

AUTOMATED  
FERTIGATION  
MOBILE APPLICATION  
(MOBILE-AFS)

LOH ZI HAO

Degree in Software Engineering

UNIVERSITI MALAYSIA PAHANG

## UNIVERSITI MALAYSIA PAHANG

### DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : LOH ZI HAO

Date of Birth

Title : AUTOMATED FERTIGATION MOBILE  
APPLICATION (MOBILE\_AFS)

Academic Session : 2022/2023

I declare that this thesis is classified as:


- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)\*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)\*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

\_\_\_\_\_  
(Student's Signature)

  
\_\_\_\_\_  
(Supervisor's Signature)

\_\_\_\_\_  
New IC/Passport Number  
Date:

MUHAMMAD ZULFAHMI TOH  
LECTURER  
Name of Supervisor  
FACULTY OF COMPUTING  
UNIVERSITI MALAYSIA PAHANG  
26600 PEKAN  
PAHANG DARUL MAKMUR

NOTE : \* If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.



## SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree in Software Engineering.

A handwritten signature in black ink, consisting of a large, stylized initial 'S' followed by a smaller, more complex signature.

---

(Supervisor's Signature)

Full Name :

Position :

Date :

---

(Co-supervisor's Signature)

Full Name :

Position :

Date :



## **STUDENT'S DECLARATION**

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

---

(Student's Signature)

Full Name : LOH ZI HAO  
ID Number : CB20175  
Date : 11 June 2023



AUTOMATED  
FERTIGATION  
MOBILE APPLICATION  
(MOBILE\_AFS)

LOH ZI HAO

Thesis submitted in fulfillment of the requirements  
for the award of the degree of  
Software Engineering

Faculty of Computing  
UNIVERSITI MALAYSIA PAHANG

JUNE 2023

## ACKNOWLEDGEMENTS

This project can be successfully developed, and I would like to thank my supervisor Sir Muhammad Zulfahmi Toh bin Abdullah @ Toh Chin Lai for his tireless support my degree final year project. Although I have many questions about the project, as long as he is capable, he will try his best to help me solve my troubles. He will guide my thesis writing with his rich experience, and then point out the problems in the thesis in a very polite tone.

In addition, I also thank all the lecturers and faculty members of the Faculty of Computing who have taught me in the past. There is a saying in ancient times that you should never forget your original intention. I always remember what they taught me. Without them, there would be no me today.

I would like to sincerely thank Dr. Mohd Azraai bin Mohd Razman, Senior Lecturer, Faculty of Technology, Manufacturing and Mechatronic Engineering (FTKPM), for his valuable guidance and support throughout the development of Mobile\_AFS. His expertise and dedication have been instrumental in shaping this project.

I would also like to thank Ms. Nurul Syafiqah binti Zaidi for her help and contribution to this project. Her valuable insights and commitment have greatly enhanced the quality of our work.

Additionally, I would like to thank the entire team for their collaborative efforts and use of the workspace provided by the iMAMS lab. Their collaboration and contributions have been critical to the successful development and testing of Mobile\_AFS.

Of course, I would also like to thank those friends who helped me when my project encountered a bottleneck. Their support is my motivation for perseverance.

I want to thank those who have questioned me and despised me. Your suspicion has become my motivation.

Finally, I would also like to thank my family, although they did not give me any help in the project. But they will always be my spiritual support, no matter what decision I make, they will support me. Thank you all.

## **ABSTRAK**

Mobile\_AFS ialah aplikasi mudah alih inovatif yang direka untuk membantu petani AGRONETICS merevolusikan amalan pertanian dengan menyediakan peladang penyelesaian yang komprehensif dan berkesan untuk menguruskan operasi mereka. Aplikasi ini menggantikan kaedah tulisan tangan tradisional dengan modul yang dipermudahkan untuk menangani cabaran utama yang dihadapi oleh petani. Daripada pengurusan tanaman dan penjadualan pengairan kepada penjejakan jualan dan pengurusan inventori, Mobile\_AFS membolehkan petani membuat keputusan berdasarkan data, mengoptimumkan peruntukan sumber dan meningkatkan kecekapan keseluruhan. Dengan menggunakan aplikasi ini, petani boleh meningkatkan hasil tanaman, mengurangkan sisa, memperkemas proses dan mencapai keuntungan yang lebih tinggi. Mobile\_AFS bertujuan untuk mengubah industri pertanian dengan menerima teknologi dan membolehkan petani berkembang maju dalam persekitaran yang kompetitif hari ini.

## **ABSTRACT**

Mobile\_AFS is an innovative mobile application designed to help farmers of AGRONETICS revolutionize agricultural practices by providing farmers with comprehensive and effective solutions to manage their operations. The app replaces traditional handwritten methods with simplified modules to address key challenges faced by farmers. From crop management and irrigation scheduling to sales tracking and inventory management, Mobile\_AFS enables farmers to make data-driven decisions, optimize resource allocation and improve overall efficiency. By using this application, farmers can increase crop yields, reduce waste, streamline processes, and achieve higher profitability. Mobile\_AFS aims to transform the agriculture industry by embracing technology and enabling farmers to thrive in today's competitive environment.

## TABLE OF CONTENT

<b>DECLARATION</b>	
<b>TITLE PAGE</b>	
<b>ACKNOWLEDGEMENTS</b>	<b>ii</b>
<b>ABSTRAK</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENT</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xvii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Aim and Objective	4
1.4 Scope	4
1.5 Significance of the project	5
1.6 Thesis Organization	7
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>8</b>
2.1 Overview of mobile application	8
2.1.1 Background of mobile application	9
2.2 Existing Systems/Works	10
2.2.1 Review of mobile application (Agrio – Precision agriculture)	10

2.2.2	Review of mobile application (ii.ri)	11
2.2.3	Review of mobile application (Smart Watering)	12
2.3	Analysis/ Comparison of Existing System	13
2.3.1	Strength & Weakness of Agrio	14
2.3.2	Strength & Weakness of ii.ri	15
2.3.3	Strength & Weakness of Smart Watering	16
2.4	Summary of comparison of system	17
2.4.1	Login	17
2.4.2	Manage profile.	17
2.4.3	Manage Crops.	17
2.4.4	Manage Timer	17
2.4.5	Manage Pesticide	18
2.4.6	Manage Sales	18
2.4.7	Manage Project	18
2.4.8	Manage Purchase	18
2.4.9	Manage Inventory	18
2.5	Overview of fertilizer used	19
2.6	Comparison of IOT device	20
2.6.1	Comparison on Internet Modules	20
2.6.2	Comparison on temperature sensors	22
2.6.3	Comparison on soil moisture sensors	23
<b>CHAPTER 3 METHODOLOGY</b>		<b>24</b>
3.1	Software Development Life Cycle (SDLC)	24
3.1.1	Reason choosing/ Advantages of Spiral Model	25
3.1.2	Disadvantages of Spiral Model	26

3.2	Project Requirement	27
3.2.1	Functional Requirement	27
3.2.2	Non-Functional Requirement	28
3.2.3	Constraints	28
3.2.4	Limitation	28
3.3	System requirement	29
3.4	Propose Design	31
3.4.1	Flowchart	31
3.4.2	Use Case Diagram	33
3.4.3	Activity Diagram	58
3.4.4	Context Diagram	67
3.4.5	Interface design/ Storyboard	68
3.5	Data Design	80
3.5.1	ERD	80
3.5.2	Data Dictionary	81
3.6	Testing/Validation Plan	92
3.6.1	Module Login	92
3.6.2	Module Profile	93
3.6.3	Manage Crops	94
3.6.4	Manage Timer	97
3.6.5	Manage Pesticide	99
3.6.6	Manage Sales	101
3.6.7	Manage Project	102
3.6.8	Manage Purchase	104
3.6.9	Manage Inventory	106
3.7	Potential Use of Proposed Solution	108

3.8	Gantt Chart	112
<b>CHAPTER 4 IMPLEMENTATION AND TESTING</b>		<b>113</b>
4.1	Introduction	113
4.2	Input/ Output Design Implementation	114
4.3	Database Implementation	116
4.3.1	Autotimer Table	116
4.3.2	Controltest Table	117
4.3.3	Crops Table	118
4.3.4	Farm Table	119
4.3.5	Farmers Table	120
4.3.6	Inventorys Table	121
4.3.7	Pesticide Table	122
4.3.8	Project Table	123
4.3.9	Purchase Table	124
4.3.10	Sales Table	125
4.3.11	Sales_history Table	126
4.4	User manual	127
4.4.1	Module Login	127
4.4.2	Main	129
4.4.3	Module Profile	130
4.4.4	Module Crops	131
4.4.5	Module Sensor	139
4.4.6	Module Timer	140
4.4.7	Module Pesticide	145
4.4.8	Module Sales	149



4.4.9	Module Schedule	152
4.4.10	Module Project	156
4.4.11	Module Purchase	161
4.4.12	Module Inventory	166
4.5	Coding implementation	172
4.5.1	Login	172
4.5.2	Forget Password (Sent Email)	174
4.5.3	Create Operation (Create Crops)	175
4.5.4	View Operation (View Crops)	178
4.5.5	Update Operation (Update Crops)	180
4.5.6	Delete Operation (Delete Crops)	183
4.5.7	Schedule Operation (Pesticide Schedule)	185
4.5.8	Validation operation	188
4.6	Testing of implementation	191
4.6.1	Proof of Testing	191
4.6.2	User Acceptance Test 1	192
4.6.3	User Acceptance Test 2	194
<b>CHAPTER 5 CONCLUSION</b>		<b>196</b>
5.1	Introduction	196
5.2	Recommendation	197
<b>REFERENCES</b>		<b>198</b>

## LIST OF TABLES

Table 2.1	Comparison between 3 existing system with project	13
Table 2.2	Strength & Weakness of Agrio	14
Table 2.3	Strength & Weakness of ii.ri	15
Table 2.4	Strength & Weakness of Smart Watering	16
Table 2.5	Comparison between 4 internet module Arduino	20
Table 2.6	Comparison between 4 Arduino temperature sensors	22
Table 2.7	Comparison between 4 Arduino soil moisture sensors	23
Table 3.1	Table of 4 phases of Spiral model	24
Table 3.2	Table of functional requirement	27
Table 3.3	Hardware and software used before development phase	29
Table 3.4	Hardware and software used in developing phase	29
Table 3.5	Hardware used after developing phase	30
Table 3.6	Table of Manage Login	34
Table 3.7	Table of Manage Profile	36
Table 3.8	Table of Manage Crops	38
Table 3.9	Table of Manage Timer	41
Table 3.10	Table of Manage Pesticide	44
Table 3.11	Table of Manage Sales	47
Table 3.12	Table of Manage Project	49
Table 3.13	Table of Manage Purchase	52
Table 3.14	Table of Manage Inventory	55
Table 3.15	Data Dictionary of AutotimerTable	81
Table 3.16	Data Dictionary of Controltest Table	82
Table 3.17	Data Dictionary of Crops Table	83
Table 3.18	Data Dictionary of Farm Table	85
Table 3.19	Data Dictionary of Farmers Table	86
Table 3.20	Data Dictionary of Inventorys Table	87
Table 3.21	Data Dictionary of Pesticide Table	88
Table 3.22	Data Dictionary of Project Table	89
Table 3.23	Data Dictionary of Purchase Table	90
Table 3.24	Data Dictionary of Sales Table	90
Table 3.25	Data Dictionary of Sales_history Table	91
Table 3.26	Test Case for Login	92

Table 3.27	Test Case for Forget Password	92
Table 3.28	Test Case for Profile Farmer	93
Table 3.29	Test Case for Add Crops	94
Table 3.30	Test Case for View Crops	94
Table 3.31	Test Case for Update Crops	95
Table 3.32	Test Case for Delete crops details	95
Table 3.33	Test Case for Edit Crops Status	96
Table 3.34	Test Case for Add Timer	97
Table 3.35	Test Case for View Timer	97
Table 3.36	Test Case for Update Timer	98
Table 3.37	Test Case for Delete Timer	98
Table 3.38	Test Case for Add Pesticide	99
Table 3.39	Test Case for View Pesticide	99
Table 3.40	Test Case for Update Pesticide	100
Table 3.41	Test Case for Delete Pesticide	100
Table 3.42	Test Case for Add Sales	101
Table 3.43	Test Case for View Sales	101
Table 3.44	Test Case for Add Project	102
Table 3.45	Test Case for View Project	102
Table 3.46	Test Case for Update Project	103
Table 3.47	Test Case for Delete Project	103
Table 3.48	Test Case for Add Purchase	104
Table 3.49	Test Case for View Purchase	104
Table 3.50	Test Case for Update Purchase	105
Table 3.51	Test Case for Delete Purchase	105
Table 3.52	Test Case for Add Inventory	106
Table 3.53	Test Case for View Inventory	106
Table 3.54	Test Case for Update Inventory	107
Table 3.55	Test Case for Delete Inventory	107

## LIST OF FIGURES

Figure 2.1	Main interface of Agrio application	10
Figure 2.2	Main interface of ii.ri application	11
Figure 2.3	Main interface of Smart Watering application	12
Figure 3.1	Basic Flowchart Mobile_AFS	31
Figure 3.2	Process Flowchart Mobile_AFS	32
Figure 3.3	Use case Diagram for Mobile_AFS	33
Figure 3.4	Use case Diagram for Manage Login	34
Figure 3.5	Use case Diagram for Manage Profile	36
Figure 3.6	Use case Diagram for Manage Crops	38
Figure 3.7	Use case Diagram for Manage Timer	41
Figure 3.8	Use case Diagram for Manage Pesticide	44
Figure 3.9	Use case Diagram for Manage Sales	47
Figure 3.10	Use case Diagram for Manage Project	49
Figure 3.11	Use case Diagram for Manage Purchase	52
Figure 3.12	Use case Diagram for Manage Inventory	55
Figure 3.13	Activity diagram for Manage Login	58
Figure 3.14	Activity diagram for Manage Profile	59
Figure 3.15	Activity diagram for Manage Crops	60
Figure 3.16	Activity diagram for Manage Timer	61
Figure 3.17	Activity diagram for Manage Pesticide	62
Figure 3.18	Activity diagram for Manage Sales	63
Figure 3.19	Activity diagram for Manage Project	64
Figure 3.20	Activity diagram for Manage Purchase	65
Figure 3.21	Activity diagram for Manage Inventory	66
Figure 3.22	Context Diagram for Mobile_AFS	67
Figure 3.23	Storyboard for Farmer Login and Forget Password	68
Figure 3.24	Storyboard for Main Interface	69
Figure 3.25	Storyboard for Manage Profile	70
Figure 3.26	Storyboard for Manage Crops	71
Figure 3.27	Storyboard for Manage Sensor	72
Figure 3.28	Storyboard for Manage Timer	73
Figure 3.29	Storyboard for Manage Pesticide	74
Figure 3.30	Storyboard for Manage Sales	75

