure accurate and complete data input.

6.2.2 Monitor Panel

The application is designed to monitor real-time data from sensors in the hydroponic system. The react-native-tab-view is used to display tabs for parameters and tasks, each associated with different sectors, as shown in Figure 5.4. Users can easily switch between sectors and view associated data by selecting the relevant tab. Data retrieval from the server is handled via API calls, and the retrieved sector info is stored locally using AsyncStorage to enhance performance and reduce unnecessary API calls.

The monitor panel also includes pages for managing farm and sector lists. Users can view, delete, and add farms or sectors using an intuitive UI built with react-native-paper. Adding a farm requires inputting the farm name and location, while adding a sector involves registering the device ID associated with the hydroponic system. Each device can be linked to only one sector, ensuring data synchronization with sensor readings.

6.2.3 User Profile

The application allows users to update account credentials and configure notification settings, as shown in Figure 5.12. Users can change their email address or password through the user profile. When updating the email address, the application sends an API request to the server with the new email. After the update, the user is signed out and must verify the new email address before signing in again. Similarly, password changes require re-authentication using the old password. The EmailAuthProvider from react-native-firebase/auth is used to validate the old password before applying the new one. Users can also modify their notification preferences, which are saved to the database via API calls.

6.2.4 Control Panel

The control panel allows users to adjust parameter ranges for the hydroponic system remotely. Settings such as temperature and humidity ranges can be updated through the application, with data stored in the database and retrieved via API. Users can also manually trigger IoT devices, such as pumps, to adjust pH and TDS levels. The hydroponic system retrieves these trigger settings from the server and updates its status, accordingly, providing users with flexibility in managing their systems.

6.2.5 Data and Insight

The application includes a Data and Insight screen for reviewing historical data, anomaly data, and statistical insights. Users can select parameters and time intervals (daily and monthly) to view trends in line charts, with data displayed in different colors for easy differentiation. The react-native-pickers and react-native-chart-kit libraries are used to create an interactive data display. Anomalies are highlighted in red on the chart, and detailed information is provided below the chart. Users can also export data by clicking the download button.

6.2.6 Plant Management

The application supports plant management, allowing users to add, edit, and track plant details. Users can upload images and record observations using the react-native-image-picker library. Plant data is managed through an API, with updates reflected in the plant list. Event listeners are used to ensure real-time updates without memory leaks.

Handling date displays in Firebase Firestore required converting timestamps to JavaScript Date objects. The format function from the date-fns library is used to format dates, while convertToMYT adjusts for time zones.

6.2.7 Notification

The application integrates Firebase Cloud Messaging (FCM) to handle notifications, ensuring users receive timely updates. The application checks for remote messaging registration on startup and manages the messaging token using setMessagingToken. Notifications are handled based on the app's state, with different handling for background and foreground notifications. Notifications are saved in the database and displayed in the notification screen, where users can view details or delete them. The app server manages notification sending, as detailed in Section 6.3.1.

6.3 Backend Implementation

The backend is implemented using Node.js and Express.js for the application server, and Flask for the model server. Initial setup involves configuring the necessary dependencies and environments for both Node.js and Flask.

6.3.1 Application Server

The application server integrates Firebase for database operations, storage, and Firebase Admin SDK. Firebase Firestore handles data management, while Firebase Storage stores files, and Firebase Admin handles administration action such as email changing and notification sending. The server uses Express.js for routing and node-cron for scheduling tasks. A daily cron job is configured to send notifications based on user settings and plant tasks.

To enable anomaly detection, the server requires a minimum of 100 data points before executing predictions. Data is sent to the model server for analysis, and the results are used to update the application with anomaly alerts and prediction data.

6.3.2 Model Server

The model server processes data received from the application server using a saved machine learning model for anomaly detection. The server uses libraries such as NumPy, scikit-learn, and TensorFlow for data processing and model inference. This server consists of two API endpoint to perform functionality using the saved model’s architecture definition which are the anomaly detection and trigger action prediction. Anomaly detection performed by comparing adaptive threshold with the losses calculated during the prediction process. The server returns anomaly detection results and predicted values to the application server, which then updates the user interface. While the trigger action prediction will check the current trigger status and compare with the predicted trigger status then return the trigger status back to the application server.

6.3.3 API Functions

The server provides multiple API services, including routes for managing user data, farms, sectors, plants, and notifications. The server also handles multipart form data using multer and processes image uploads for plant management.

Table 6.1 API List of Multiple Backend Services

Service: Plant		**Default service endpoint: /plant**	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/getplants	GET	{sectorId}	Retrieve list of plants associated with the sector
/addplant/{sectorId}	POST	{name, status}, file	Add a new plant to a sector
/plant/{plantId}	GET	-	Fetch and return detail of a specific plant
/deleteplant	DELETE	{plantId, sectorId}	Delete a plant from database
/updateplant/{plantId}	PUT	{status, imageUrl, records, importantDates}	Update detail of existing plant
/plant/{plantId}/records	POST	{observation, measurement}	Add a new record (observation and measurement)
/plant/{plantId}/importantDates	POST	{date, note}	Add a task with a date to the specific plant

Service: User		Default service endpoint: /user	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/register	POST	{userId, farmList, messageToken}	Register a new user
/updateEmail	POST	{uid, newEmail}	Update user email address
/getUser	GET	{userId}	Retreive and return user data from database
/updateUserSettings	POST	{userId, notificationSettings}	Update user’s notification settings
/checkToken	POST	{userId, messageToken}	Check messageToken is changed or not
/checkEmail	POST	{email}	Check user exists or not, based on email
Service: Farm		**Default service endpoint: /farm**	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/getfarm/{userId}	GET	-	Retreive and return list of farms associated with the user
/addfarm	POST	{userId, name, location, createdAt}	Add a new farm to database
/deletefarm/{farmId}	DELETE	{userId}	Delete the specific farm from database

Service: Sector		Default service endpoint: /sector	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/getLatestData/{sectorId}	GET	-	Retrieve and return latest data for a specific sector
/getSector/{farmId}	GET	-	Retrieve and return list of sectors for a specific farm
/addSector	POST	{farmId, deviceId, userId}	Add a new sector to specific farm
/updateData	POST	{userId, sectorId, parameters}	Update parameter data for specific sector
/updateParameterSettings	POST	{sectorId, parameterSettings}	Update the parameter settings for the sector
/getStatus/{sectorId}	GET	-	Fetch and return status of specific sector
/getParameterSettings/{sectorId}	GET	-	Retrieve and return parameter settings of specific sector
/getParameterData	POST	{sectorId, selectedInterval}	Fetch and return parameter data for specific sector based on selected interval
/getAnomaliesData	POST	{sectorId, selectedInterval}	Retrieve and return anomaly data for specific sector
/deleteSector	DELETE	{farmId, sectorId}	Delete a specific sector from given farm

/getTriggerSettings/{sectorId}	GET	-	Retrieves the trigger settings for a specific sector
/updateTriggerSettings	POST	{sectorId, triggerSettings}	Updates the trigger settings for a specific sector
/triggerResult/{userId}	POST	{triggerType, status, detail}	Records the result of a trigger execution and send a notification
/getDataForExport	POST	{sectorId, year, month, day}	Fetches and returns the specific sector parameter data for a specific day from the Firestore database.
Service: Message		**Default service endpoint: /message**	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/getNotification	GET	{userId}	Retrieve and return list of notifications for specific user
/deleteNotification	DELETE	{userId, notificationIds}	Delete specific notification based on array of notification ID
/checkAndSaveNotifications	POST	{userId}	Check condition and save new notification for specific user

/sendAlert	POST	{userId, sectorId}	Check alert condition for specific sector and user, and save related notification
Service: Device		**Default service endpoint: /device**	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/register	POST	{deviceName, deviceLocation}	Register a new device
/getDevice/{deviceId}	GET	-	Retrieve and return information for specific device
Service: Model (model-server)		**Default service endpoint: /**	
Endpoint	**HTTP method**	**Body / Query**	**Description**
/receive-data	POST	{latestData}	Processes and analyses incoming sensor data for anomalies, returning anomaly summaries and predictions with actual values.
/predict-trigger	POST	{latestData}	Predicts and checks trigger conditions based on real-time data and predefined thresholds.

6.4 Model Training

In this section, we detail the training process of the Long Short-Term Memory (LSTM) model, implemented in Google Collab, for identifying abnormal patterns or deviations within the data collected from the hydroponic farming system. The model is trained on historical sensor data collected from a hydroponic system, with preprocessing steps including data cleaning and scaling.

6.4.1 Data Preparation

Data preparation is a crucial step in effectively training a machine learning model, particularly for time series anomaly detection. This project employed the Pandas, NumPy, and Scikit-learn libraries to manage data manipulation, cleaning, numerical operations, array handling, and feature scaling.

Initially, the dataset was loaded into a Pandas DataFrame (df) from a Comma Separated Value (CSV) file using the read_csv() function, ensuring the parse_dates parameter was set to interpret the "Time" column as a datetime object. Data cleaning was then performed using the dropna() function to remove rows with missing (NaN) values. Non-numeric columns were converted to numeric types, with errors coerced to NaN, except for the "Time" column. The "Time" column was subsequently set as the index of the DataFrame, and specific columns required for analysis were selected, as listed in Table 3.6.

To prepare the data for the LSTM model, the features were standardized using StandardScaler, which normalizes the data. This step ensures that all features are on the same scale, improving the model's performance. A sequence of length 10 was then created from the scaled data, allowing the model to learn patterns over multiple time steps. The create_sequences() function generated overlapping sequences, converting them into an array with the shape (num_sequences, seq_length, num_features).

6.4.2 Model Definition

The LSTM model was defined and trained using the build_and_train_lstm() function, which leverages the TensorFlow/Keras library. This library was chosen for its high-level API that simplifies the process of defining neural networks, training models, and evaluating performance. Additionally, NumPy was used for numerical operations and array handling, particularly for sequence creation and data splitting.

The data was split into training and validation sets, with 80% of the sequences allocated for training and the remaining 20% for validation. The training and validation datasets (X_train, y_train, X_val, y_val) were prepared by splitting each sequence into features and target values. The model was constructed using the Sequential API, which is ideal for models where each layer has a single input and output tensor.