Figure 3.31	Storyboard for Manage Schedule	76
Figure 3.32	Storyboard for Manage Project	77
Figure 3.33	Storyboard for Manage Purchase	78
Figure 3.34	Storyboard for Manage Inventory	79
Figure 3.35	ERD for Mobile_AFS	80
Figure 3.36	Figure for prototype farm	108
Figure 3.37	Figure for control panel and water tank	109
Figure 3.38	Figure for valve and piping	110
Figure 3.39	Gantt Chart 1	112
Figure 3.40	Gantt Chart 2	112
Figure 4.1	Input design screen of Mobile_AFS	114
Figure 4.2	Output design screen of Mobile_AFS	115
Figure 4.3	Database of Autotimer table	116
Figure 4.4	Database of Controltest table	117
Figure 4.5	Database of Crops table	118
Figure 4.6	Database of Farm table	119
Figure 4.7	Database of Farmers table	120
Figure 4.8	Database of Inventories table	121
Figure 4.9	Database of Pesticide table	122
Figure 4.10	Database of Project table	123
Figure 4.11	Database of Purchase table	124
Figure 4.12	Database of Sales table	125
Figure 4.13	Database of Sales_history table	126
Figure 4.14	Login User Manual of Mobile_AFS	127
Figure 4.15	Forget Password User Manual of Mobile_AFS	128
Figure 4.16	Main Interface User Manual of Mobile_AFS	129
Figure 4.17	Edit Profile User Manual of Mobile_AFS	130
Figure 4.18	Manage Crops Menu User Manual of Mobile_AFS	131
Figure 4.19	Add Crops User Manual of Mobile_AFS	132
Figure 4.20	Edit Crops User Manual of Mobile_AFS	133
Figure 4.21	View Crops User Manual of Mobile_AFS	134
Figure 4.22	Delete Crops User Manual of Mobile_AFS	135
Figure 4.23	Edit Crops Status Menu User Manual of Mobile_AFS	136
Figure 4.24	Edit Crops Status User Manual of Mobile_AFS	137
Figure 4.25	View Crops Status User Manual of Mobile_AFS	138

Figure 4.26	Manage Sensor User Manual of Mobile_AFS	139
Figure 4.27	Manage Timer Menu User Manual of Mobile_AFS	140
Figure 4.28	Add Timer 1 User Manual of Mobile_AFS	141
Figure 4.29	Add Timer 2 User Manual of Mobile_AFS	141
Figure 4.30	Add Timer 3 User Manual of Mobile_AFS	142
Figure 4.31	Add Timer 4 User Manual of Mobile_AFS	142
Figure 4.32	Edit Timer 1 User Manual of Mobile_AFS	143
Figure 4.33	Edit Timer 2 User Manual of Mobile_AFS	143
Figure 4.34	View Timer User Manual of Mobile_AFS	144
Figure 4.35	Manage Pesticide Menu User Manual of Mobile_AFS	145
Figure 4.36	Add Pesticide 1 User Manual of Mobile_AFS	146
Figure 4.37	Add Pesticide 2 User Manual of Mobile_AFS	146
Figure 4.38	Add Pesticide 3 User Manual of Mobile_AFS	147
Figure 4.39	Edit Pesticide User Manual of Mobile_AFS	147
Figure 4.40	View Pesticide Menu User Manual of Mobile_AFS	148
Figure 4.41	Manage Sales Menu User Manual of Mobile_AFS	149
Figure 4.42	Add Sales User Manual of Mobile_AFS	150
Figure 4.43	View Sales User Manual of Mobile_AFS	150
Figure 4.44	Edit Sales History User Manual of Mobile_AFS	151
Figure 4.45	View Sales Graph Manual of Mobile_AFS	151
Figure 4.46	Manage Schedule Menu User Manual of Mobile_AFS	152
Figure 4.47	View Water Schedule User Manual of Mobile_AFS	153
Figure 4.48	View Fertilizer Schedule User Manual of Mobile_AFS	154
Figure 4.49	View Pesticide Schedule User Manual of Mobile_AFS	155
Figure 4.50	Manage Project Menu User Manual of Mobile_AFS	156
Figure 4.51	Create Project 1 User Manual of Mobile_AFS	157
Figure 4.52	Create Project 2 User Manual of Mobile_AFS	158
Figure 4.53	Edit Project User Manual of Mobile_AFS	159
Figure 4.54	View Project User Manual of Mobile_AFS	160
Figure 4.55	Manage Purchase Menu User Manual of Mobile_AFS	161
Figure 4.56	Create Purchase 1 User Manual of Mobile_AFS	162
Figure 4.57	Create Project 2 User Manual of Mobile_AFS	163
Figure 4.58	Edit Purchase User Manual of Mobile_AFS	164
Figure 4.59	View Purchase User Manual of Mobile_AFS	165
Figure 4.60	Manage Sotck Status Menu User Manual of Mobile_AFS	166

Figure 4.61	Create Inventory 1 User Manual of Mobile_AFS	167
Figure 4.62	Create Inventory 2 User Manual of Mobile_AFS	168
Figure 4.63	Edit Inventory User Manual of Mobile_AFS	169
Figure 4.64	View Inventory User Manual of Mobile_AFS	170
Figure 4.65	View Total Stock User Manual of Mobile_AFS	171
Figure 4.66	Login code 1	172
Figure 4.67	Login code 2	172
Figure 4.68	Login code 3	173
Figure 4.69	Forget Password code 1	174
Figure 4.70	Forget Password code 2	174
Figure 4.71	Create crops Android Studio code 1	175
Figure 4.72	Create crops Android Studio code 2	176
Figure 4.73	Create crops Visual Studio code	177
Figure 4.74	View crops Android Studio code 1	178
Figure 4.75	View crops Android Studio code 2	178
Figure 4.76	View crops Visual Studio code	179
Figure 4.77	Update crops Android Studio code 1	180
Figure 4.78	Update crops Android Studio code 2	181
Figure 4.79	Update crops Visual Studio code	182
Figure 4.80	Delete crops Android Studio code 1	183
Figure 4.81	Delete crops Android Studio code 2	184
Figure 4.82	Delete crops Visual Studio code	184
Figure 4.83	Schedule Pesticide Android Studio code – Retrieve from db	185
Figure 4.84	Schedule Pesticide Android Studio code – Display selected day only	185
Figure 4.85	Schedule Pesticide Android Studio code – Sort timer	186
Figure 4.86	Schedule Pesticide Android Studio code – Sort timer 1	186
Figure 4.87	Schedule Pesticide Android Studio code – Sort timer 2	187
Figure 4.88	Schedule Pesticide Android Studio code – Swipe Action	187
Figure 4.89	Empty Validation	188
Figure 4.90	Email Validation	188
Figure 4.91	Password Validation	189
Figure 4.92	Decimal Validation	189
Figure 4.93	Date Validation	190
Figure 4.94	Proof of Testing with Farmer	191
Figure 4.95	User Acceptance Test 1 (Part 1)	192

Figure 4.96	User Acceptance Test 1 (Part 2)	193
Figure 4.97	User Acceptance Test 2 (Part 1)	194
Figure 4.98	User Acceptance Test 2 (Part 2)	195



## **LIST OF ABBREVIATIONS**

CR	Change requests
CF-N	Chemical Fertilizer Nitrogen
UAT	User Acceptance Test
SDLC	Software Development Life Cycle
BYOD	Bring Your Own Device

## CHAPTER 1

### INTRODUCTION

#### 1.1 Introduction

According to Department of Statistics Malaysia Official Portal, agricultural sector in our country contributed 7.4% of the country's GDP in 2020. Sector growth shrank by 2.2% from 2.0% the year before. The negative growth of 3.6% particularly in oil palm caused the fall of commodity sub-sector (Department of Statistics Malaysia Official Portal, 2021).

Certain measures must be put forth to address the ongoing fall in the GDP contribution of agriculture in Malaysia. Application of liquid nutrients through an irrigation system is known as fertigation. The principal grower-controlled input for plant growth, nutrients and water, may be precisely regulated using micro-irrigation and fertilisation. Water-soluble nutrients are less likely to be lost due to excessive rainfall or over-irrigation, thus fertiliser is used sparingly when necessary (Boman & Obreza, 2008).

During the fertigation process, the nutrient solution is injected into the irrigation water by the suitable injection device. With the presence of the fertigation system, the soil moisture, production level, and sunlight intensity may all be visualized. According to the requirements of the crop, nutrients can be given during the growth season, reducing leaching losses and increasing water usage effectiveness. Fertigation is safer since it reduces the possibility of root injury from greater doses. When fertigation with a fertilizing system, less or even not required the mechanical activity. Small amounts of fertiliser can be supplied to swiftly remedy any shortages. Highly mobile nutrients, like nitrogen, should also be carefully regulated to guarantee faster crop uptake. (Ranjan, Shivani & Sow, Sumit., 2021)

## 1.2 Problem Statement

The main problem of this study was inefficient inventory management practices lead to farmer always frustrated when they are looking for items in the storage. Farmers face huge challenges due to lack of inventory visibility, which not only wastes valuable time looking for items, but can also lead to product spoilage and financial loss. In addition, incomplete inventory management systems make these problems more complicated by preventing farmers from effectively tracking quantities and prices associated with their stocks and sales receipts. As a result, farmers do not have a comprehensive view of their inventory levels and financial transactions, blocking their ability to make better decisions and optimize operations (Tranquil, 2023).

Next, farmers are currently facing challenges in effectively managing budgets. Water bills, fertilizer, pesticide, and supplement purchases, resulting in excessive spending. This is largely due to a lack of budgetary control tools and techniques, uncertainty about costs, and limited visibility into spending. For example, fluctuations in the fertilizers make it difficult to accurately estimate and plan expenditures, leading to budget overruns. To solve this problem, Mobile\_AFS provides a mobile application that enables farmers to track and plan their planting and fertilizing activities budget. The farmer now can create the Project which allocate the duration and budget for the specific project to run. In that project, the farmer can insert the Purchase details which including the price and receipt for the items they buy. Therefore, the farmer can easily track the expense for the project in their farm.

Lastly, the waste of water and fertilizer also become one the problem facing by the farmer. From the oral interview from the customer, we knew that the water and fertilizer often waste when the gardener watering the plant. This is because the water will spread around and some of the water will not absord by the plant will causing the waste of water and fertilizer. Nevertheless, increased fertiliser use during irrigation frequently leads to pollution because it raises the nutrient levels in both groundwater and surface water. Nitrate levels in the Thames rose by around 8 mg over a forty-six-year period beginning in 1938 (Hagin, J., & Lowengart, A., 1996). Regardless of the fertigation method, drip irrigation 0.5\* Epan enhanced fruit yield by 51.7% and reduced irrigation water use by 48.1% as compared to the advised greenhouse practise. Water consumption is more effective with drip irrigation compared to surface watering because the rate of water loss

via evaporation from the soil surface is significantly lower (Mahajan, G., & Singh, K. G. , 2006). The result of the study is a plan to minimise water and fertilizer waste so customers can save more money for other infrastructure upgrades. At the end of the project, we want the farmer to produce an output that achieves the optimal combination of water and fertilizer for the different crops on the farm.

### **1.3 Aim and Objective**

The aim of this project is to develop a mobile application to assist the customer to monitor and visualize the growth of plants and revenue. In order to comply this, the objectives of the project are the following:

- i. To investigate the GUI and database for Mobile\_AFS
- ii. To develop a mobile development system for Mobile\_AFS
- iii. To evaluate the effectiveness of the system through the User Acceptance Test (UAT)

### **1.4 Scope**

- i. This mobile application only serves the farmer that under AGRONETICS
- ii. The mobile application will utilize web hosting PHPMyAdmin for data storage and management.
- iii. This system will be designed under Android Studio

## **1.5 Significance of the project**

As coming to the end of this project, the key benefit of this mobile application is it will help our clients, particularly farmers who maintain their crops. This is because most of the companies is encouraging the concept of BYOD, smartphone can be easily used in the home, at work and anywhere. The farmer can easily control the spray of the fertilizers and water just by using their fingertips. This mobile software also enables farmers to substantially reduce the amount of fertiliser waste that is sprayed onto the ground and then washed away by rain.

First, the app's "Manage Crops" module allows farmers to maintain detailed records and track the status of their crops. This includes important information such as crop details, growth stage and health. By having a centralized platform for monitoring crop growth, farmers can detect problems early, take targeted interventions, and improve overall crop yield and quality.

Second, the Manage Timer and Manage Pesticide modules provide farmers with precise control over irrigation and pesticide application schedules. Farmers can ensure that crops receive the necessary water and pesticide treatments at the correct time intervals, minimizing waste and enhancing crop health. This level of precision and efficiency can improve resource management, reduce costs, and ultimately increase profitability.

Additionally, the "Manage Sales" module enables farmers to record and analyse their sales data, including profit margins and market prices. This valuable information enables farmers to make data-driven pricing and marketing decisions, optimize their revenue streams, and build stronger relationships with buyers.

The Manage Project, Manage Purchases and Manage Inventory modules further enhance the app's prominence by providing farmers with the tools to efficiently manage projects, track purchases and monitor stock levels. By streamlining these important aspects of farming operations, the app improves workflow efficiency, reduces human errors, and helps farmers maintain optimal inventory levels for uninterrupted production.

Overall, the Mobile\_AFS app represents a major advance in agricultural technology, providing farmers with a powerful tool to enhance crop management, increase

productivity, optimize resource use and drive sustainable growth in an evolving agricultural landscape.

## **1.6 Thesis Organization**

As a summary for every chapter in this thesis, it contains five chapters which Chapter 1 is Introduction of the project, Chapter 2 is literature review, Chapter 3 is Methodology, Chapter 4 is implementation and Testing, and Chapter 5 is Conclusion.

The first chapter will briefly describe the case study of the project. The problems faced by our target users and their hope that we will develop a mobile application to solve their problems will be documented in this chapter. In addition, the scope of the project will also be defined in this chapter.

Chapter 2 will explain the overview and background of mobile applications. In addition, we will analyze our mobile application against existing mobile applications in the market.

Chapter 3 will analyze methods for developing mobile applications. We will also list system requirements and user requirements. This chapter will detail the proposed, database and interface design.

Chapter 4 will explain the input and output implementation, database implementation, user manual, and User Acceptance Test (UAT).

Chapter 5 will give conclusions and recommendations to develop better mobile applications.



## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Overview of mobile application**

The existing mobile application that marketed on the largest android download platform, Google Play Store will be used for comparison with our mobile application. Throughout the chapter, we will elaborate on the reasons for choosing good specifications as an advantage on the previous project and propose new ideas for improving my project into a better mobile application. At the end of this chapter, I will list the characteristics of my project as a conclusion.

### **2.1.1 Background of mobile application**

In Malaysia, the use of mobile phones is crucial to getting farmers to work in agriculture. They can receive weather forecasts on their mobile devices to apply agricultural inputs such as pesticides and fertilizers. When disaster approaches the farm, the crops will be destroyed. Now, this mobile apps can alert farmers to natural disasters. Most farmers in Malaysia have difficulty getting in touch with agricultural professionals due to lack of communication. This community still uses traditional tactics like posters and voice amplifiers. Information obtained in this way may be different than expected. These incidents show that poor communication is a major cause of problems in farming communities. (Chhachhar & Md Salleh Hassan, 2012)

A significant advantage of using mobile apps is the elimination of human error. Using the handwriting method, farmers may forget important details such as when to apply fertilizer, how much to apply, or where to store stock. These errors can lead to suboptimal crop growth, wasted resources and financial losses. Mobile apps solve this problem by providing a systematic and organized way to record and manage information. Farmers can set view for important tasks at the schedule or in app calendar such as water, fertilizer, and fertilization scheduling. This helps them avoid forgetting key activities and ensures that operations are performed in the right way at the right time.

When customer is finding for a solution, the purpose of developing this mobile application can be reflected here. The mobile application that will come out in this study will have easy-to-understand interface so that users logging in for the first time will also understand all the buttons present on my mobile application. In addition, the database will be implemented to store data also that the information can be viewed by the farmer and admin at any time. For functionally, the mobile application can automatically mix the water and fertilizer and control the timing to fertigate the correct amount of solution to the crops after set by the farmer.

## 2.2 Existing Systems/Works

### 2.2.1 Review of mobile application (Agrio – Precision agriculture)

Agrio is published by the Sailog Ltd company. Farmer and crop advisors can predict, identify, and treat plant diseases, pests and nutritional deficiencies with the help of the precision plant protection system Agrio. Agrio helps farmers, crop advisors and agronomists with crop management, plant disease identification, plant diagnostics and yield improvement by applying its patented artificial intelligence and computer vision algorithms. The combined and enhanced knowledge of many agricultural professionals from around the world can be found in Digital Plant Doctor. We offer multi-layered strategies to maintain plant health, with an emphasis on prediction and prevention.

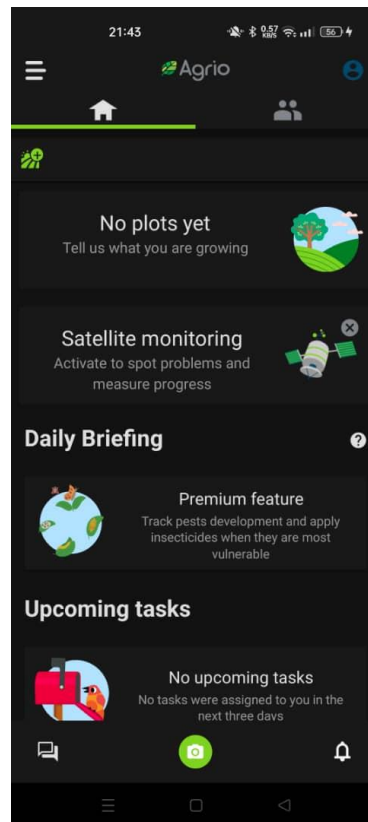


Figure 2.1 Main interface of Agrio application

### 2.2.2 Review of mobile application (ii.ri)

ii.ri is published by the Baccara company. ii.ri is the first irrigation controller that allows anyone to manage and control their own computerized garden irrigation system through a simple smartphone app. It's a smart controller that simplifies everything with a simple but effective device. ii.ri is compatible with all taps and hoses currently on the market. It has no screen or keyboard and is completely wireless.

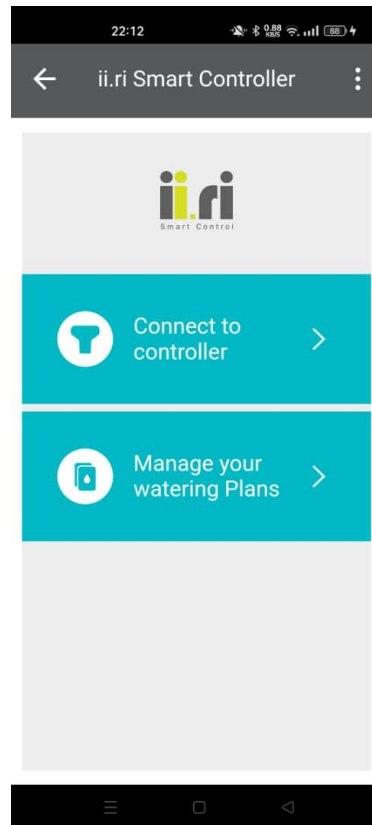


Figure 2.2 Main interface of ii.ri application

### 2.2.3 Review of mobile application (Smart Watering)

Smart Watering is published by the Smart Watering company. Smart Watering apps can remotely control irrigation systems. It is suitable for farmers who do not live near the fields they cultivate, and for all farmers who want to make the irrigation process easier for themselves. With the help of the app, an unlimited number of irrigation programs can be set. It provides insight into soil moisture status and field weather forecasts, so irrigation can be optimized by application, resulting in significant savings in irrigation time, money, and water.

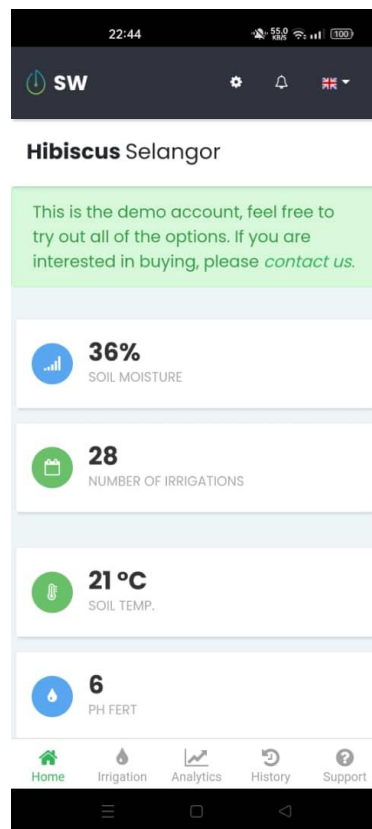


Figure 2.3 Main interface of Smart Watering application

### 2.3 Analysis/ Comparison of Existing System

There are many similarities and differences in functionality between currently available farm irrigation mobile applications available in Malaysia. Table 2.1 shows the similarity and differences between Agrio, ii.ri, Smart Watering mobile application and Mobile\_AFS which is my project.

Table 2.1 Comparison between 3 existing system with project

<b>Comparison</b>				
<b>Feature</b>	<b>Agrio</b>	<b>ii.ri</b>	<b>Smart Watering</b>	<b>MOBILE_AFS</b>
<b>Scheduled irrigation</b>	Not Available	Available	Available	Available
<b>Map/ Satellite</b>	Available	Not available	Available	Not available
<b>Forum</b>	Available	Not available	Not available	Not available
<b>Graph</b>	Not available	Not available	Available	Available
<b>Advertisement</b>	Yes	No	No	No
<b>Pesticides</b>	Available	Not available	Not available	Not available
<b>Fertilizer</b>	Not available	Not available	Available	Available
<b>Sales</b>	Not available	Not available	Not available	Available
<b>Project</b>	Not available	Not available	Not available	Available
<b>Purchase</b>	Not available	Not available	Not available	Available
<b>Inventory</b>	Not available	Not available	Not available	Available

Source: (Baccara, 2018), (Saillog Ltd, 2022), (Smart Watering, 2021)

However, the system will also have the similarity which are: -

- All three mobile application can view the status of the sensors.
- All can maintain different categories of farm crops.

Detailed explanation will be explained below in the strength and weakness of all three existing system.

### 2.3.1 Strength & Weakness of Agrio

Table 2.2 Strength & Weakness of Agrio

<b>Strength of Agrio</b>	
Map/ Satellite	Farmer able to subscribe to watch the plant growth progress and receive alerts when problems are discovered.
Forum	The farmer will have the forum or society to post the feed, question, and thoughts.
Enquiry	Premium user can get experts opinions by sending unlimited number of inquiries and answers will be reply by less than an hour.
Pesticides	Track pests' development and apply insecticides when they are most vulnerable.
<b>Weakness of Agrio</b>	
Advertisement	The application contains annoying advertisement that need user to pay to unlock ad-free service.
Scheduler/ Timer	The application does not have timer that will help farmer to water the plant when on time.
Graph	The application does not have graph to help user to visually understand the development of crops.

### 2.3.2 Strength & Weakness of ii.ri

Table 2.3 Strength & Weakness of ii.ri

<b>Strength of ii.ri</b>	
Scheduler/ Timer	The application does not have timer that will help farmer to water the plant when on time.
Advertisement	The application does not contain annoying advertisement.
<b>Weakness of ii.ri</b>	
Forum	The application does not contain forum that help farmer share the feed, question, thoughts.
Graph	The application does not have graph to help user to visually understand the development of crops.
Pesticides	The application does not have function to apply insecticides to kill the insects.



### 2.3.3 Strength & Weakness of Smart Watering

Table 2.4 Strength & Weakness of Smart Watering

<b>Strength of Smart Watering</b>	
Scheduler/ Timer	The application does not have timer that will help farmer to water the plant when on time.
Map/ Satellite	Farmer able to subscribe to watch the plant growth progress and receive alerts when problems are discovered.
Graph	The application does not have graph to help user to visually understand the development of crops.
Advertisement	The application does not contain annoying advertisement.
<b>Weakness of Smart Watering</b>	
Forum	The application does not contain forum that help farmer share the feed, question, thoughts.
Pesticides	The application does not have function to apply insecticides to kill the insects.

## **2.4 Summary of comparison of system**

From my observation from these 3 current existing mobile applications, the mobile application that I develop will have the following characteristics:

### **2.4.1 Login**

Farmers can log in to the mobile application through the login page of the mobile application. Farmer also have the option to reset their password when they forget about the password.

### **2.4.2 Manage profile.**

This module allows farmers to update their own user profiles by edit the username, phone number and address.

### **2.4.3 Manage Crops.**

This module allows farmers to efficiently manage their crops by providing comprehensive details and tracking their growth, health, and stages.

### **2.4.4 Manage Timer**

The Manage Timer module assists farmers in record the scheduling irrigation and fertilization activities for each crop in every day.

#### **2.4.5 Manage Pesticide**

The Manage Pesticide module assists farmers in record the scheduling pesticide activities for each crop in a week.

#### **2.4.6 Manage Sales**

The Manage Sales module enables farmers to track and record sales transactions, including profit calculations and market prices.

#### **2.4.7 Manage Project**

This module helps farmers manage project details such as duration and budget.

#### **2.4.8 Manage Purchase**

The Manage Purchase module enables farmers to record and manage their purchasing activities for each project.

#### **2.4.9 Manage Inventory**

This module allows farmers to keep track of their inventory, including incoming stock, total quantity, and date of storing for each purchase.

## 2.5 Overview of fertilizer used

According to many scientists, there are 16 components that are essential for the growth and development of higher green plants (Roberts, 2007). The elements that make up proteins and the living matter of all living organisms are Carbon (C), Hydrogen (H), Oxygen (O), Nitrogen (N), Phosphorus (P) and Sulphur (S). However, different crops in different fields have different requirements for key nutrients. Due to the different soil types and nutrient ranges of different crops, some of them require nitrogen, phosphorus, and potassium.

A 10-year period regression analysis showed that there was a significant positive correlation between China's annual grain output and the consumption of chemical fertilizer nitrogen (CF-N) from 1949 to 1998. However, the excessive use of nitrogen poses problems for the environment. Some of the major damages are the discharge of gaseous nitrogen and increased nitrogen concentrations in groundwater and lake water bodies. (Zhu & Chen, 2002) Most of the unused nitrogen will end up in lakes, rivers and discharged as pollutants. Excess nitrogen left in the air can have severe effects on human respiration, restrict visibility and alter plant growth.

Based on the article, CF30 treatment (30% of chemical nitrogen fertilizer with 70% of cow manure compost) were even better than those with pure chemical fertilizer. In this study, it was clearly documented that cow manure compost could be used as a potential nutrient source to enhance plant growth, enhance tomato yield, and improve soil fertility in clay loam structures. Cow manure or compost mixed with cow manure has great commercial value. Therefore, we will suggest out client for mixing the use of chemical nitrogen fertilizer and cow manure compost. (Hasnain et al., 2020)

## 2.6 Comparison of IOT device

The IoT device we are going to use is Arduino. There are many Arduino devices, but we will only compare certain categories related to our Mobile\_AFS. The categories that will be compared are Internet Modules, Soil Moisture and Temperature.

### 2.6.1 Comparison on Internet Modules

The internet modules that we going to compare are ESP32 NodeMCU, ESP8266 NodeMCU V2, SIM7600 4G GSM, and DFRobot SIM7000C Arduino Expansion Shield.

Table 2.5 Comparison between 4 internet module Arduino

	ESP32	ESP8266	SIM7600	SIM7000C
Operating Voltage	3.3V	3.3V	5.0V	5.0V
Port	USB	USB	USB + Type C	USB 2.0
Internet Type	Wifi	Wifi	Cellular	Cellular
Bluetooth	No	Yes	No	No
Temperature Sensor	No	Yes	No	Yes
Humidity sensor	No	No	No	Yes
Navigation	No	No	No	Yes
Onboard charger	No	No	Yes	No
Audio support	No	No	Yes	No
Price (approx.)	RM27	RM48	RM75	RM193

Source: (DIYIOT, 2021), (How To Electronics, 2022), (Mouser Electronics, Inc., 2018)

The Arduino module we recommend is SIM7600 4G GSM. This is because it uses cellular data as the network type. This allows farms that are not covered by Wi-Fi to use IoT devices and connect to the network without problems. Next, the Arduino device is equipped with a onboard charger, allowing farmers to continue using the device in the event of sudden power outage power outage. Finally, it has audio support, making sounds when something happens to the crop.

The disadvantage is that SIM7600 4G GSM has no temperature and humidity sensor. However, this may be compensated as we will be integrating separate temperature and humidity sensors.

## 2.6.2 Comparison on temperature sensors

The temperature sensors that we going to compare are DHT11, DHT22 (AM2302), LM35, and DS18B20.

Table 2.6 Comparison between 4 Arduino temperature sensors

	DHT11	DHT22	LM35	DS18B20
Humidity Sensors	Yes	Yes	No	No
Communication protocol	One-wire	One-wire	Analog	One-wire
Voltage	Up to 5.5V	Up to 6V	Up to 30V	Up to 5.5V
Temperature Range	0 to 50°C	-40 to 80°C	-55 to 150°C	-55 to 125°C
Accuracy	2 °C	0.5 °C	0.5 °C	0.5 °C
Price (approx.)	RM30	RM48	RM5	RM44

Source: (RandomNerdTutorials, 2019)

Due to the large price difference, we will recommend the cheapest LM35. Compared with other products, it can support the most voltage and the largest temperature range. It also has very good accuracy with only 0.5 °C error margin.

### 2.6.3 Comparison on soil moisture sensors

The soil moisture sensors that we going to compare are Grove – Capacitive Moisture Sensor, and Grove – Moisture Sensor (Resistive Option).

Table 2.7 Comparison between 4 Arduino soil moisture sensors

	Capacitive	Resistive
Suitable	Every type of environment	Home Gardening
Corrosion Free	Yes	No
Accuracy	High	Low
Price (approx.)	RM29	RM15

Source: (Shawn, 2019)

Due to the soil moisture sensor will be used in the farm which will receive various kind of liquid including the liquid fertilizer and water which will corrode the soil moisture sensor. Therefore, the best choice for us is resistive type soil moisture sensor. Besides, it also has the higher accuracy compared to the capacitive type of soil moisture sensor.