The model architecture includes two LSTM layers, as detailed in Table 6.2. The Rectified Linear Unit (ReLU) activation function was employed in both LSTM layers. The first LSTM layer returned the full sequence of outputs, while the second returned only the output of the last time step. The input shape was defined for the first LSTM layer which include the timesteps (10 data point) with input_dim, the 13 features of the data then inferred for the second. A Dropout layer with a 0.2 rate and a Dense layer with regularization were included to prevent overfitting.

Table 6.2 LSTM Layer configuration

Units	Activation function	Return Sequence	Input Shape
64	ReLu	True	(timesteps, input_dim)
32	ReLu	False	(timesteps, input_dim)

After defining the architecture, the model was compiled using the Adam optimizer with a learning rate of 1e-3, which adjusts the model's weights during training by minimizing the loss function. The loss function, along with metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), was used to evaluate the model's performance.

6.4.3 Training Loop

The training process, encapsulated in the build_and_train_lstm() function, began by defining the EarlyStopping callback from TensorFlow/Keras. This callback monitored the validation loss (val_loss) during training and halted the process if no improvement was observed over 10 consecutive epochs (patience parameter set to 10). The restore_best_weights parameter was set to true, ensuring the model reverted to the best-performing state based on validation loss.

The model was trained using the model.fit() function, with X_train and y_train serving as the training data. The training was initially set for 200 epochs, although it could terminate earlier if the EarlyStopping callback was triggered. Validation data (X_val, y_val) was provided to evaluate the model's performance after each epoch, offering insights into its generalization capabilities. The verbose parameter was set to 1 to display a progress bar and relevant metrics during each epoch. Upon completing the training at epoch 182, the model was saved in HDF5 format (model.h5) for deployment, as mentioned in Section 6.3.2. Figure 6.2 illustrates the training loop results.

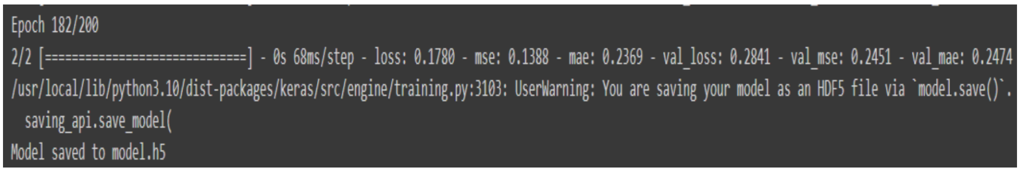

Figure 6.2 Model Training Result

6.4.4 Model Evaluation

To assess the trained LSTM model's performance, we conducted an evaluation using a separate test dataset from the hydroponic farm system. This evaluation step was critical in determining the model's ability to generalize to unseen data and its reliability in predicting real-world scenarios.

The test dataset underwent similar preprocessing steps, including data scaling and sequencing, as applied during the training phase. The model.evaluate() function was utilized to compute the test loss, MSE, and MAE, which reflect the model's prediction accuracy compared to the actual target values. The evaluation was conducted using X_test_seq (feature sequences) and y_test_seq (true target values). The results of the model evaluation are presented in Figure 6.3.

```
3/3 [==============================] - 1s 20ms/step - loss: 0.1610 - mse: 0.1192 - mae: 0.1769
Test Loss: 0.16104093194007874
Test MSE: 0.11917976289987564
Test MAE: 0.17693285644054413
```

Figure 6.3 Model Evaluation Result

6.5 Summary

Chapter 6 detailed the comprehensive implementation process of the Hydroponic Farm Monitoring Application, covering the frontend, backend, and model training components. The **frontend** implementation involved building the user interface with React Native, integrating Firebase for user authentication, and creating intuitive modules for monitoring and managing farm data. Key features included real-time data display, user profile management, and control panels for adjusting system parameters.

The **backend** was developed using Node.js and Express.js, with a focus on establishing robust APIs for data handling, CRUD operations, and machine learning model deployment. Firebase was integrated to manage database operations, file storage, and user notifications. The backend also facilitated anomaly detection and system control by interfacing with the machine learning model server.

The **model training** process was conducted using Google Collab, where an LSTM model was trained on historical sensor data to identify anomalies in the hydroponic system. The training involved data preprocessing, model definition, and evaluation, ensuring the model could accurately detect deviations and predict system behaviour.

This chapter demonstrates the successful integration of these components into a cohesive system that enhances the management of hydroponic farms. Future improvements could focus on optimizing algorithms, enhancing data handling, and refining the integration between various system components to further improve performance and user experience.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

This chapter outlines the testing activities conducted during the project development. It includes the test plan, and the results of various tests performed, including unit tests for the mobile application, API tests, and integration tests. The testing was designed to ensure that the system meets the requirements and functions reliably under different conditions.

7.2 Test Plan

7.2.1 Objectives

The objective of the test plan is to ensure that the application meets the requirements specified in Chapter 4.2, and to validate its performance and stability across different test scenarios, including application functional unit tests, API unit tests, and integration tests. The goal is to identify and resolve defects discovered during testing and to confirm that the application complies with the requirements specifications.

7.2.2 Test Scope

The test scope for this project includes the mobile application, app-server, and model-server. Testing for both frontend and backend was conducted in separate testing environments due to the different frameworks used during development. Additionally, performance testing was conducted to evaluate the mobile application's ability to efficiently monitor and control the hydroponic system remotely, as specified in the requirements. The following test activities were performed for this project:

- Unit testing of the mobile application
- Unit testing of the API
- Performance testing

7.2.3 Test Basis

The following sections from the report served as the basis for designing the test plan:

- Chapter 4.2 Requirements Specification
- Chapter 4.3 and 4.4 Use Case Diagram and Description
- Chapter 6.3.3 API Functions

7.2.4 Test Items

Table 7.1 lists the functional services of the system that were planned for testing.

Table 7.1 Functional Services to be tested

Mobile Application	
Functional Service	**Description**
Authentication Component	The main validation of this component is to test the Firebase Authentication that integrated with application able to login, register, send verification email, send password reset email, change email, and change password
Monitor Component	The main validation of this component is to test out the availability to send HTTP requests to the server and retrieve the data.
Control Component	The main validation of this component is to test out the availability to send HTTP requests to server for retrieving the data and updating the data
Plant Management Component	The main validation of this component is to test the create and update of the plant management able to operate correctly

Data Insight Component	The main validation of this component is to test the availability of data able to be displayed in multiple formats correctly.
Notification Component	The main validation of this component is to test the operation of retrieval and delete functions
App-Server	
Functional Service	**Description**
API Component	The main validation of this component is to ensure the API can effectively process the HTTP request and response to the request.
Cron Component	The main validation of this component is to ensure the services able to perform the action on the correct period
Model-Server	
Functional Service	**Description**
API Component	The main validation of this component is to ensure the API can effectively process the HTTP request and response to the request.

7.2.5 Test Strategy

The table 7.2 shown the overview of the test strategy used for this project.

Table 7.2 Testing Levels, Types and Tools

Testing Levels	**Testing Types**	**Tools**
Unit Test	Functional Testing	Android Emulator
API Test	Functional Testing	Postman
Performance Test	Performance Testing	Firebase Performance Monitoring

7.2.6 Test Criteria

Entry Criteria

The test can begin when the entry criteria listed are met.

- All features to be tested were completed and functionable.
- The testing tools and devices have been setup in the test environment.
- The test data are prepared for the API unit test

Exit Criteria

- All test case performed and successfully passed
- All defects found during the test phase are fix and solved
- No critical issue remaining

7.3 Functionality Test

Functionality test is one of the software testing processes which involves independent testing on function or components of a software system. The reason to perform functionality test is to validate that the function able to perform as same as the requirement specified and ensuring the unit of code are working correctly before integrates to whole system. The functionality test also able to enhance the documentation and maintainability for the software system as it describes the expected process and result of the specific functionality.

7.3.1 Mobile Application Functionality Test

This project performed the following functionality tests by manually inserting the values, clicking on the buttons, and performing any steps that stated in the test case on the emulator. A total of fourteen functionality tests are performed on the emulator by following the test steps with test data. The test status is recorded for every unit test after the expected result achieved. The test cases are recorded from Table 7.3 to Table 7.16.

Table 7.3 Test Case MA-1 User registration

Test Case ID	MA-1	**Test Case Name**	User Registration	**Component**	Authentication
Test Case Description	To validate that the user able to register a new account				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Register an account with a valid email, password and confirm password	1. Click register button	-	Verification Email sent	Verification Email sent	Pass
	2. Enter email	test@gmail.com			
	3. Enter password	123456789			
	4. Enter confirm password	123456789			

	5. Click register button	-			

Table 7.4 Test Case MA-2 User Login

Test Case ID	MA-2	**Test Case Name**	User Login	**Component**	Authentication
Test Case Description	To validate that the user able to login to their account				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Login account with valid email, and password	1. Enter email	test@gmail.com	Navigate to Add Farm Screen	Navigate to Add Farm Screen	Pass
	2. Enter password	123456789			
	3. Click on login button	-			
Login account with invalid email, and password	1. Enter email	Test1@gmail.com	Error message prompt	Error message prompt	Pass
	2. Enter password	123			
	3. Click on login button	-			
Forgot Password	1. Click forgot password button	-	Password reset email sent	Password reset email sent	Pass

	2. Enter email	test@gmail.com			
	3. Click send reset email	-			

Table 7.5 Test Case MA-3 Change Email

Test Case ID	MA-3	**Test Case Name**	Change Email	**Component**	Authentication
Test Case Description	To validate that the user able to change the email				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Change to new email address	1. Click change email button	-	HTTP request sent, Success Message Prompt and Verification Email sent	HTTP request sent, Success Message Prompt and Verification Email sent	Pass
	2. Enter new email	test2@gmail.com			
	5. Click confirm button	-			
Change to invalid email	1. Click change email button	-	Failed Message Prompt	Failed Message Prompt	Pass
	2. Enter invalid email	test2@gmail			

	5. Click confirm button	-			

Table 7.6 Test Case MA-4 Change Password

Test Case ID	MA-4	**Test Case Name**	Change Password	**Component**	Authentication
Test Case Description	To validate that the user able to change the password				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Change to new password	1. Click change password button	-	Success Message Prompt and logout the user	Success Message Prompt and logout the user	Pass
	2. Enter old password	123456789			
	3. Enter new password	abc123456			
	4. Click confirm button	-			
Use invalid old password to change new password	1. Click change password button	-	Failed Message Prompt	Failed Message Prompt	Pass
	2. Enter invalid old password	19191919			
	3. Enter new password	abc123456			

	4. Click confirm button	-			

Table 7.7 Test Case MA-5 Retrieve farm and sector data

Test Case ID	MA-5	**Test Case Name**	Retrieve farm and sector data	**Component**	Monitor
Test Case Description	To validate that the application able to retrieve farm and sector data				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Retrieve latest parameter data	1. Click on reload button	-	HTTP request sent, Parameters data display	HTTP request sent, Parameters data display	Pass
Retrieve sector tasks to do	1. Click on tasks tab	-	HTTP request sent, Tasks data display	HTTP request sent, Tasks data display	Pass
Retrieve user farm list	1. Click on edit farm option	-	HTTP request sent, Farm list display	HTTP request sent, Farm list display	Pass
Retrieve user sector list	1. Click on edit sector option	-	HTTP request sent, Sector list display	HTTP request sent, Sector list display	Pass

Table 7.8 Test Case MA-6 Update Sector Settings

Test Case ID	MA-6	**Test Case Name**	Update Sector Settings	**Component**	Control
Test Case Description	To validate that the user able to update sector's parameter and trigger settings				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Update Parameter Setting	1. Click on edit button for light intensity	-	HTTP request sent; the parameter lower boundary updated	HTTP request sent; the parameter lower boundary updated	Pass
	2. Edit Lower Boundary	150			
	3. Click save button	-			
Switch on trigger Setting	1. Switch on the High pH Trigger	-	HTTP request sent	HTTP request sent	Pass
Switch off trigger Setting	1. Switch off the High pH Trigger	-	HTTP request sent	HTTP request sent	Pass

Table 7.9 Test Case MA-7 Add Plant

Test Case ID	MA-7	**Test Case Name**	Add Plant	**Component**	Plant Management
Test Case Description	To validate that the user able to add new plant to sector				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Add Plant with valid input	1. Click on add button	-	HTTP request sent; Success message prompt	HTTP request sent; Success message prompt	Pass
	2. Click to add image	Image of plant			
	3. Enter plant name	Lettuce			
	4. Enter plant status	Seed			
	5. Select sector to add	Sector 1			
	6. Click on save button	-			
Add Plant with empty field	1. Click on add button	-	Error message prompt	Error message prompt	Pass
	2. Click to add image	-			
	3. Enter plant name	Lettuce			
	4. Enter plant status	-			
	5. Select sector to add	Sector 1			
	6. Click on save button	-			

Table 7.10 Test Case MA-8 Add Plant Record

Test Case ID	MA-8	**Test Case Name**	Add Plant Record	**Component**	Plant Management
Test Case Description	To validate that the user able to add new record for a plant				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Add Record with valid input	1. Click on add record button	-	HTTP request sent; record updated	HTTP request sent; record updated	Pass
	2. Enter observation	test			
	3. Enter measurement	0.5			
	4. Click on save button	-			
Add Record with empty field	1. Click on add record button	-	Invalid input message prompt	Invalid input message prompt	Pass
	2. Enter observation	-			
	3. Enter measurement	1			
	4. Click on save button	-			

Table 7.11 Test Case MA-9 Add Plant Task

Test Case ID	MA-9	**Test Case Name**	Add Plant Task	**Component**	Plant Management
Test Case Description	To validate that the user able to add new task for a plant				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Add Task with valid input	1. Click on add task button	-	HTTP request sent; task updated	HTTP request sent; task updated	Pass
	2. Enter date	2024-08-21			
	3. Enter task	test			
	4. Click on save button	-			
Add Task with empty field	1. Click on add task button	-	Invalid input message prompt	Invalid input message prompt	Pass
	2. Enter date	-			
	3. Enter task	1			
	4. Click on save button	-			
Add Task with invalid date format	1. Click on add task button	-	Invalid date format message prompt	Invalid date format message prompt	Pass
	2. Enter date	20240821			

	3. Enter task	Test			
	4. Click on save button	-			

Table 7.12 Test Case MA-10 Edit Plant Detail

Test Case ID	MA-10	**Test Case Name**	Edit Plant Detail	**Component**	Plant Management
Test Case Description	To validate that the user able to edit plant detail				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Edit Plant Status	1. Click on edit button	-	HTTP request sent; plant status updated	HTTP request sent; plant status updated	Pass
	2. Edit Plant Status	Germination			
	3. Click on save button	-			
Edit Plant Image	1. Click on edit button	-	HTTP request sent; plant image updated	HTTP request sent; plant image updated	Pass
	2. Select to change image	New Image File			
	3. Click on save button	-			

Table 7.13 Test Case MA-11 Data Insight Display

Test Case ID	MA-11	**Test Case Name**	Data Insight Display	**Component**	Data Insight
Test Case Description	To validate that the user able to view the data in daily and monthly format for selected sector				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Select daily data for sector	1. Click on Daily option	-	HTTP request sent; display one day parameter and anomaly data	HTTP request sent; display one day parameter and anomaly data	Pass
Select monthly data for sector	1. Click on Monthly option	-	HTTP request sent; display one month parameter and anomaly data	HTTP request sent; display one month parameter and anomaly data	Pass

Table 7.14 Test Case MA-12 Data Export

Test Case ID	MA-12	**Test Case Name**	Data Export	**Component**	Data Insight
Test Case Description	To validate that the user able to export the selected date parameter data				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Export parameter for sector	1. Click on download button	-	HTTP request sent; Success message prompt	HTTP request sent; Success message prompt	Pass
	2. Select the date	20-8-2024			
	3. Click on confirm button	-			

Table 7.15 Test Case MA-13 Notification Retrieval

Test Case ID	MA-13	**Test Case Name**	Notification Retrieval	**Component**	Notification
Test Case Description	To validate that the user able to refresh the notification				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Retrieve latest notification	1. Click on refresh button	-	HTTP request sent; notification list updated	HTTP request sent; notification list updated	Pass

Table 7.16 Test Case MA-14 Notification Delete

Test Case ID	MA-14	**Test Case Name**	Notification Delete	**Component**	Notification
Test Case Description	To validate that the user able to remove the notification				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Delete notification	1. Click edit button	-	HTTP request sent; notification list updated	HTTP request sent; notification list updated	Pass
	2. Select notification to delete	-			
	3. Click on delete selected notification button	-			

7.3.2 App Server Functionality Test

For the app server functionality test, there are two component which are API and Cron component. Postman is used to test with the app-server API component because it allows to send HTTP request without the client side. A total of thirteen unit and API tests are performed on the postman and development server by following the test steps with test data. The test status is recorded for every unit test after the expected result achieved. The test cases are recorded from Table 7.17 to Table 7.29.

Table 7.17 Test Case AS-1 Cron Job for Checking and Saving Notifications

Test Case ID	AS-1	**Test Case Name**	Cron Job for Checking and Saving Notifications	**Component**	Cron, API
Test Case Description	To validate that the Cron job able to perform the daily notification sending				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Verify Cron Job Scheduling	1. Start the app-server	-	Log: Cron Start	Log: Cron start	Pass
	2. Set the system time to 23:59.	-			
	3. Check logs for execution result	-			
	1. Select POST method	-			Pass

Verify Notification Check and Save Process	2. Set test data to request body	jNpmgDej52T9D758EoZyS0HF4Yl2	The notification created and saved to database, then send notification via Firebase Cloud Messaging	The notification created and saved to database, then send notification via Firebase Cloud Messaging	
	3. Send POST request	-			

Table 7.18 Test Case AS-2 Sector Status Update Cron Job

Test Case ID	AS-2	**Test Case Name**	Sector Status Update Cron Job	**Component**	Cron
Test Case Description	To validate that the Cron job able to perform the sector status update				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Verify Cron Job Scheduling	1. Start the app-server	-	Log: Sector Status Updated	Log: Sector Status Updated	Pass
	2. Set the system time to just before the next hour	-			

	3. Wait for the system time to reach the next hour	-			
	4. Check logs for execution result	-			

Table 7.19 Test Case AS-3 User Registration

Test Case ID	AS-3	**Test Case Name**	User Registration	**Component**	API
Test Case Description	To validate that the API able to register the user data to database				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**

Register with valid data	1. Send POST Request with valid request body	{
		userId: abc123
		farmList: []
		messageToken: abc123
		}
			The user documents created and saved the data	The user documents created and saved the data	Pass

Register with empty data	2. Send POST Request with empty request body	{}	Invalid request data	Invalid request data	Pass

Table 7.20 Test Case AS-4 Update Email

Test Case ID	AS-4	**Test Case Name**	Update Email	**Component**	API
Test Case Description	To validate that the API able to update the user email				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**

Update email with valid user ID	1. Send POST Request with valid request body	{
		userId: abc123
		newEmail: abc@gmail.com
		}
			Status 200, Email Updated Successfully	Status 200, Email Updated Successfully	Pass

Update email with empty data	2. Send POST Request with empty request body	{}	Failed to update email	Failed to update email	Pass

Table 7.21 Test Case AS-5 Check and Update Message Token

Test Case ID	AS-5	**Test Case Name**	Check and Update Message Token	**Component**	API
Test Case Description	To validate that the API able to check and update message token				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Check and update message token with valid data	1. Send POST Request with valid request body	{ userId: abc123 messageToken: abc123 }	Status 200, Token Check Successfully	Status 200, Token Check Successfully	Pass
Check and update message token with empty data	2. Send POST Request with empty request body	{}	No message token or user ID found	No message token or user ID found	Pass

Table 7.22 Test Case AS-6 Update Notification Settings

Test Case ID	AS-6	**Test Case Name**	Update Notification Settings	**Component**	API
Test Case Description	To validate that the API able to update user's notification settings				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**

Update notification settings with valid data	1. Send POST Request with valid request body	{
		userId: abc123
		notificationSettings: true, true, true
		}
			Status 200, Notification settings updated successfully	Status 200, Notification settings updated successfully	Pass