## CHAPTER 3

### METHODOLOGY

#### 3.1 Software Development Life Cycle (SDLC)

As we go through the software development life cycle (SDLC) process, we expect to produce the highest quality software within the budget and time allocated. Each model has advantages and disadvantages. To ensure project success, it may be necessary to select an appropriate SDLC model based on the specific concerns and requirements of the project. The chosen SDLC model for the Mobile\_AFS is Spiral Model. The spiral model has four different phases includes planning, risk analysis, engineering, and evaluation. (Gurung et al., 2020)

Table 3.1 Table of 4 phases of Spiral model

Planning	During this phase, the client's needs are gathered, and objectives are identified, articulated, and analysed. Cost, schedule, and resources will also be estimated based on the client's specifications. For example, requirements will be documented as SRS and SDD, as will project timelines, Gantt charts, and cost estimates for customers.
Risk analysis	After identifying, articulating and analyzing the objectives, stakeholders will propose the best possible solutions in the second quadrant. Risks hidden in the solution will be identified and countermeasures will be prepared to overcome them.

	Prototypes for optimal solutions are built at the end of this quadrant.
Engineering	At this stage, the functionality required by the customer is developed and then integrated into the system. Software tester will test the system to make sure there are no problems. Testing, coding, and deploying software at customer sites are the examples of activities.
Evaluation	The customer will evaluate the software. User Acceptance Testing is the most important way to gather satisfaction (UAT). After all the phases is ended, the project manager will start thinking about the next spiral.

### 3.1.1 Reason choosing/ Advantages of Spiral Model

- Good for large projects
- Ability to deal with risks: Some SDLCs, such as Waterfall, cannot deal with risks and must go back to the previous stage when something goes wrong. However, spiral model captures risk and deals with it in each spiral.
- Flexibility in requirements: Change requests (CR) from stakeholders such as customers or project managers can be easily accommodated and incorporated.
- Customer Satisfaction: The spiral model is useful for collecting customer feedback. By showing prototypes to customers, they can view and evaluate their products at this stage. This allows them to voice their dissatisfaction or make changes before the software is released. As a result, this saves development teams time and money.

### **3.1.2 Disadvantages of Spiral Model**

- Complexity: The spiral model is one of the most complex models compared to other SDLC models. All protocols must be followed to ensure that the stages are completed flawlessly and on time. The model requires a lot of documentation because it contains many spirals and stages.
- Risk of not meeting deadlines or budget: Since the project starts with an unknown number of phases, it is difficult to estimate time.

## 3.2 Project Requirement

The project requirement consists of the functional requirement, non-functional requirement, constraints, and limitations.

### 3.2.1 Functional Requirement

Table 3.2 Table of functional requirement

Manage Login	In this module, the farmers can log in to the mobile application through the login page of the mobile application
Manage Profile	In this module, the farmer have the authority to view and update their own profile.
Manage Crops	In this module, the farmer is able to create, view, update and delete details of crops.
Manage Timer	In this module, farmers can record the scheduling irrigation and fertilization activities for each crop in every day.
Manage Pesticide	In this module, farmers create, view, update and delete the record the scheduling pesticide activities.
Manage Sales	In this module, farmers can create and view the sales details.
Manage Project	This module helps farmers create, view, update and delete project details.
Manage Purchase	In this module, farmers can create, view, update and delete the purchase details.
Manage Inventory	In this module, farmers can create, view, update and delete the inventory details.

### **3.2.2 Non-Functional Requirement**

- The mobile application supports 1000 concurrent users.
- The mobile application shall be available at anytime and anywhere.
- The mobile application should be able to run on all Android mobile devices.

### **3.2.3 Constraints**

- The system should have at least one administrator to approve farmers' registrations.
- Farmer must have internet connection to use the mobile application.
- Farmers need to have an account to log in before using the system.
- Farmer must register under Teraju.

### **3.2.4 Limitation**

- Time frame to complete the mobile application is 10 months.
- The supported OS for the mobile application is Android only.
- All API used must be free due to no budget allocated.

### 3.3 System requirement

The user requirement consists of the hardware and software requirements. There are three phases in order develop the Mobile\_AFS, which are the before development, in developing and after development phase. Different hardware and software will be used in different phases.

Table 3.3 Hardware and software used before development phase

<b>HARDWARE</b>	<b>SPECIFICATION</b>	<b>IMPORTANCE</b>	<b>FUNCTION</b>
Laptop	Operating system: Window 10 Processor: Intel i5 RAM: 8GB	Act as medium for running the software.	Install software to design the GUI and write the documentation.
<b>SOFTWARE</b>	<b>VERSION</b>	<b>IMPORTANCE</b>	<b>FUNCTION</b>
Figma	9.0 (Latest Version)	To design the GUI and prototype for Mobile_AFS	To design the GUI and prototype for Mobile_AFS
Microsoft Word	2304 (Latest Version)	To open the thesis templete.	To record the documentation of thesis

Table 3.4 Hardware and software used in developing phase

<b>HARDWARE</b>	<b>SPECIFICATION</b>	<b>IMPORTANCE</b>	<b>FUNCTION</b>
Laptop	Operating system: Window 10 Processor: Intel i5 RAM: 8GB	Act as medium for Android Studio and phpMyAdmin	Install software or IDE to code the program and database for Mobile_AFS
<b>SOFTWARE</b>	<b>VERSION</b>	<b>IMPORTANCE</b>	<b>FUNCTION</b>
Android Studio	Version: Electric Eel	Fast coding with the intelligent code editor for auto complete coding	To code the mobile application and extract into apk file
Xampp	Version: 3.3.0	Act as a medium to connect to phpMyAdmin	Run the php files in localhost setup
Visual Studio Code	Version: 1.78 (Latest version)	To run the php files	Act as the IDE for code PHP files for the backend for Mobile_AFS

Table 3.5 Hardware used after developing phase

<b>HARDWARE</b>	<b>SPECIFICATION</b>	<b>IMPORTANCE</b>	<b>FUNCTION</b>
Smartphone	Operating system: Android Version: 6.0 (Marshmallow) and above	Make sure the user can run under the same devices	To download and run the project mobile application (apk)
Sim 800 GSM	Connectivity: GSM SIM Card Support: Yes (Sim 800)	To establish the connection via GSM using mobile data	To upload the Arduino sensor data to database

### 3.4 Propose Design

#### 3.4.1 Flowchart

The figure below shows the flowchart of Mobile\_AFS. This is basically showing how the user input their data to the system and store to database then receive output from the system.

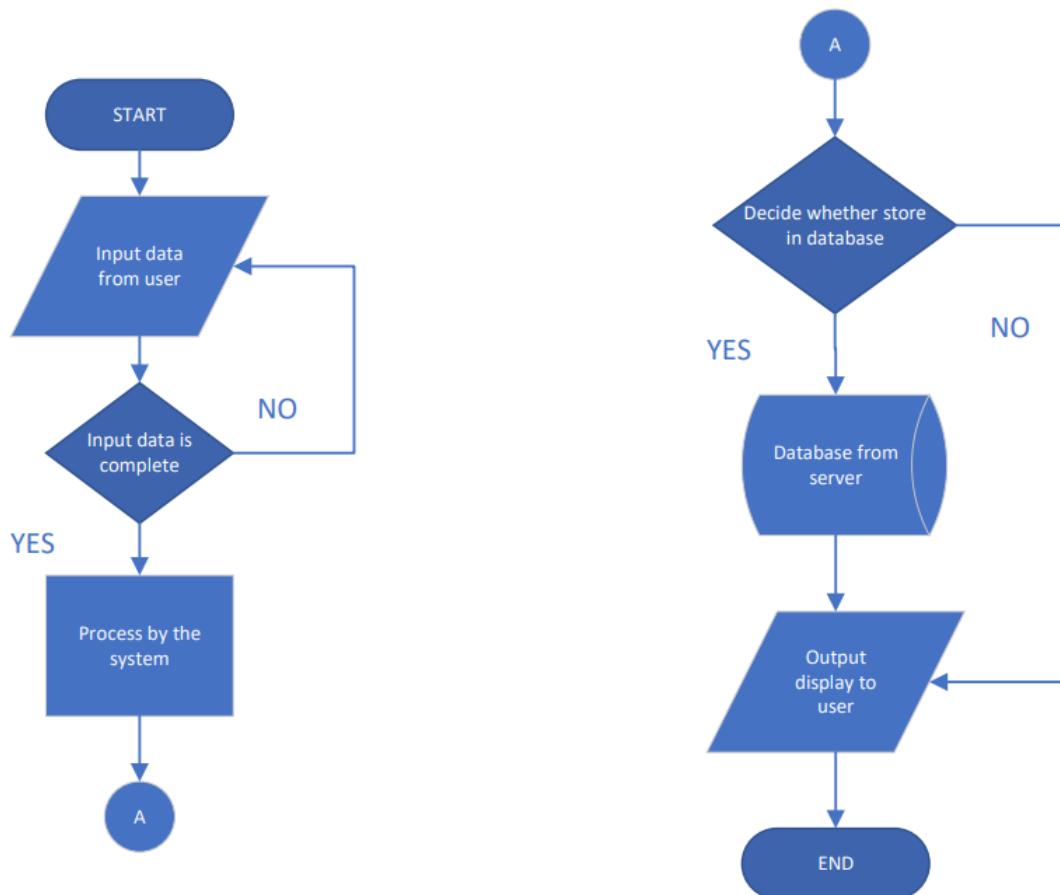


Figure 3.1 Basic Flowchart Mobile\_AFS



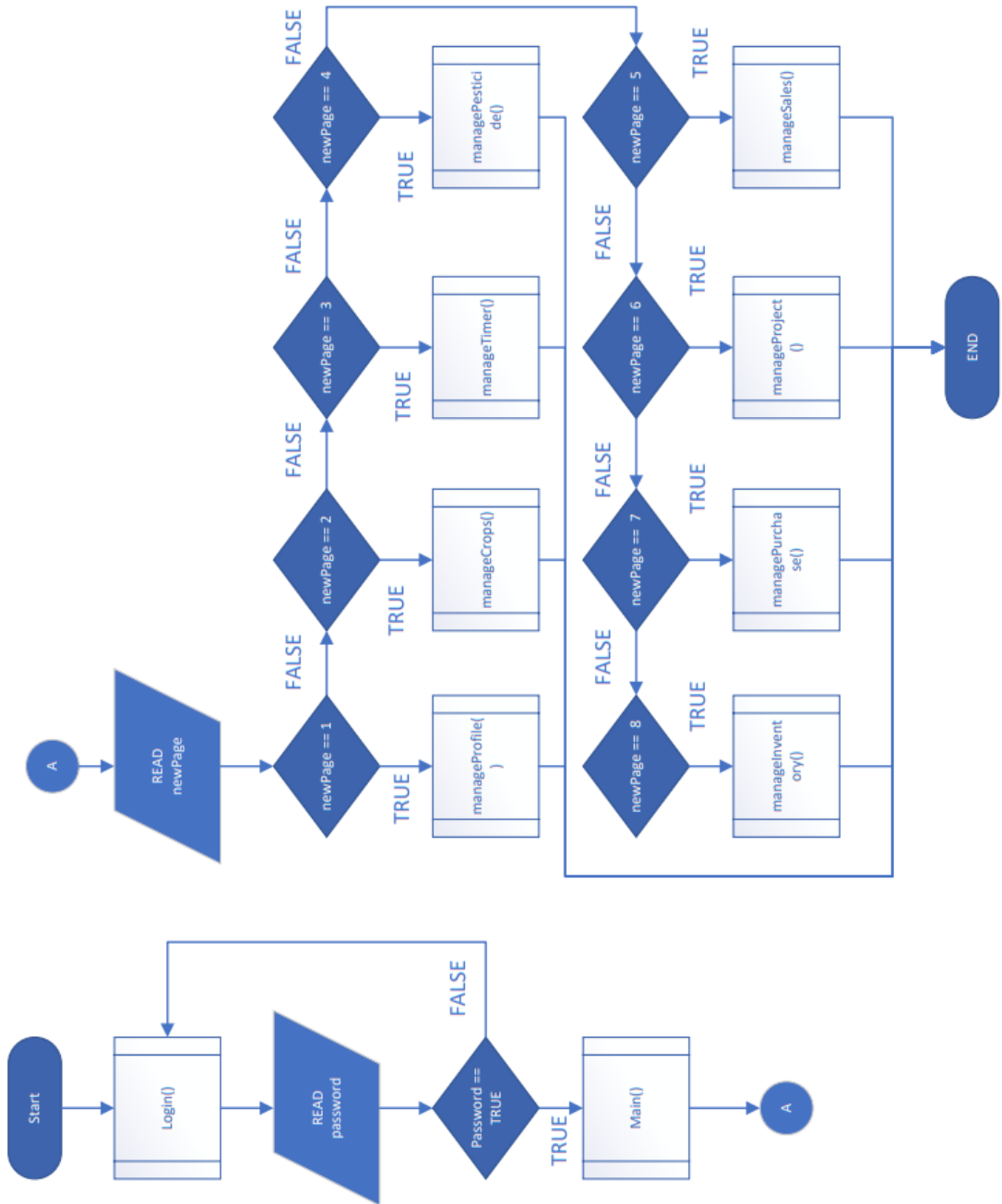


Figure 3.2 Process Flowchart Mobile\_AFS

### 3.4.2 Use Case Diagram

There are several functions that carried out the Mobile\_AFS. Figure below shows the interaction between farmers.

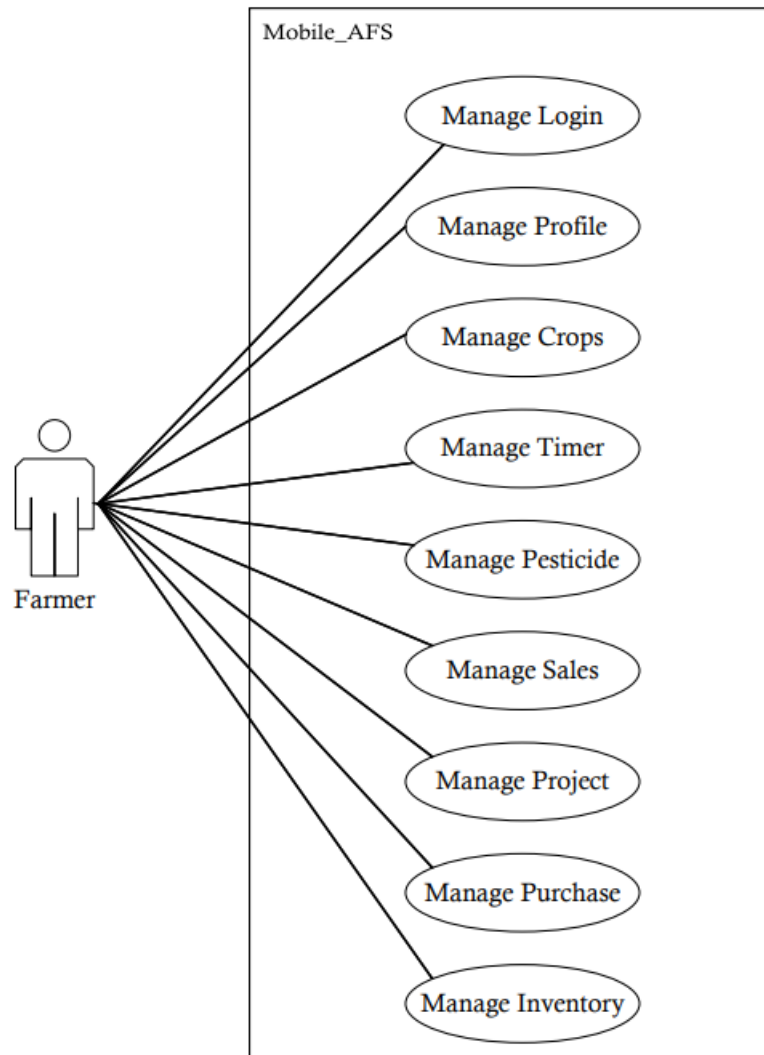


Figure 3.3 Use case Diagram for Mobile\_AFS

According to Figure 3.3, Mobile\_AFS have one actor which is farmer. Farmer have the access to the login, profile, manage crops, manage timer, manage pesticide, manage sales, manage project, manage purchase and manage inventory.