Update notification settings with empty data	2. Send POST Request with empty request body	{}	Invalid request data	Invalid request data	Pass

Table 7.23 Test Case AS-7 Get Sector Latest Data

Test Case ID	AS-7	**Test Case Name**	Get Sector Latest Data	**Component**	API
Test Case Description	To validate that the API able to get specific sector latest data				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Get sector latest data with valid data	1. Send GET Request with valid request params (sectorID)	tRm5N2mF8oqEF7jl1Xi4	latestData object	latestData object	Pass
Get sector latest data with empty data	2. Send GET Request with empty request params	{}	Error	Error	Pass

Table 7.24 Test Case AS-8 Update Parameter Settings

Test Case ID	AS-8	**Test Case Name**	Update Parameter Settings	**Component**	API
Test Case Description	To validate that the API able to update the sector’s parameter settings				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**

Update parameter settings with valid data	1. Send POST Request with valid request body	{
		sectorId: abc123,
		parameterSettings
		}
			Status 200, Update Successfully	Status 200, Update Successfully	Pass

Update parameter settings with empty data	2. Send POST Request with empty request body	{}	Error updating parameter data	Error updating parameter data	Pass

Table 7.25 Test Case AS-9 Add Sector

Test Case ID	AS-9	**Test Case Name**	Add Sector	**Component**	API
Test Case Description	To validate that the API able to add a new sector				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Add sector with valid data	1. Send POST Request with valid request body	{farmId; abc deviceId: device userId: abc123}	Status 200, return sectorId	Status 200, return sectorId	Pass

Add sector with empty data	2. Send POST Request with empty request body	{}	Farm ID, Device ID, and User ID are required!	Farm ID, Device ID, and User ID are required!	Pass

Table 7.26 Test Case AS-10 Update Parameter Data

Test Case ID	AS-10	**Test Case Name**	Update Parameter Data	**Component**	API
Test Case Description	To validate that the API able to update the sector's parameter data				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Update parameter data with valid data	1. Send POST Request with valid request body	{userId: abc123, sectorId: abc123, parameters}	Status 200, Update Successful	Status 200, Update Successful	Pass
Update parameter data with empty data	2. Send POST Request with empty request body	{}	Error updating parameter data	Error updating parameter data	Pass

Table 7.27 Test Case AS-11 Post Trigger Result

Test Case ID	AS-11	**Test Case Name**	Post Trigger Result	**Component**	API
Test Case Description	To validate that the API able to post the trigger result to database				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Post the trigger result with valid data	1. Send POST Request with valid request body with params userID	{triggerType: highPh, Status: success Detail: ON }	Status 200, Execution result recorded	Status 200, Execution result recorded	Pass
Post the trigger result with empty data	2. Send POST Request with empty request body and params	{}	User not found	User not found	Pass

Table 7.28 Test Case AS-12 Register Device

Test Case ID	AS-12	**Test Case Name**	Register Device	**Component**	API
Test Case Description	To validate that the API able to register device (hydroponic system) to database				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Register device with valid data	1. Send POST Request with valid request body	{deviceName: abc, deviceLocation: Sg Long}	Status 200, return device ID	Status 200, return device ID	Pass
Register device with empty data	2. Send POST Request with empty request body	{}	Name and location are required	Name and location are required	Pass

Table 7.29 Test Case AS-13 Add Plant

Test Case ID	AS-13	**Test Case Name**	Add Plant	**Component**	API
Test Case Description	To validate that the API able to add a new plant for a specific sector				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Add plant with valid data	1. Send POST Request with valid request body and params (sectorId)	{name: lettuce, seed}, Image file	Status 200, Plant added successfully	Status 200, Plant added successfully	Pass
Add plant with empty data	2. Send POST Request with empty request body	{}	Name, status, and image are required.	Name, status, and image are required.	Pass

7.3.3 Model Server Functionality Test

For the model server functionality test, there is only one component to test which is the API component. Postman is used to test with the model-server API component because it allows to send HTTP request without the need of client side. Two API test are performed as recorded the actual result with test status at Table 7.30 and Table 7.31.

Table 7.30 Test Case MS-1 Receive Data for Anomaly Detection

Test Case ID	MS-1	**Test Case Name**	Receive Data for Anomaly Detection	**Component**	API
Test Case Description	To validate that the API able to return the anomaly detection result				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Anomaly Detection with valid data	1. Send POST Request with valid request body	{latestData}	{summary, predictions, actual_values}	{summary, predictions, actual_values}	Pass
Anomaly Detection with empty data	2. Send POST Request with empty request body	{}	"error": "No data received"	"error": "No data received"	Pass

Table 7.31 Test Case MS-2 Predict Trigger Status

Test Case ID	MS-2	**Test Case Name**	Predict Trigger Status	**Component**	API
Test Case Description	To validate that the API able to return trigger status				
Test Item	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Test Status**
Predict trigger status with valid data	1. Send POST Request with valid request body	{latestData}	{predictions, trigger_status}	{predictions, trigger_status}	Pass
Predict trigger status with empty data	2. Send POST Request with empty request body	{}	"error": "No data received"	"error": "No data received"	Pass

7.4 Performance Test

Performance testing is a critical aspect of mobile application testing, ensuring that the application delivers a seamless and user-friendly experience. This testing assesses the scalability, stability, and responsiveness of the application under several scenarios. For this project, performance testing focused on validating the application's capability to monitor and control the hydroponic system effectively. Firebase Performance Monitoring was used to perform real-time performance testing, providing insights into application performance. Testing was conducted using a physical smartphone model (HONOR 9X) connected via Wi-Fi. The key areas of performance testing included:

- Response time
- App start time
- Response success rate
- Frozen frames percentage

7.4.1 Response Time

Response time is a crucial metric that measures the time elapsed between a request or query being sent and the corresponding response being received. Faster response times generally indicate better application performance and higher user satisfaction. In this context, response time can be influenced by factors such as network latency, server load, and software efficiency. Figure 7.1 shows the response time trend for the application server deployed on an Amazon Web Services (AWS) instance with IP address 13.229.207.3.

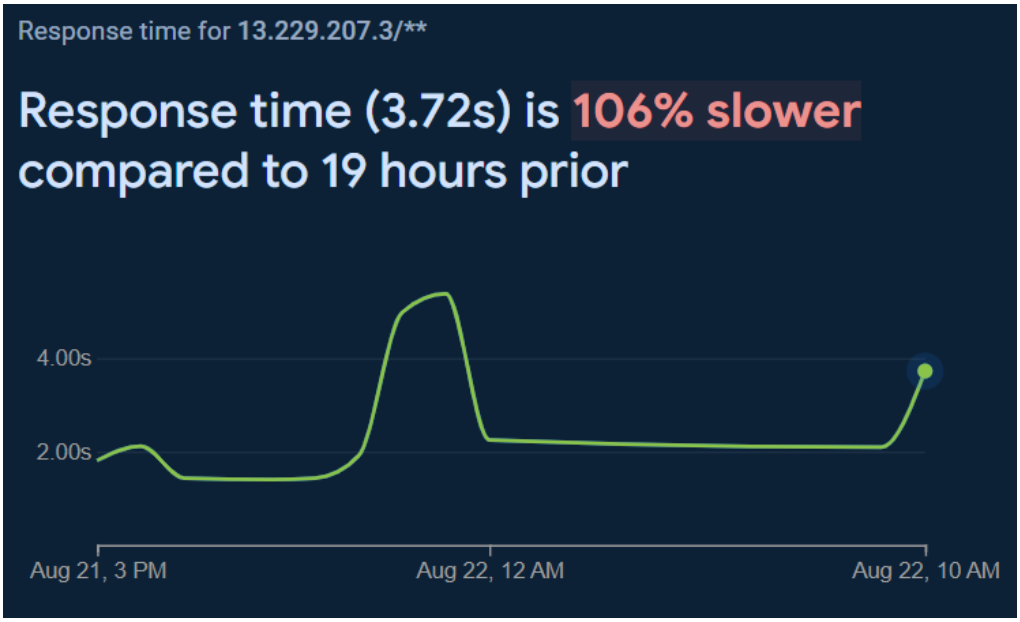

Figure 7.1 19 Hours Response Time Trend from IP 13.229.207.3

As illustrated in Figure 7.1, the response time for IP address 13.229.207.3/** has significantly increased over the past 19 hours, rising from approximately 2 seconds to around 4 seconds. The current response time of 3.72 seconds is 106% slower compared to the response time recorded 19 hours earlier.

7.4.2 App Start Time

App start time measures the duration it takes for an application to become fully functional and ready for user interaction after launch. A faster app start time enhances user experience by reducing waiting time. Factors affecting app start time include code complexity, resource loading, and device performance. Figure 7.2 presents the app start time trend over a 19-hour period.

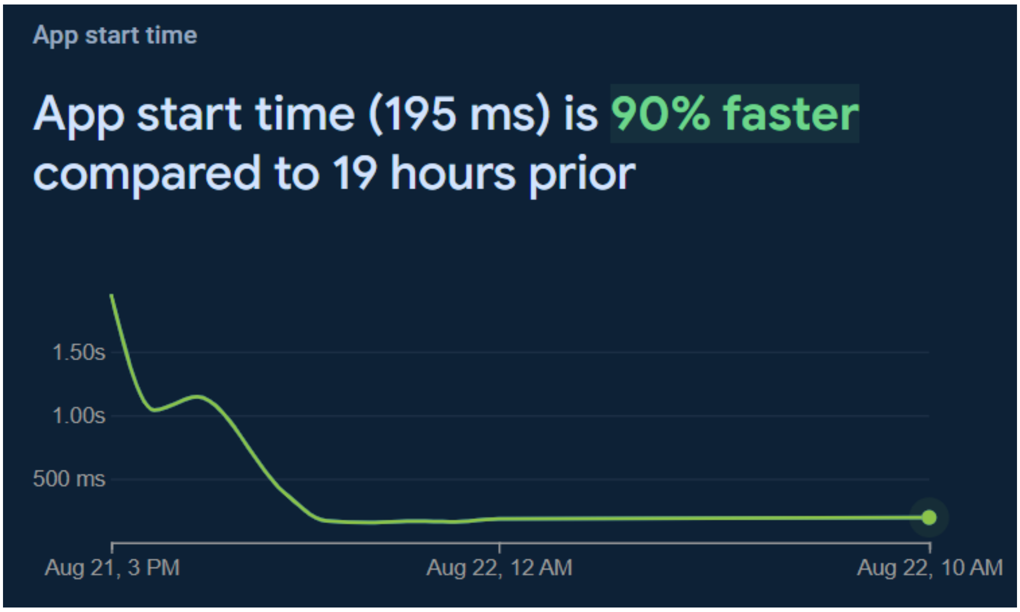

Figure 7.2 19 Hours App Start Time

As shown in Figure 7.2, the app start time has improved significantly over the past 19 hours, decreasing from approximately 1.5 seconds to 195 milliseconds (ms), indicating a 90% improvement. The initially higher start time might have been caused by temporary fluctuations in measurement accuracy or timing.

7.4.3 Response Success Rate

The response success rate measures the percentage of successful outcomes or attempts within a given context. This metric can be influenced by various factors, including network connectivity, data transfer efficiency, and overall application performance. A high success rate indicates that the application is reliable and efficient. Figure 7.3 shows the response success rate for the application server with IP 13.229.207.3 over the past 19 hours.

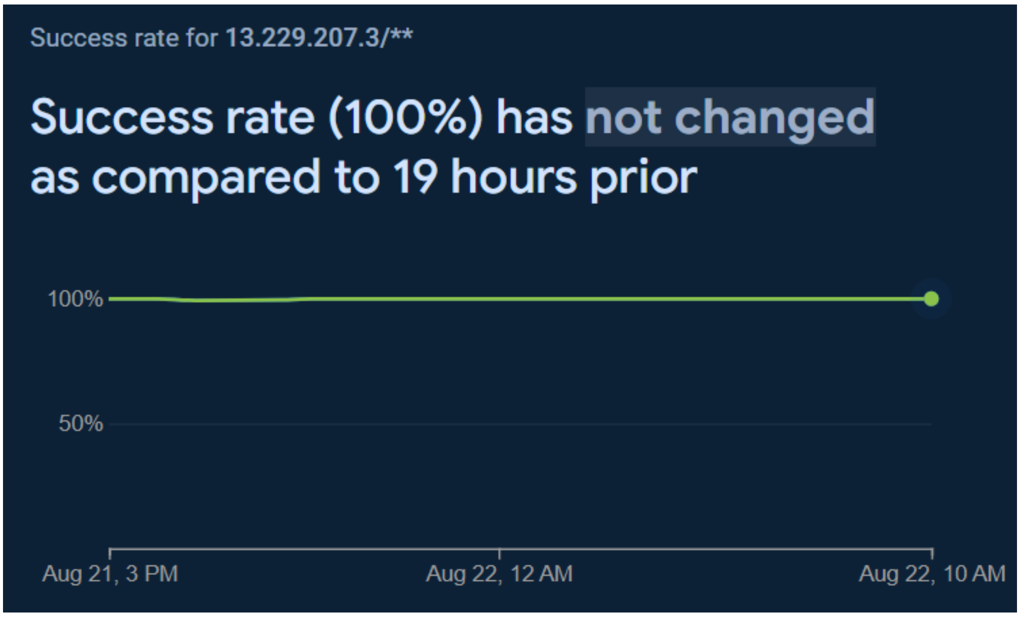

Figure 7.3 19 Hours Response Success Rate from IP 13.229.207.3

According to Figure 7.3, the response success rate for IP address 13.229.207.3/** has remained constant at 100% over the past 19 hours. This suggests that there were no failures or errors in the system or network connection associated with this IP address during the specified timeframe.

7.4.4 Frozen Frames Percentage

Frozen frames are a performance metric indicating instances where an application's user interface becomes unresponsive or freezes, potentially degrading user experience. The testing included three instances: MainActivity (the initial and main user interface), dashboardScreen (monitor panel), and reportScreen (data and insight). Figures 7.4 to 7.6 show the results for MainActivity, dashboardScreen, and reportScreen, respectively. All instances reported a frozen frame percentage of 0% over the testing period, indicating that the application did not experience any UI freezes, thereby suggesting stable and reliable performance.

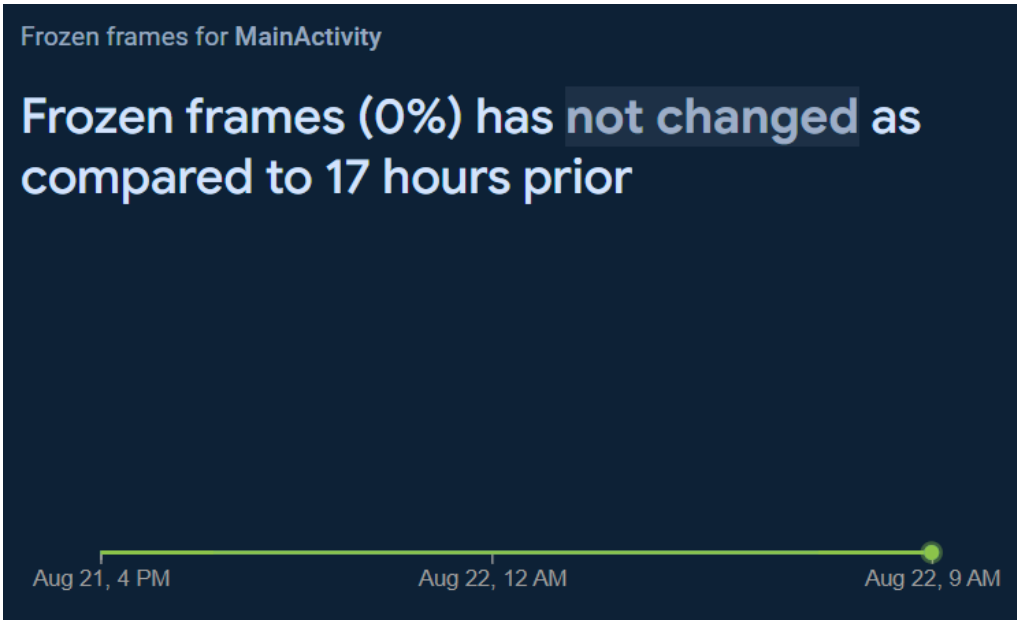

Figure 7.4 17 Hours Frozen Frame Percentage for MainActicity instance

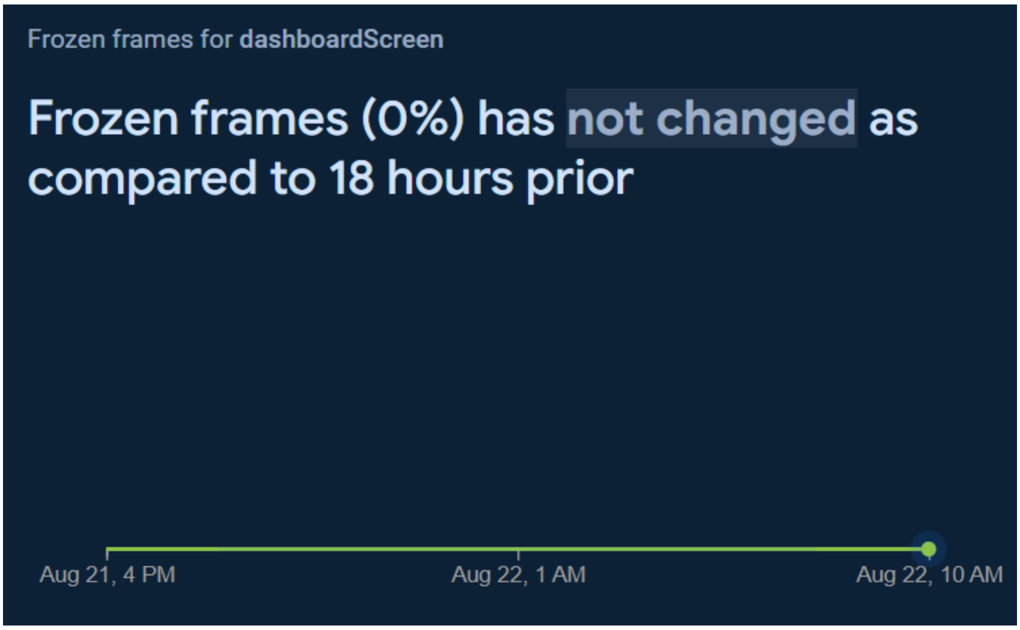

Figure 7.5 18 Hours Frozen Frame Percentage for dashboardScreen instance

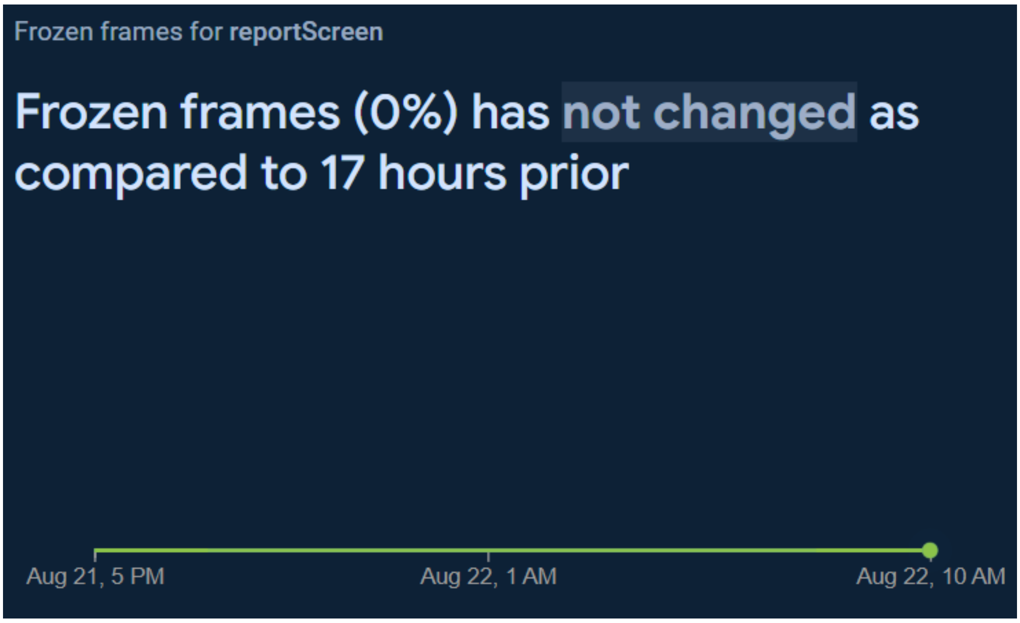

Figure 7.6 17 Hours Frozen Frame Percentage for reportScreen instance

7.5 Anomaly Detection Evaluation

For the evaluation of anomaly detection models, a dataset of 2 weeks for training and a dataset of 4 weeks for testing are used to trained and identify the abnormal pattern of the hydroponic farm environment. Since these models are trained unsupervised, it is not possible to quantify accuracy and dependability with conventional metrics such as Precision, Recall, or F1-Score because there is not enough labeled data. Rather, the assessment concentrated on contrasting the consistency of anomaly detection between various datasets and models. This evaluation employed four different approaches: LSTM Autoencoder-Based, LSTM Prediction-Based, LSTM with Isolation Forest, and LSTM with One-Class Support Vector Machine (SVM). Every model utilized a distinct approach to identify abnormalities and underwent testing to confirm its resilience and ability to apply to previously unseen data.