### 3.4.2.1 Manage Login

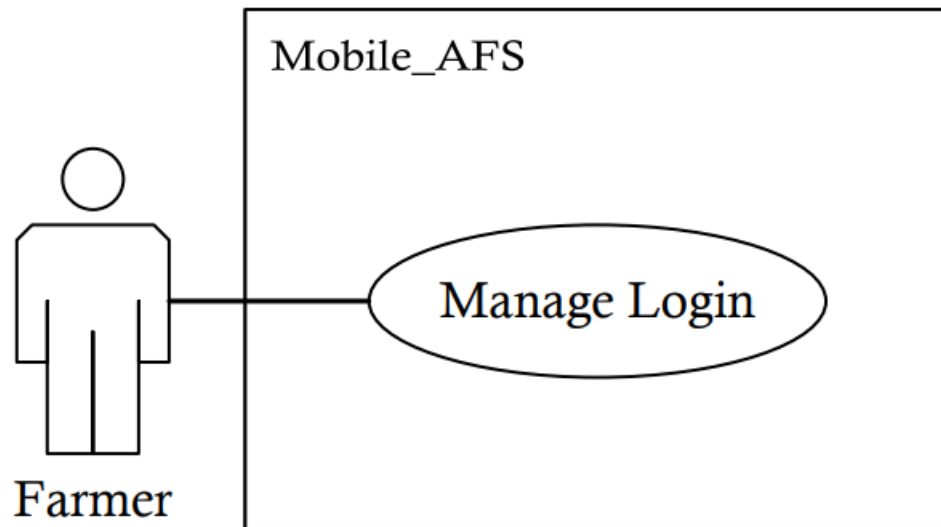


Figure 3.4 Use case Diagram for Manage Login

Table 3.6 Table of Manage Login

Use Case ID	AFMS-UC01
Brief Description	Farmer and admin can login to the system.
Actor	Farmer
Pre-Condition	Farmer is registered to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the users open the system.</li> <li>2. The system shows the Login page.</li> <li>3. The user can do the following option:               <ul style="list-style-type: none"> <li><b>[A1] Login</b></li> <li><b>[A2] Forget Password</b></li> </ul> </li> <li>4. The use case ends.</li> </ol>
Alternative Flow	<p><b>[A1] Login [AFMS-UC01-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user enters the username and password.</li> <li>2. The user clicks &lt;&lt;Login&gt;&gt; button.</li> <li>3. The system retrieves and verify the details. <b>[E1] Invalid Login Data</b></li> </ol>

	<ol style="list-style-type: none"> <li>4. The user login to the system.</li> <li>5. The use case continues step 3 in basic flow.</li> </ol> <p><b>[A2] Forget Password [AFMS-UC01-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Forget Password&gt;&gt; button.</li> <li>2. The system display Forget Password page.</li> <li>3. The user enters the email address.</li> <li>4. The system validates the email address.</li> </ol> <p><b>[E2] Invalid Email address</b></p> <ol style="list-style-type: none"> <li>5. The system sends reset password verification email.</li> <li>6. The use case continues step 3 in basic flow.</li> </ol>
Exception Flow	<p><b>[E1] Invalid Login Data [AFMS -UC01-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid login data.</li> <li>2. The system displays invalid login data error message.</li> <li>3. The use case continues step 5 in alternative flow [A1] of User.</li> </ol> <p><b>[E2] Invalid Email address [AFMS -UC01-E02]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid email address.</li> <li>2. The system displays invalid login data error message.</li> <li>3. The use case continues step 4 in alternative flow [A2] of User.</li> </ol>
Post-Condition	User able to login to the system
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.2 Manage Profile

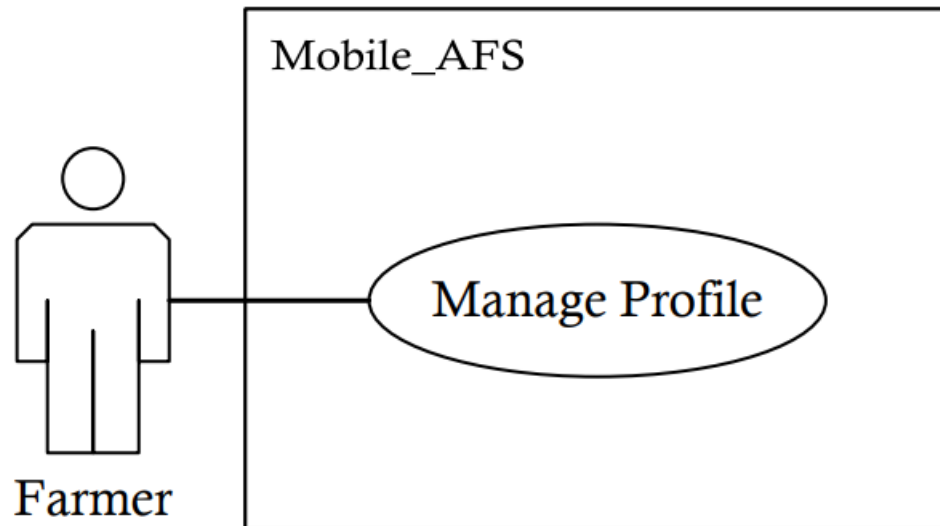


Figure 3.5 Use case Diagram for Manage Profile

Table 3.7 Table of Manage Profile

Use Case ID	AFMS-UC02
Brief Description	Farmer can edit their user profile to the system.
Actor	Farmer
Pre-Condition	Farmer already login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Profile&gt;&gt; button.</li> <li>2. The system display Edit profile page.</li> <li>3. The user updates the user details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data. <b>[E1] Invalid Input Data</b></li> <li>6. The system updates the user details to the database.</li> <li>7. The use case ends.</li> </ol>

Alternative Flow	None
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC02-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid profile data.</li> <li>2. The system displays invalid profile data error message.</li> <li>3. The use case continues step 6 in basic flow [A1].</li> </ol>
Post-Condition	User able to edit user profile to the system
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.3 Manage Crops

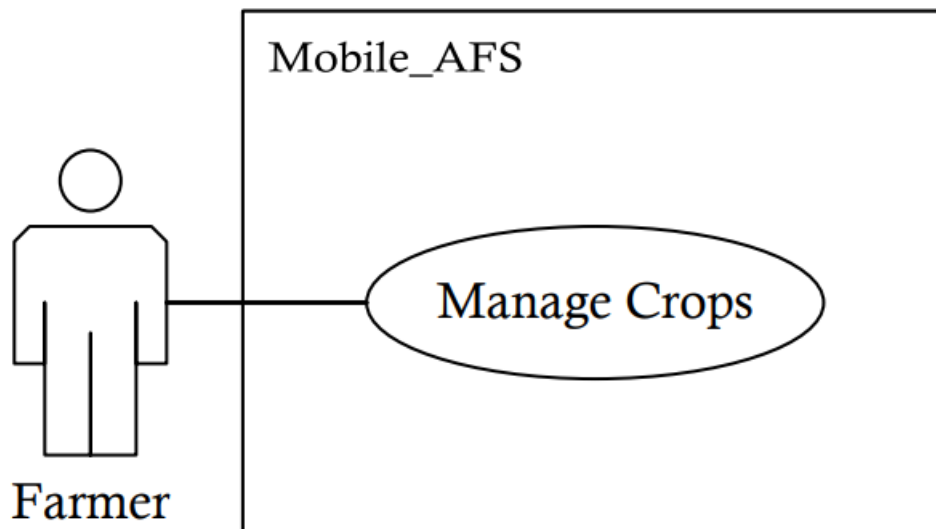


Figure 3.6 Use case Diagram for Manage Crops

Table 3.8 Table of Manage Crops

Use Case ID	AFMS-UC03
Brief Description	Farmer can add, view, edit and delete the crops to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<p>Farmer</p> <ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Manage crops&gt;&gt; button.</li> <li>2. The system shows the Manage crops page.</li> <li>3. The user can do the following option: <ul style="list-style-type: none"> <li>[A1] Add new crops</li> <li>[A2] View crops</li> <li>[A3] Edit crops</li> <li>[A4] Delete crops</li> </ul> </li> <li>4. The use case ends.</li> </ol>

Alternative Flow	<p><b>[A1] Add new crops [AFMS-UC03-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Crops page.</li> <li>3. The user enters the crops details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Crops records.</li> <li>6. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A2] View Crops [AFMS-UC03-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Crops&gt;&gt; button.</li> <li>2. The system direct user to the view crops page.</li> <li>3. The system retrieves and display the crops details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Crops [AFMS-UC03-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit crops&gt;&gt; button.</li> <li>2. The system display Edit crops page.</li> <li>3. The user updates the user details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the user details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Crops [AFMS- UC03-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Crops&gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the crops deletion.</li> <li>4. The system deletes the crops details from the database.</li> <li>5. The use case continues step 4 in basic flow</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC03-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid crops data.</li> <li>2. The system displays invalid crops data error message.</li> </ol>



	3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3] of User.
Post-Condition	The latest crops details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.4 Manage Timer

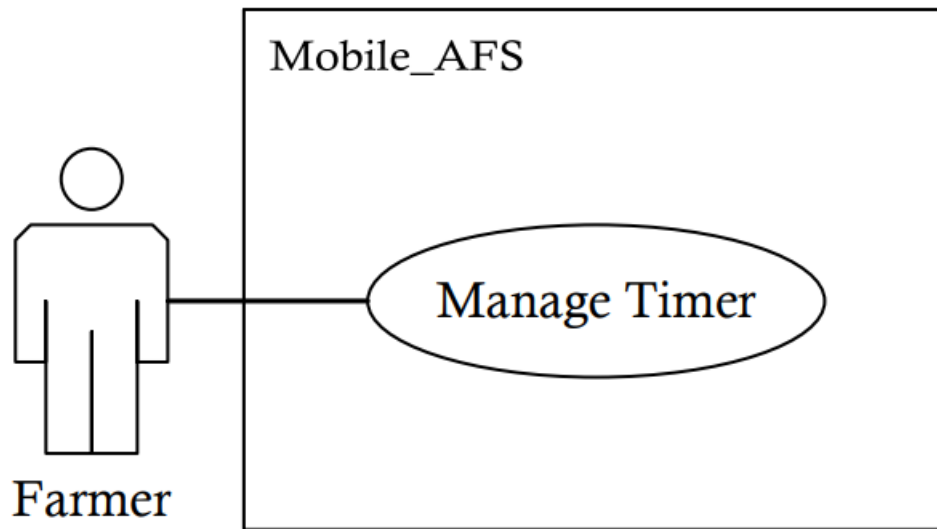


Figure 3.7 Use case Diagram for Manage Timer

Table 3.9 Table of Manage Timer

Use Case ID	AFMS-UC04
Brief Description	Farmer can add, view, edit and delete the timer to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Timer&gt;&gt; button.</li> <li>2. The system shows the Manage timer page.</li> <li>3. The user can do the following option:               <ul style="list-style-type: none"> <li>[A1] Add Timer</li> <li>[A2] View Timer</li> <li>[A3] Edit Timer</li> <li>[A4] Delete Timer</li> </ul> </li> <li>4. The use case ends.</li> </ol>

Alternative Flow	<p><b>[A1] Add Timer [AFMS-UC04-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Timer page.</li> <li>3. The user enters the timer details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Timer records.</li> <li>6. The use case continues step 4 in basic flow</li> </ol> <p><b>[A2] View Timer [AFMS-UC04-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Timer&gt;&gt; button.</li> <li>2. The system direct user to the view timer page.</li> <li>3. The system retrieves and display the timer details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Timer [AFMS-UC04-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit timer&gt;&gt; button.</li> <li>2. The system display Edit timer page.</li> <li>3. The user updates the timer details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the timer details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Timer [AFMS-UC04-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Timer&gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the timer deletion.</li> <li>4. The system deletes the timer details from the database.</li> <li>5. The use case continues step 4 in basic flow</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC04-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid timer data.</li> <li>2. The system displays invalid timer data error message.</li> </ol>

	3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3] of User.
Post-Condition	The latest timer details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.5 Manage Pesticide

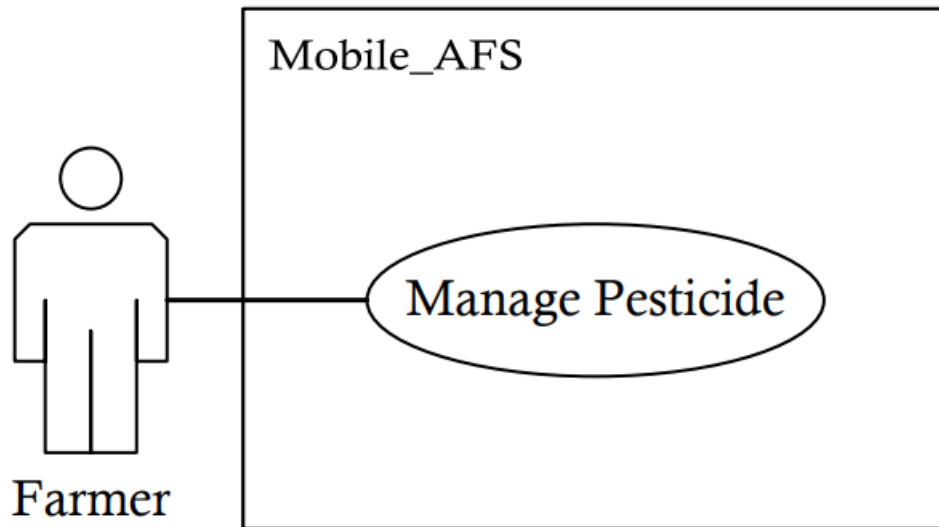


Figure 3.8 Use case Diagram for Manage Pesticide

Table 3.10 Table of Manage Pesticide

Use Case ID	AFMS-UC05
Brief Description	Farmer can add, view, edit and delete the pesticide details to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Pesticide&gt;&gt; button.</li> <li>2. The system shows the Manage pesticide page.</li> <li>3. The user can do the following option: <ul style="list-style-type: none"> <li>[A1] Add Pesticide</li> <li>[A2] View Pesticide</li> <li>[A3] Edit Pesticide</li> <li>[A4] Delete Pesticide</li> </ul> </li> <li>4. The use case ends.</li> </ol>
Alternative Flow	[A1] Add Pesticide [AFMS-UC05-A01]

	<ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Pesticide page.</li> <li>3. The user enters the pesticide details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Pesticide records.</li> <li>6. The use case continues step 4 in basic flow</li> </ol> <p><b>[A2] View Pesticide [AFMS-UC05-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Pesticide&gt;&gt; button.</li> <li>2. The system direct user to the view pesticide page.</li> <li>3. The system retrieves and display the pesticide details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Pesticide [AFMS-UC05-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit Pesticide &gt;&gt; button.</li> <li>2. The system display Edit pesticide page.</li> <li>3. The user updates the pesticide details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the pesticide details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Pesticide [AFMS-UC05-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Pesticide &gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the pesticide deletion.</li> <li>4. The system deletes the pesticide details from the database.</li> <li>5. The use case continues step 4 in basic flow.</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC05-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid pesticide data.</li> <li>2. The system displays invalid pesticide data error message.</li> <li>3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3].</li> </ol>

Post-Condition	The latest pesticide details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.6 Manage Sales

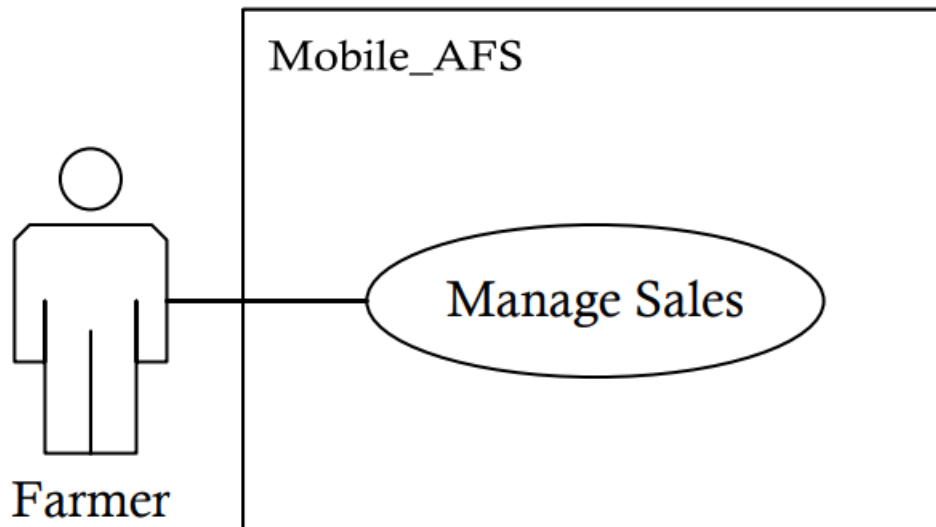


Figure 3.9 Use case Diagram for Manage Sales

Table 3.11 Table of Manage Sales

Use Case ID	AFMS-UC06
Brief Description	Farmer can add and view the sales details in the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Sales&gt;&gt; button.</li> <li>2. The system shows the Manage Sales page.</li> <li>3. The user can do the following option:           <ul style="list-style-type: none"> <li>[A1] Add Sales</li> <li>[A2] View Sales</li> </ul> </li> <li>4. The use case ends.</li> </ol>
Alternative Flow	<p>[A1] Add Sales [AFMS-UC06-A01]</p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Sales page.</li> <li>3. The user enters the sales details.</li> </ol>



	<p>4. The system validates the required field data.  <b>[E1] Invalid Input Data</b></p> <p>5. The system saves the data to Sales records.  6. The use case continues step 4 in basic flow.</p> <p><b>[A2] View Sales [AFMS-UC06-A02]</b></p> <p>1. The user clicks &lt;&lt; View Sales &gt;&gt; button.  2. The system direct user to the view sales page.  3. The system retrieves and display the sales details.  4. The use case continues step 4 in basic flow.</p>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC06-E01]</b></p> <p>1. The system detects invalid sales data.  2. The system displays invalid sales data error message.  3. The use case continues step 5 in alternative flow [A1]</p>
Post-Condition	The latest sales details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.7 Manage Project

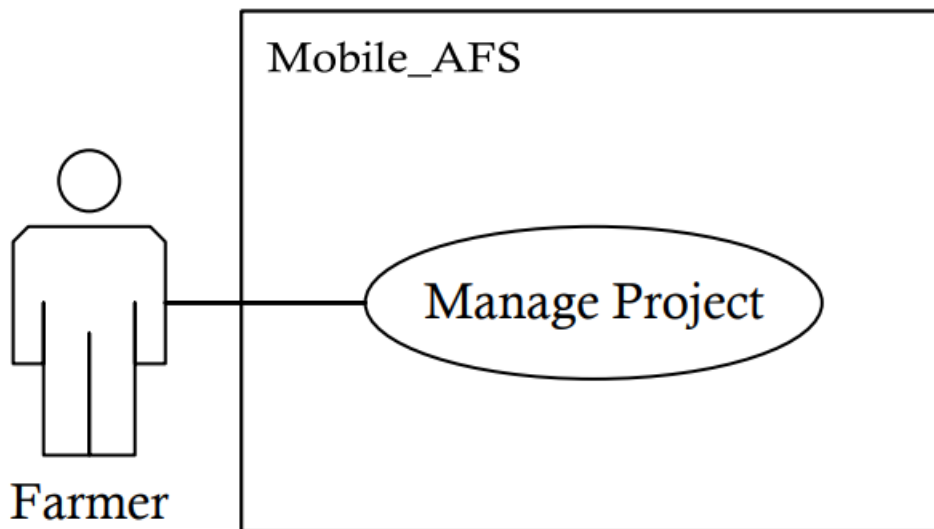


Figure 3.10 Use case Diagram for Manage Project

Table 3.12 Table of Manage Project

Use Case ID	AFMS-UC07
Brief Description	Farmer can add, view, edit and delete the project details to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt;Project&gt;&gt; button.</li> <li>2. The system shows the Manage Project page.</li> <li>3. The user can do the following option:           <ul style="list-style-type: none"> <li><b>[A1] Add Project</b></li> <li><b>[A2] View Project</b></li> <li><b>[A3] Edit Project</b></li> <li><b>[A4] Delete Project</b></li> </ul> </li> <li>4. The use case ends.</li> </ol>

Alternative Flow	<p><b>[A1] Add Project [AFMS-UC07-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Project page.</li> <li>3. The user enters the project details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Project records.</li> <li>6. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A2] View Project [AFMS-UC07-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Project&gt;&gt; button.</li> <li>2. The system direct user to the view project page.</li> <li>3. The system retrieves and display the project details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Project [AFMS-UC07-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit Project&gt;&gt; button.</li> <li>2. The system display Edit Project page.</li> <li>3. The user updates the project details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the project details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Project [AFMS-UC07-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Project&gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the project deletion.</li> <li>4. The system deletes the project from the database.</li> <li>5. The use case continues step 4 in basic flow.</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC07-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid project data.</li> <li>2. The system displays invalid project data error message.</li> </ol>

	3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3].
Post-Condition	The latest project details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.8 Manage Purchase

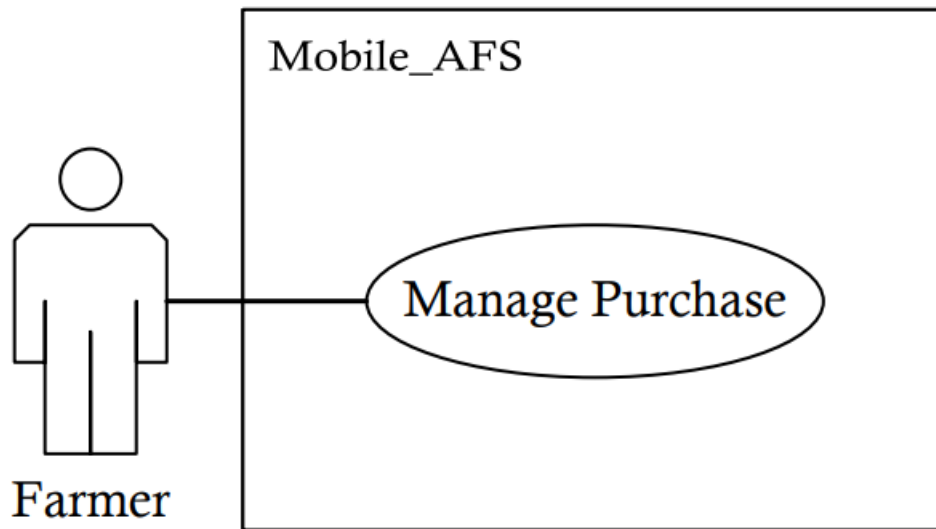


Figure 3.11 Use case Diagram for Manage Purchase

Table 3.13 Table of Manage Purchase

Use Case ID	AFMS-UC08
Brief Description	Farmer can add, view, edit and delete the purchase details to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt; Purchase&gt;&gt; button.</li> <li>2. The system shows the Manage Purchase page.</li> <li>3. The user can do the following option:           <ul style="list-style-type: none"> <li><b>[A1] Add Purchase</b></li> <li><b>[A2] View Purchase</b></li> <li><b>[A3] Edit Purchase</b></li> <li><b>[A4] Delete Purchase</b></li> </ul> </li> <li>4. The use case ends.</li> </ol>

Alternative Flow	<p><b>[A1] Add Purchase [AFMS-UC08-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Purchase page.</li> <li>3. The user enters the purchase details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Purchase records.</li> <li>6. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A2] View Purchase [AFMS-UC08-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Purchase&gt;&gt; button.</li> <li>2. The system direct user to the view purchase page.</li> <li>3. The system retrieves and display the purchase details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Purchase [AFMS-UC08-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit Purchase&gt;&gt; button.</li> <li>2. The system display Edit Purchase page.</li> <li>3. The user updates the purchase details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the purchase details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Purchase [AFMS-UC08-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Project&gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the purchase deletion.</li> <li>4. The system deletes the purchase from the database.</li> <li>5. The use case continues step 4 in basic flow.</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC07-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid purchase data.</li> </ol>

	<ol style="list-style-type: none"> <li>2. The system displays invalid purchase data error message.</li> <li>3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3].</li> </ol>
Post-Condition	The latest purchase details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.2.9 Manage Inventory

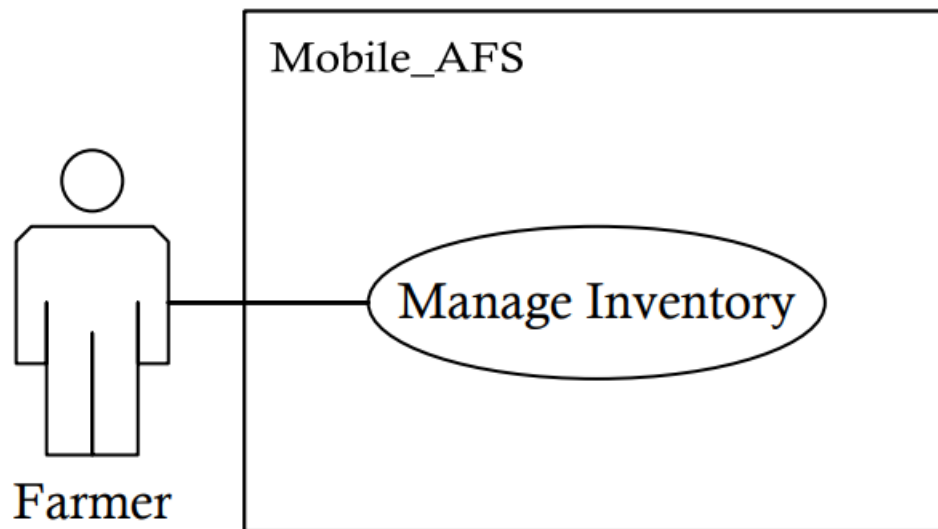


Figure 3.12 Use case Diagram for Manage Inventory

Table 3.14 Table of Manage Inventory

Use Case ID	AFMS-UC09
Brief Description	Farmer can add, view, edit and delete the inventory details to the system.
Actor	Farmer
Pre-Condition	Farmer is login to the system.
Basic Flow	<ol style="list-style-type: none"> <li>1. The use case begins when the user clicks &lt;&lt; Inventory&gt;&gt; button.</li> <li>2. The system shows the Manage Inventory page.</li> <li>3. The user can do the following option:           <ul style="list-style-type: none"> <li><b>[A1] Add Inventory</b></li> <li><b>[A2] View Inventory</b></li> <li><b>[A3] Edit Inventory</b></li> <li><b>[A4] Delete Inventory</b></li> </ul> </li> <li>4. The use case ends.</li> </ol>



Alternative Flow	<p><b>[A1] Add Inventory [AFMS-UC09-A01]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Create new&gt;&gt; button.</li> <li>2. The system direct user to the Create Inventorypage.</li> <li>3. The user enters the inventory details.</li> <li>4. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>5. The system saves the data to Inventory records.</li> <li>6. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A2] View Inventory [AFMS-UC09-A02]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; View Inventory&gt;&gt; button.</li> <li>2. The system direct user to the view inventory page.</li> <li>3. The system retrieves and display the inventory details.</li> <li>4. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A3] Edit Inventory [AFMS-UC09-A03]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt; Edit Inventory&gt;&gt; button.</li> <li>2. The system display Edit Inventory page.</li> <li>3. The user updates the inventory details.</li> <li>4. The user clicks &lt;&lt;Update&gt;&gt; button</li> <li>5. The system validates the required field data.</li> </ol> <p style="text-align: center;"><b>[E1] Invalid Input Data</b></p> <ol style="list-style-type: none"> <li>6. The system updates the inventory details to the database.</li> <li>7. The use case continues step 4 in basic flow.</li> </ol> <p><b>[A4] Delete Inventory [AFMS-UC09-A04]</b></p> <ol style="list-style-type: none"> <li>1. The user clicks &lt;&lt;Delete Inventory&gt;&gt; button.</li> <li>2. The system display deletes confirmation message box for the user.</li> <li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm the inventory deletion.</li> <li>4. The system deletes the inventory from the database.</li> <li>5. The use case continues step 4 in basic flow.</li> </ol>
Exception Flow	<p><b>[E1] Invalid Input Data [AFMS -UC07-E01]</b></p> <ol style="list-style-type: none"> <li>1. The system detects invalid inventory data.</li> </ol>

	<ol style="list-style-type: none"> <li>2. The system displays invalid inventory data error message.</li> <li>3. The use case continues step 5 in alternative flow [A1] and step 6 in alternative flow [A3].</li> </ol>
Post-Condition	The latest inventory details are updated to the database.
Rules	No applicable.
Constraints	No applicable.