The LSTM Prediction-Based approach identifies anomalies by calculating the deviations between predicted values and actual observations. The model has identified 6 anomalies on the 2-week training dataset, as displayed in Figure 7.7. Similarly, the LSTM Autoencoder-Based approach which relies on reconstruction errors to flag deviations also detected 6 anomalies using a threshold set at the 95th percentile of the mean squared error (1.1559), as shown in Figure 7.8. The similarity of the two models results in consistency on capturing deviations in short-term datasets thus suggesting that both prediction and reconstruction methods can reliably identify key anomalies.

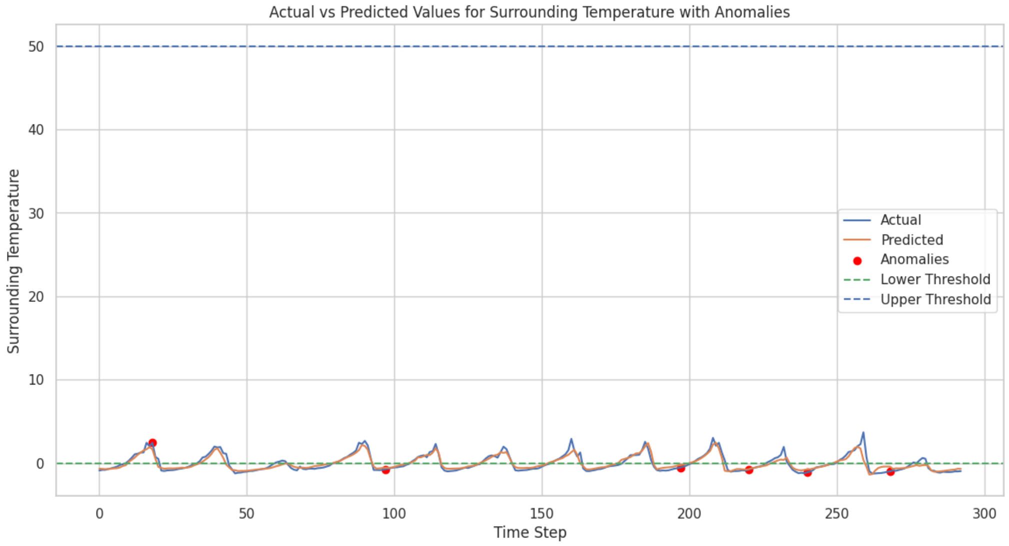

Figure 7.7 LSTM Prediction Based 2 Weeks Anomaly Detection Result

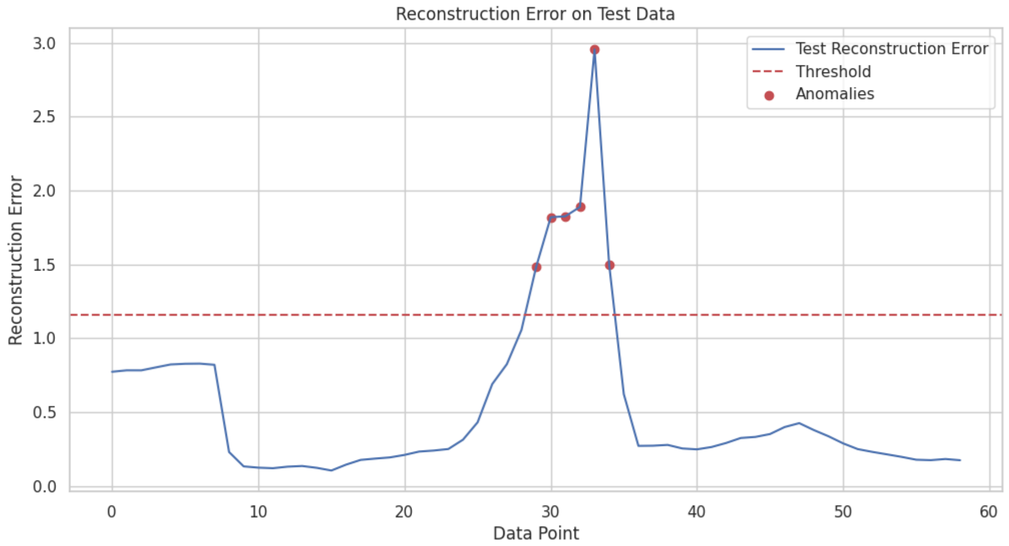

Figure 7.8 LSTM Autoencoder Based 2 Weeks Anomaly Detection Result

Moreover, LSTM Autoencoder was combined with Isolation Forest and One-Class SVM to perform model comparison and resulting slightly higher number of anomalies — 7 in total — were detected in the 2-week dataset or 12% of the total data points, as depicted in Figure 7.9. This increase indicates that combining reconstruction errors with clustering-based techniques may increase sensitivity to subtle variations, potentially identifying more nuanced anomalies that would be missed by simpler models or false positives.

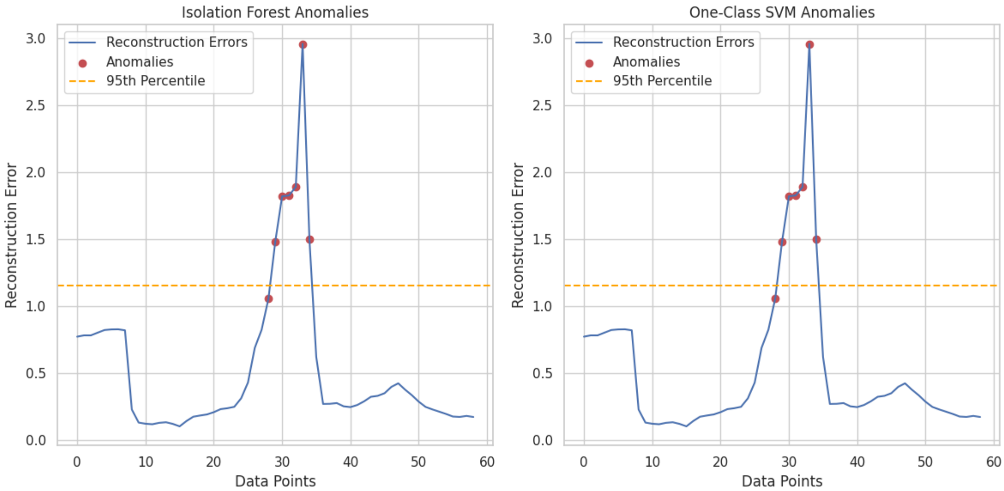

Figure 7.9 Isolation Forest and One Class Support Vector Machine 2 Weeks Anomaly Detection Result

Subsequently, a larger 4-week dataset was used to test the models' scalability and dependability over an extended period of time. As seen in Figure 7.10, the LSTM Prediction-Based model has identified 14 anomalies, suggesting that it retains its anomaly detection ability even when subjected to longer data sequences. Furthermore, the LSTM Autoencoder-Based method produced comparable outcomes, identifying 15 anomalies throughout the same time frame (Figure 7.11), illustrating its stability performance on both long- and short-term datasets.