### 3.4.3 Activity Diagram

#### 3.4.3.1 Manage Login

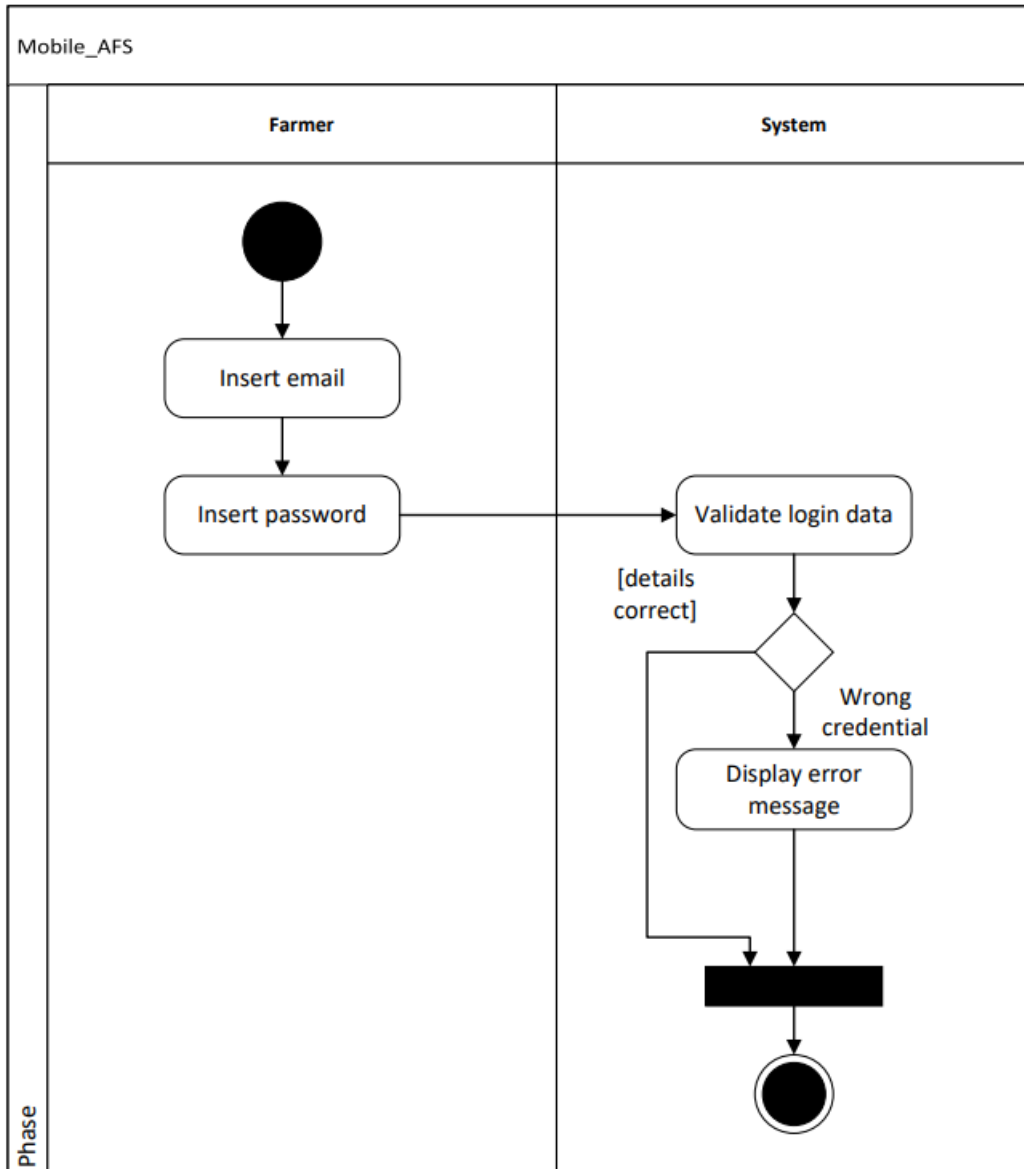


Figure 3.13 Activity diagram for Manage Login

### 3.4.3.2 Manage Profile

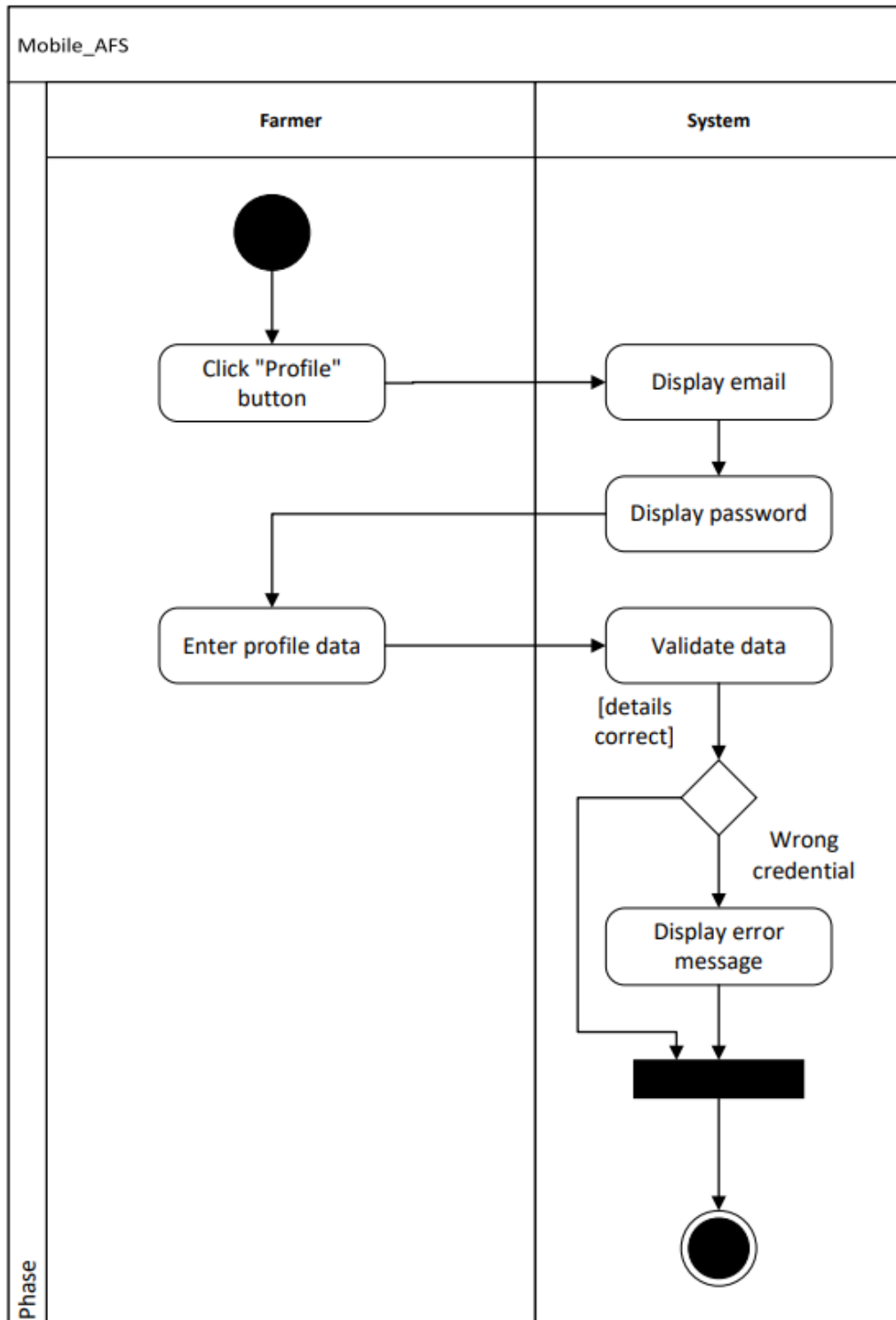


Figure 3.14 Activity diagram for Manage Profile

### 3.4.3.3 Manage Crops

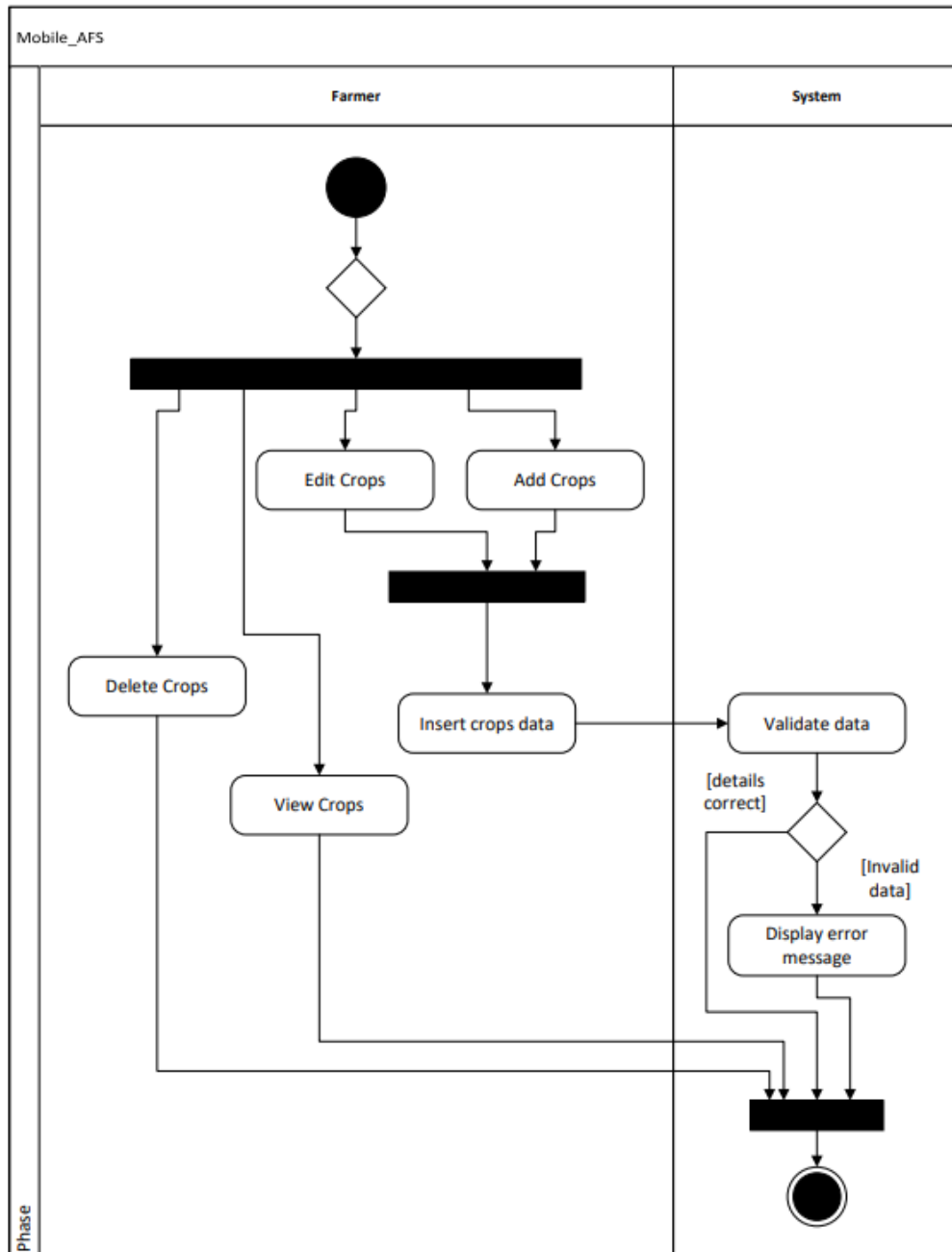


Figure 3.15 Activity diagram for Manage Crops

### 3.4.3.4 Manage Timer

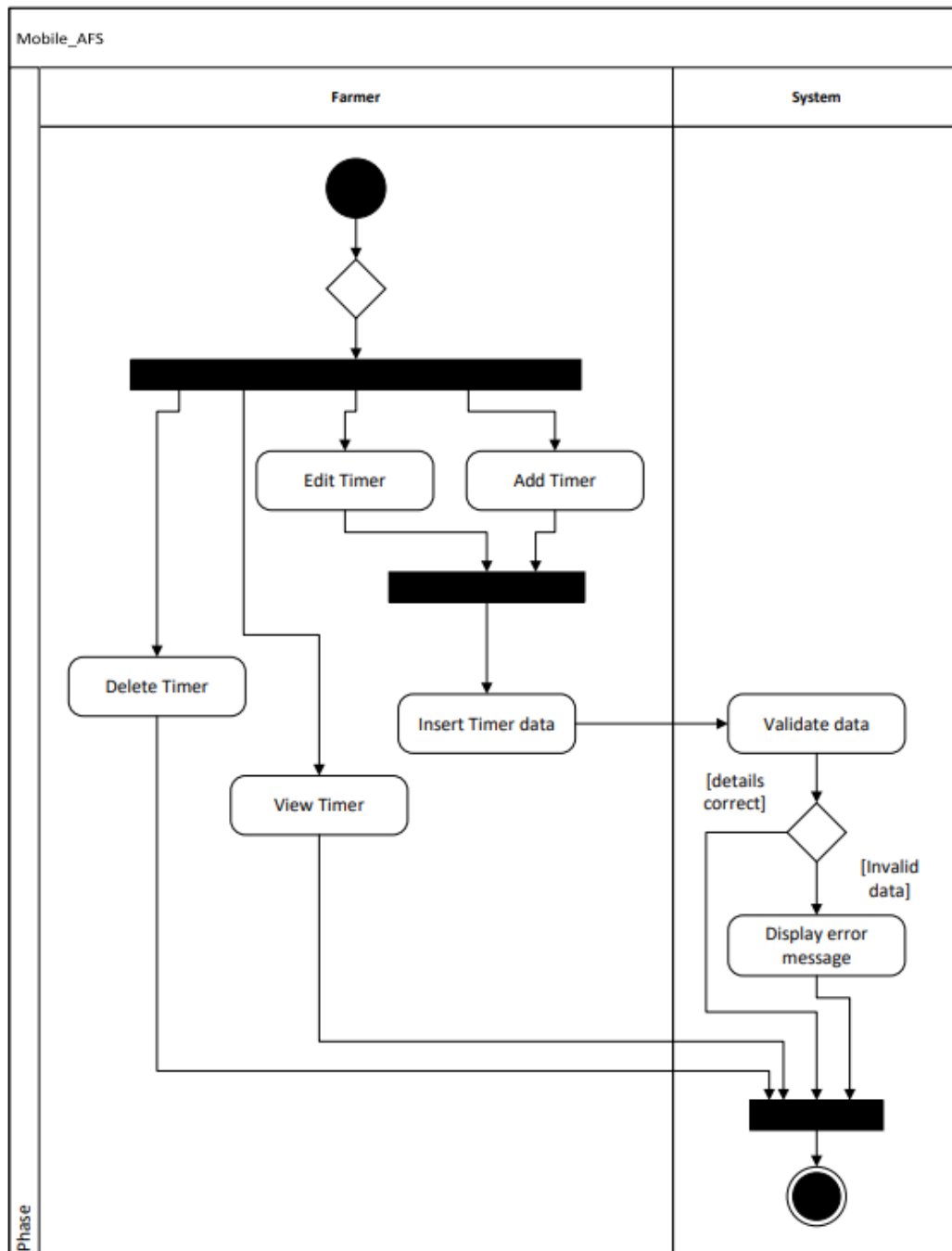


Figure 3.16 Activity diagram for Manage Timer