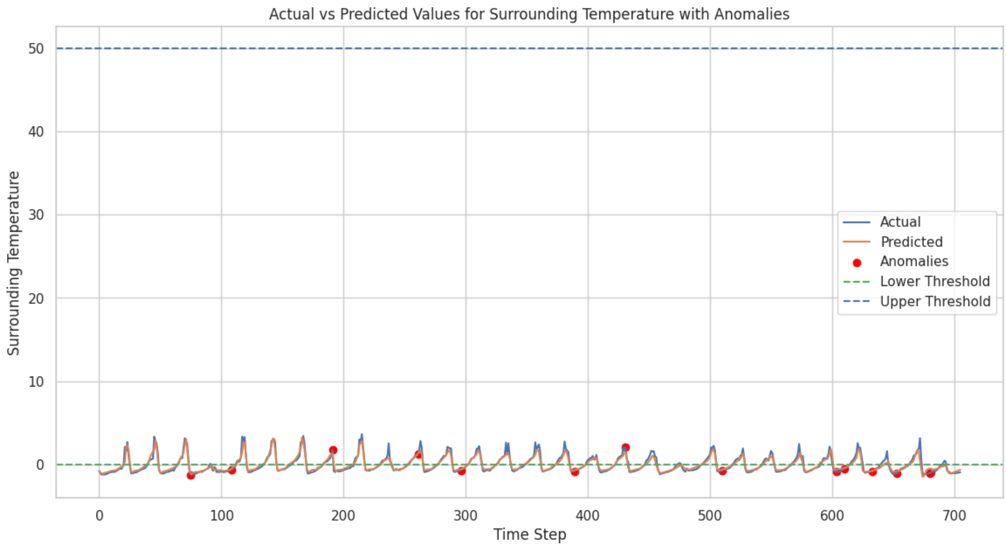

Figure 7.10 LSTM Prediction Based 4 Weeks Anomaly Detection Result

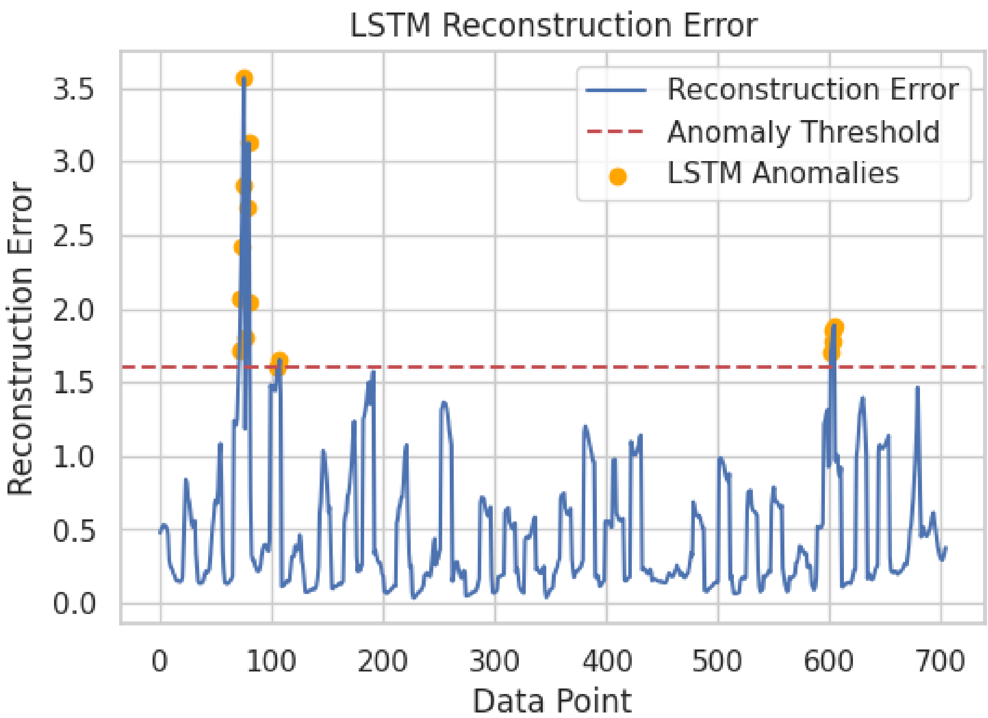

Figure 7.11 LSTM Autoencoder Based 4 Weeks Anomaly Detection Result

As demonstrated in Figures 7.12 and 7.13, the LSTM Autoencoder with Isolation Forest and One-Class SVM, on the other hand, demonstrates a notable increase in detected anomalies observed. The One-Class SVM model detected 70 anomalies, and the Isolation Forest model identified 71 anomalies, representing roughly 18% of the 4-week dataset. An increased detection capability but also a larger rate of false positives could result from these hybrid models' heightened sensitivity to deviations in reconstruction errors, as suggested by the growing number of anomaly counts.

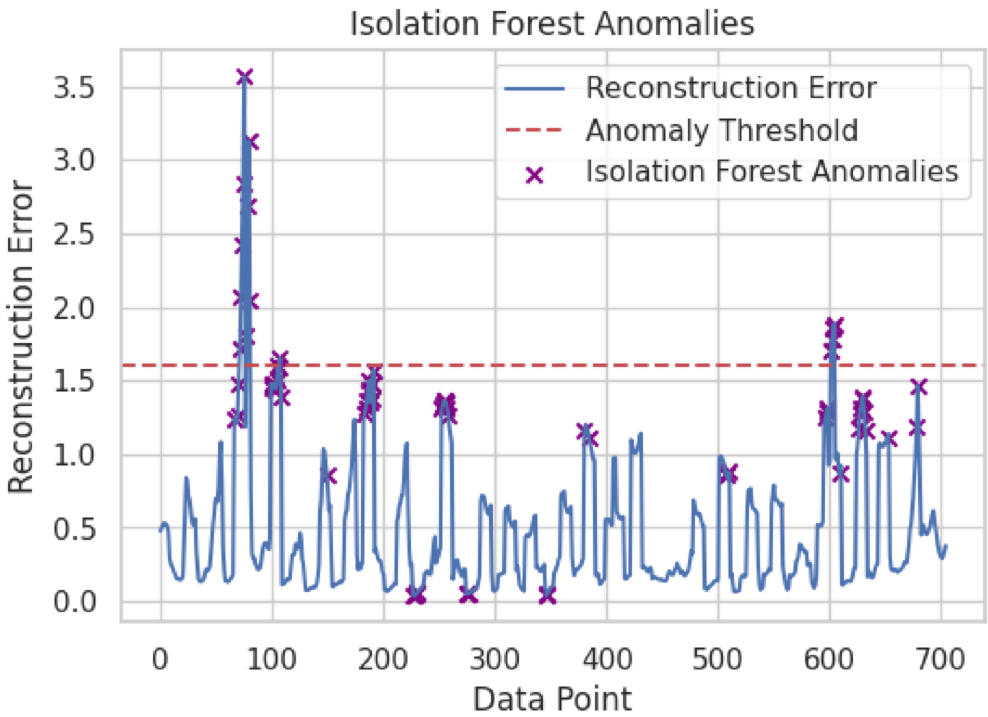

Figure 7.12 Isolation Forest 4 Weeks Anomaly Detection Result

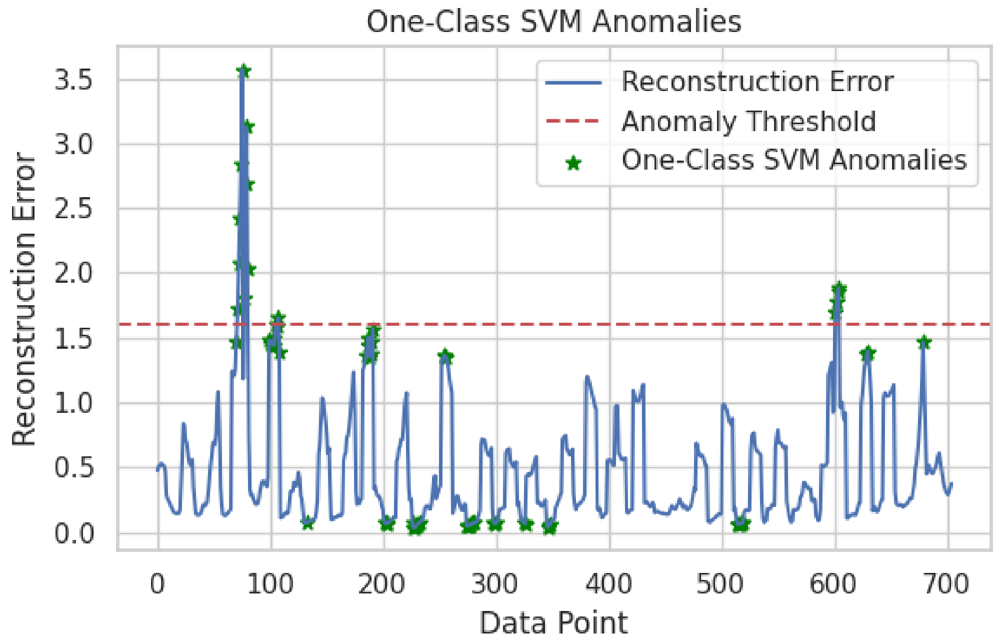

Figure 7.13 One Class Support Vector Machine 4 Weeks Anomaly Detection Result

Ultimately, even though the LSTM Prediction-Based and LSTM Autoencoder-Based models performed consistently and dependably in both short- and long-term assessments, the integration with One-Class SVM and Isolation Forest led to an increased sensitivity to anomalies. This shows that hybrid approaches like this might work better in situations when it's important to capture every potential abnormality. One should consider the possible false positive trade-off when applying these models in real-world scenarios. The models' scalability and resilience are further demonstrated by the evaluation on the larger dataset, which makes them the best options for ongoing monitoring in hydroponic systems, where the timely identification of anomalies is essential to preserving perfect environmental conditions.

7.6 Summary

In summary, the testing phase of this project thoroughly validated the system's functionality, performance, and reliability. Unit tests ensured that individual components met their expected requirements, while API tests verified the interactions between external systems and the application. Additionally, performance tests evaluated the system's responsiveness under load. Through these tests, critical issues were identified and resolved, thereby enhancing the system's overall quality and ensuring it meets the specified requirements for deployment. Moreover, evaluation on anomaly detection also have been performed to ensuring the reliability of the anomaly detection result.

CHAPTER 8

CONCLUSION AND FUTURE IMPROVEMENT

8.1 Conclusion

An Android mobile application that addresses the challenges mentioned in Chapter 1.3 has been developed successfully by this project. Firstly, the issue faced by urban dwellers—lack of time to manage and monitor hydroponic farms—has been effectively resolved. Additionally, the application has tackled the challenges associated with the farming skills gap and the implementation of technology in hydroponic farm monitoring. By leveraging a trained AI model, the application automates the management of hydroponic farm environmental parameters, thus achieving the following project objectives:

- **Develop a mobile application capable of monitoring and controlling hydroponic farming systems remotely.**

- **Develop a mobile application that can notify users of critical issues and required tasks within the hydroponic farming systems.**

- **Utilize machine learning with IoT and ICT technologies for detecting normal and abnormal environmental patterns in hydroponic farming, enabling automated adjustments and simplifying management.**

The first objective was met by completing the mobile application with the functionality to monitor and control the hydroponic farm system remotely. Users are relieved from constantly overseeing the system, as the application provides real-time environmental parameters and trends. Additionally, the application allows remote control of the hydroponic system, enhanced with AI-driven automation based on model predictions.

The second objective was achieved by integrating Firebase Cloud Messaging services into the application. This service enables the application server to send notifications based on specific conditions, such as positive anomaly detection, scheduled tasks, daily reminders, and successful trigger notifications. This feature ensures that users are promptly informed of critical or important events, allowing them to take necessary actions without delay.

Lastly, to fulfil the third objective, data from the hydroponic farm system was used to train the Long Short-Term Memory (LSTM) model. With the support of data provided by Chua Shi Jian’s hydroponic farm system from the Electrical and Electronic Engineering course (3E), the LSTM model was successfully trained and deployed to the model server. This deployment allows the model to identify abnormal patterns in real-time data, based on the training outcomes.

8.2 Limitations

Despite the success in achieving the project objectives, certain limitations remain that could be addressed in future work.

The first limitation is the limited size and duration of the dataset used for training the machine learning model. The dataset was restricted to one month of data, which, while sufficient for establishing initial patterns for anomaly detection, may not provide comprehensive insights into long-term environmental variations. A larger dataset spanning multiple months or years would improve the model's ability to generalize across different conditions and yield more accurate predictions.

The second limitation is the specificity of the dataset. The model was trained primarily on data from a hydroponic farm growing lettuce, which limits its applicability to other crops. Different crops, such as cabbage or chili, have unique environmental and nutrient requirements that the current model may not adequately address. To make the application more versatile, it would be necessary to gather data specific to other crops and retrain the model accordingly.

The third limitation is the requirement for manual processes such as harvesting and refilling nutrient solutions. While the application automates the control of environmental parameters, human intervention is still needed for some farm tasks. This reliance on manual processes could introduce inefficiencies and delays, reducing the overall potential for automation in system monitoring.

8.3 Recommendation for Future Improvements

To enhance the capabilities of this project, several future improvements are recommended:

Expand the Dataset for Enhanced Model Accuracy: Collecting a larger volume of data over a more extended period will enable the model to recognize a broader range of patterns and variations, leading to more accurate predictions and anomaly detection. An expanded dataset will also allow the system to generalize across a wider range of environmental conditions.

Support for a Variety of Crops: Incorporating data from a wider range of vegetables or plants into the model training will allow the application to recognize the different environmental and nutrient requirements for each type of crop. Implementing a crop-selection feature would further enhance the system's flexibility, enabling farmers to switch between different crops and receive tailored monitoring and control settings for each one.

Prediction and Automation of Farm Tasks: By training more advanced machine learning models, it may be possible to predict certain maintenance tasks, such as nutrient refilling or plant harvesting, based on historical data, and automate these processes without human intervention. For example, if the model predicts that nutrient depletion is likely within a few days, the system could notify the user to prepare for refilling or even automate the refilling process via IoT-connected devices.

REFERENCES

Abdallah, M., Wo Jae Lee, Raghunathan, N., Charilaos Mousoulis, Sutherland, J.W. and Bagchi, S. (2021). Anomaly Detection through Transfer Learning in Agriculture and Manufacturing IoT Systems. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2102.05814.

Adkisson, M., Kimmell, J.C., Gupta, M. and Abdelsalam, M. (2021). Autoencoder-based Anomaly Detection in Smart Farming Ecosystem. doi:https://doi.org/10.1109/bigdata52589.2021.9671613.

Baheti, P. (2021). *Train, Validation, and Test Set: How to Split Your Machine Learning Data*. [online] V7labs.com. Available at: https://www.v7labs.com/blog/train-validation-test-set.

Bandar Alanazi and Ibrahim Alrashdi (2023). Anomaly Detection in Smart Agriculture Systems on Network Edge Using Deep Learning Technique. *Sustainable Machine Intelligence Journal*, 3. doi:https://doi.org/10.61185/smij.2023.33104.

Bhandari, P. (2021). *What Is Data Cleansing? | Definition, Guide & Examples*. [online] Scribbr. Available at: https://www.scribbr.com/methodology/data-cleansing.

Brownlee, J. (2016). *How to Normalize and Standardize Time Series Data in Python*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/normalize-standardize-time-series-data-python/.

Catalano, C., Paiano, L., Calabrese, F., Cataldo, M., Mancarella, L. and Tommasi, F. (2022). Anomaly detection in smart agriculture systems. *Computers in Industry*, 143, p.103750. doi:https://doi.org/10.1016/j.compind.2022.103750.

Department of Statistics Malaysia (2022). DEPARTMENT OF STATISTICS MALAYSIA KEY FINDINGS OF POPULATION AND HOUSING CENSUS OF MALAYSIA 2020: URBAN AND RURAL. [online] Available at: https://v1.dosm.gov.my/v1/index.php?r=column/pdfPrev&id=ZFRzTG 9ubTkveFR4YUY2OXdNNk1GZz09 [Accessed 15 Feb. 2024].

Department of Statistics, Malaysia (2022). ICT Use and Access by Individuals and Households Survey Report, Malaysia, 2021. [online] v1.dosm.gov.my. Available at: https://v1.dosm.gov.my/v1/index.php?r=column/cthemeByCat&cat=395&bul_id=bCs4UlNSQktybTR3THZ3a0RzV2RkUT09&menu_id=amVoWU54UTl0a21NWmdhMjFMMWcyZz09.

Developer.com. (2022). *What is Lean Development?* [online] Available at: https://www.developer.com/project-management/what-is-lean-development/.

Encyclopedia Britannica. (n.d.). *Hydroponics | horticulture*. [online] Available at: https://www.britannica.com/topic/hydroponics.

Gallagher, A., Dunleavy, J. and Reeves, P. (2019). *The Waterfall Model: Advantages, disadvantages, and when you should use it*. [online] IBM Developer. Available at: https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/.

GeeksforGeeks (2019). *What is LSTM Long Short Term Memory?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory [Accessed 18 Mar. 2024].

GeeksforGeeks. (2019). *Machine Learning for Anomaly Detection*. [online] Available at: https://www.geeksforgeeks.org/machine-learning-for-anomaly-detection/?ref=header_search.[Accessed 18 Mar. 2024].

GeeksforGeeks. (2024). *Understanding One-Class Support Vector Machines*. [online] Available at: https://www.geeksforgeeks.org/understanding-one-class-support-vector-machines/.[Accessed 18 Mar. 2024].

GeeksforGeeks. (2024). *Unified Process in OOAD*. [online] Available at: https://www.geeksforgeeks.org/unified-process-in-ooad/. [Accessed 23 Mar. 2024].

Human Resources Development Fund (2019). AGRICULTURE SECTOR OVERVIEW: Will Agriculture Be the next Sector Covered under PSMB Act? [online] Available at: https://hrdcorp.gov.my/wp-content/uploads/2021/03/08.-issue_July02_2019-Human-Capital-Report-Agriculture-Sector-Overview.pdf.

Kaur, G., Upadhyaya, P. and Chawla, P. (2022). IoT Based Mobile Application for Monitoring of Hydroponic Vertical Farming. *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. doi:https://doi.org/10.1109/icrito56286.2022.9964872.

Khare, P., Abhishek Koti and Khare, A. (2023). Solar-smart hydroponics farming with IoT-based AI controller with mobile app. doi:https://doi.org/10.1109/ls1858153.2023.10170491.

knowledge.hannainst.com. (n.d.). *What is the relationship between TDS and EC?* [online] Available at: https://knowledge.hannainst.com/en/knowledge/ec-tds-what-is-the-relationship-between-tds-and-ec.

Kularbphettong, K., Ampant, U. and Kongrodj, N. (2019). An Automated Hydroponics System Based on Mobile Application. *International Journal of Information and Education Technology*, [online] 9(8), pp.548–552. doi:https://doi.org/10.18178/ijiet.2019.9.8.1264.

Lakshmanan, R., Djama, M., Perumal, S. and Abdulla, R. (2020). Automated smart hydroponics system using internet of things. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(6), p.6389. doi:https://doi.org/10.11591/ijece.v10i6.pp6389-6398.

Laoyan, S. (2024). *What Is Agile Methodology? (A Beginner's Guide)*. [online] Asana. Available at: https://asana.com/resources/agile-methodology.

Liu, F.T., Ting, K.M. and Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*. doi:https://doi.org/10.1109/icdm.2008.17.

MUHAMMAD, R. M. & RABU, M. R. 2015. The potential of urban farming technology in Malaysia: Policy intervention. FFTC Agricultural Policy Article.

Peuchpanngarm, C., Srinitiworawong, P., Samerjai, W. and Sunetnanta, T. (2016). *DIY sensor-based automatic control mobile application for hydroponics*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICT-ISPC.2016.7519235.

Polanitzer, R. (2022). *The Minimum Mean Absolute Error (MAE) Challenge*. [online] Medium. Available at: https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031.

Rahimi, M.K.H., Saad, M.H.M., Hussain, A. and Hamdan, N.M. (2022). Secure Cloud Connected Indoor Hydroponic System via Multi-factor Authentication. *International Journal of Advanced Computer Science and Applications*, 13(9). doi:https://doi.org/10.14569/ijacsa.2022.0130925.

Ramakrishnam Raju, S.V.S., Dappuri, B., Ravi Kiran Varma, P., Yachamaneni, M., Verghese, D.M.G. and Mishra, M.K. (2022). Design and Implementation of Smart Hydroponics Farming Using IoT-Based AI Controller with Mobile Application System. *Journal of Nanomaterials*, 2022, pp.1–12. doi:https://doi.org/10.1155/2022/4435591.

Rashid, T.A., Fattah, P. and Awla, D.K. (2018). Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms. *Procedia Computer Science*, 140, pp.324–333. doi:https://doi.org/10.1016/j.procs.2018.10.307.

Shin, K.G., Tan Ping Ping, Ling, Chong Chee Jiun and Noor Alamshah Bolhassan (2024). SMART GROW – Low-cost automated hydroponic system for urban farming. *HardwareX*, 17, pp.e00498–e00498. doi:https://doi.org/10.1016/j.ohx.2023.e00498.

trident (2021). *12 Software Development Methodologies: When, Where and How to Use Them*. [online] Trident Technolabs. Available at: https://tridenttechnolabs.com/12-software-development-methodologies-when-where-and-how-to-use-them/. [Accessed 23 Mar. 2024].

World Bank Open Data. (n.d.). *World Bank Open Data*. [online] Available at: https://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS?end=2022&locations=MY&most_recent_year_desc=true&start=2022&view=bar. [Accessed 15 Feb. 2024].

www.productplan.com. (n.d.). *What is Feature Driven Development (FDD)? | Definition*. [online] Available at: https://www.productplan.com/glossary/feature-driven-development/.

www.westlab.com. (n.d.). *What is EC? How can you differentiate between EC and TDS?* [online] Available at: https://www.westlab.com/blog/what-is-ec-how-can-you-differentiate-between-ec-and-tds.