### 3.4.3.5 Manage Pesticide

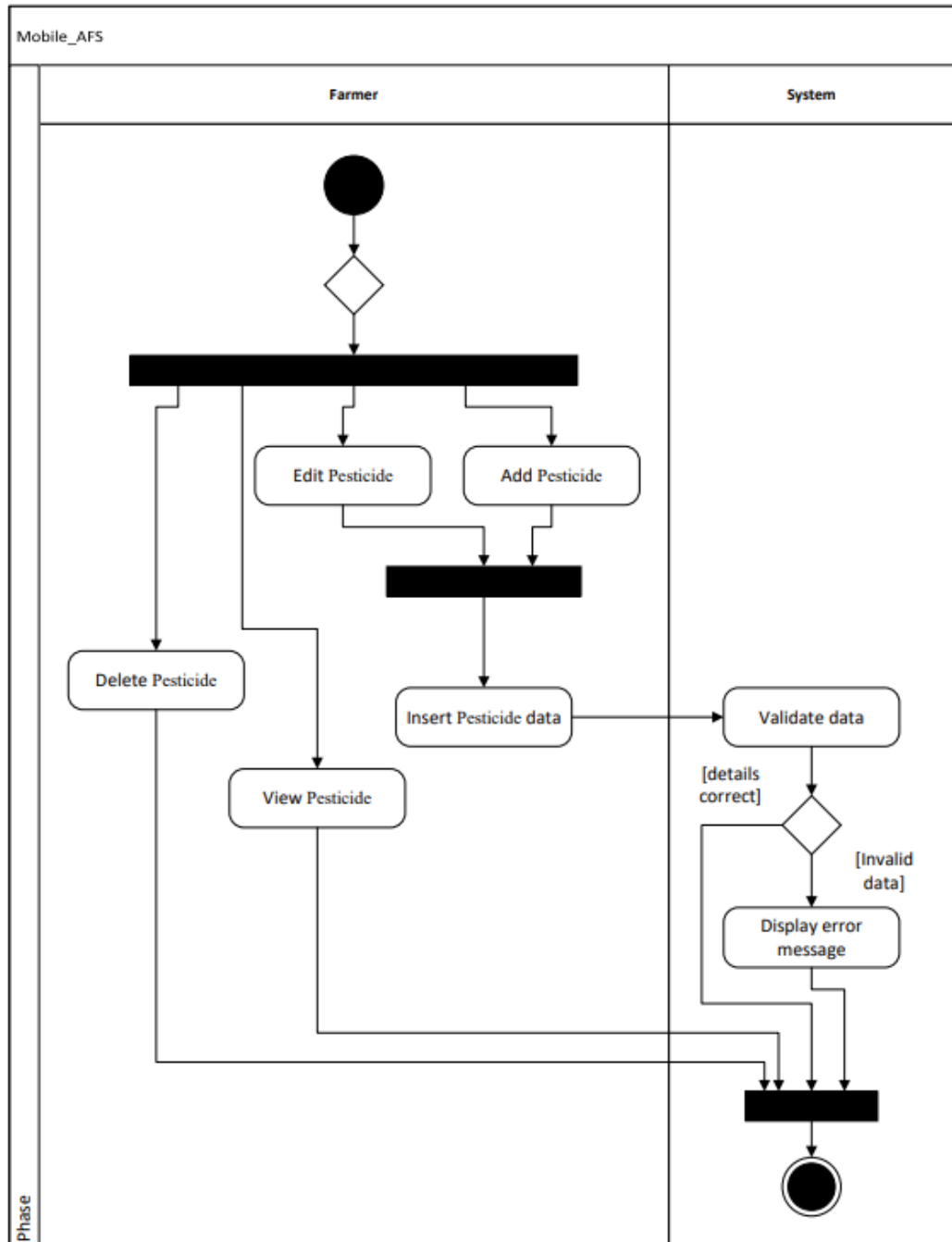


Figure 3.17 Activity diagram for Manage Pesticide

### 3.4.3.6 Manage Sales

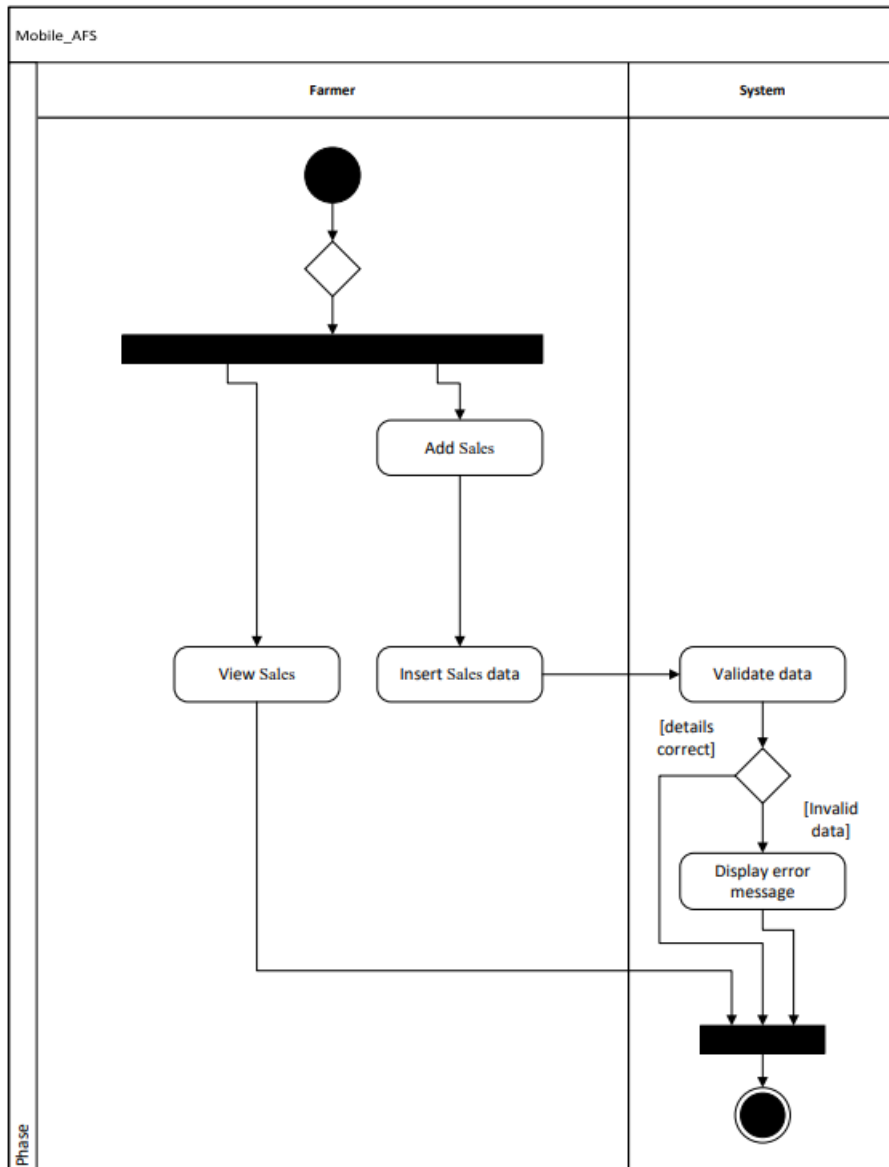


Figure 3.18 Activity diagram for Manage Sales



### 3.4.3.7 Manage Project

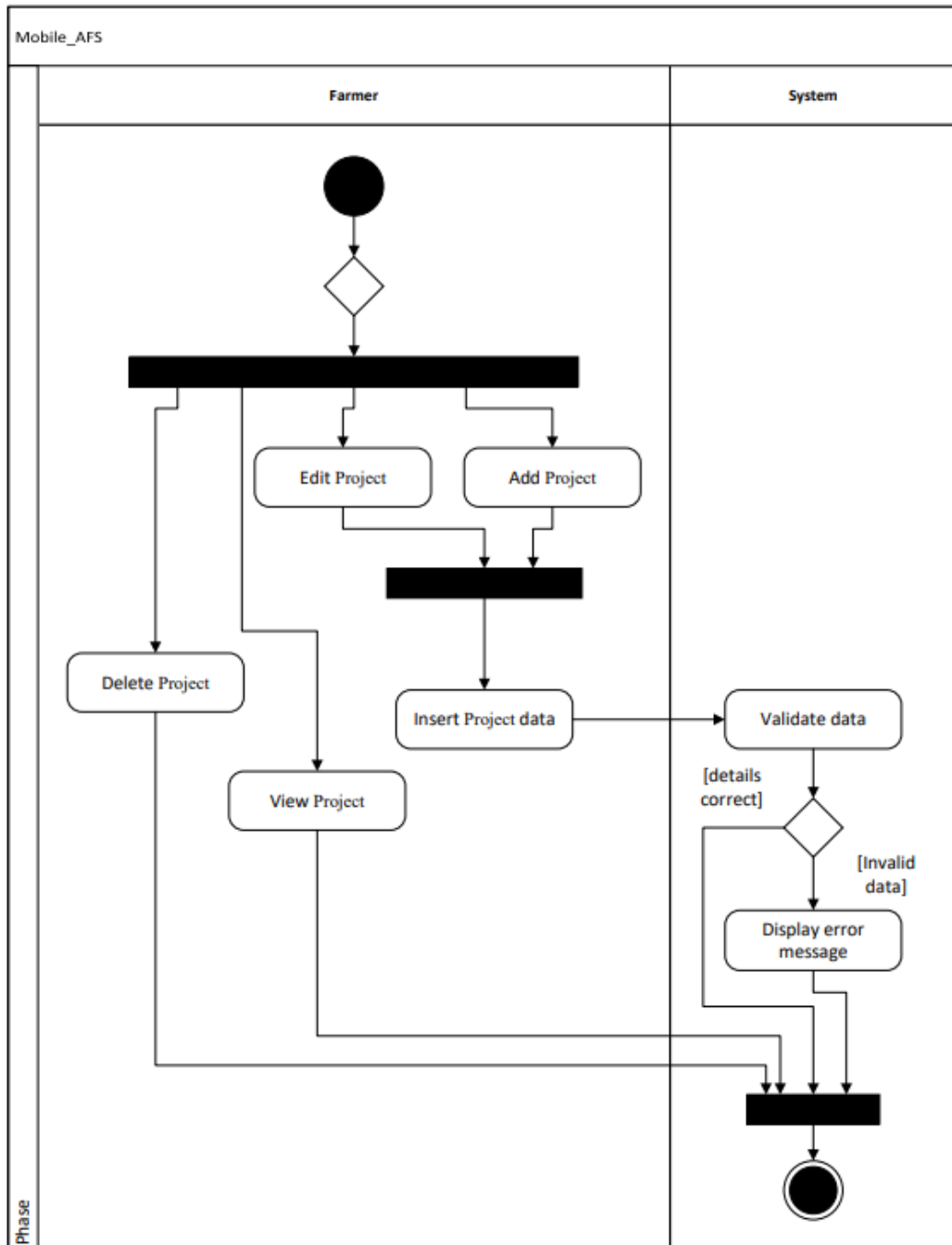


Figure 3.19 Activity diagram for Manage Project

### 3.4.3.8 Manage Purchase

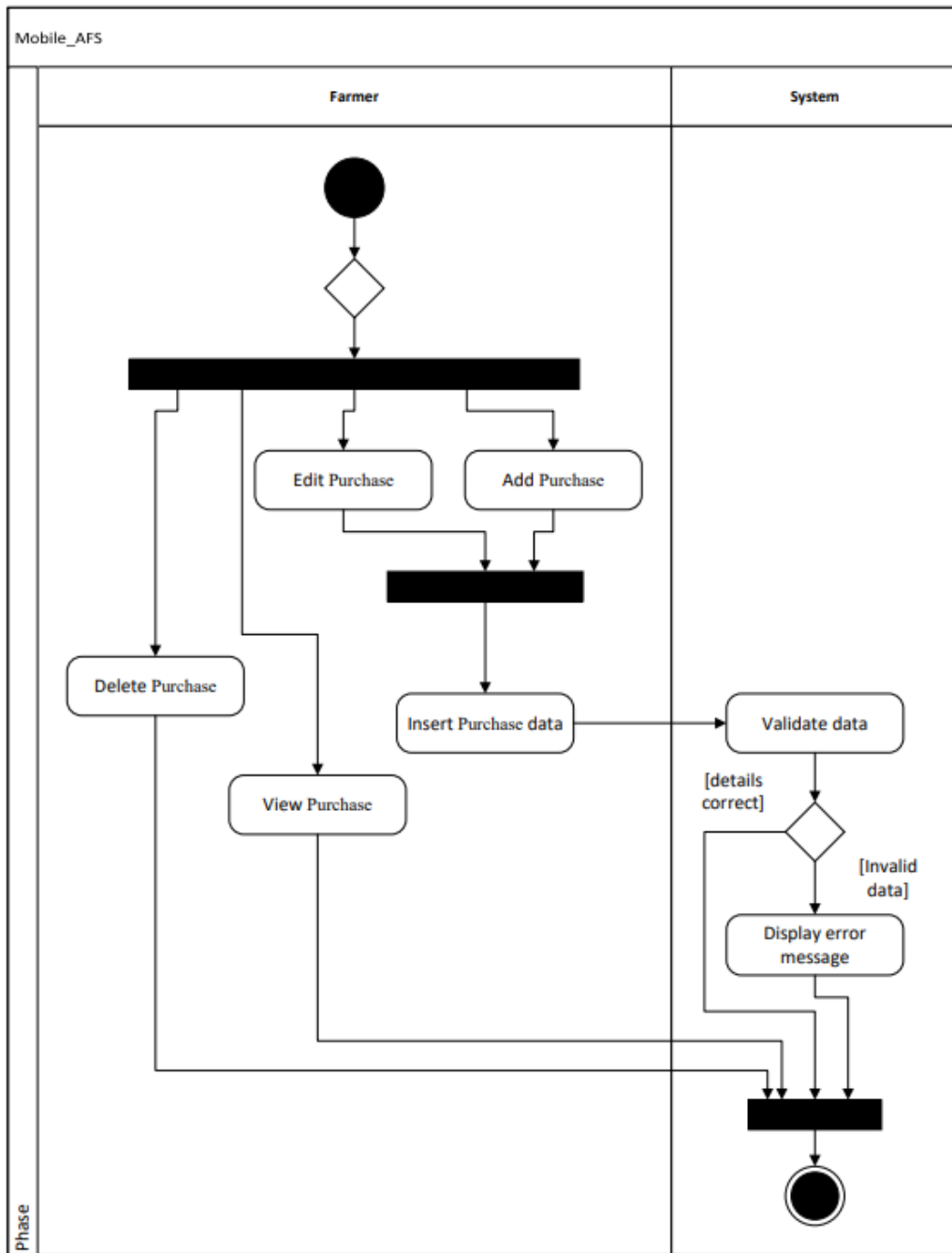


Figure 3.20 Activity diagram for Manage Purchase

### 3.4.3.9 Manage Inventory

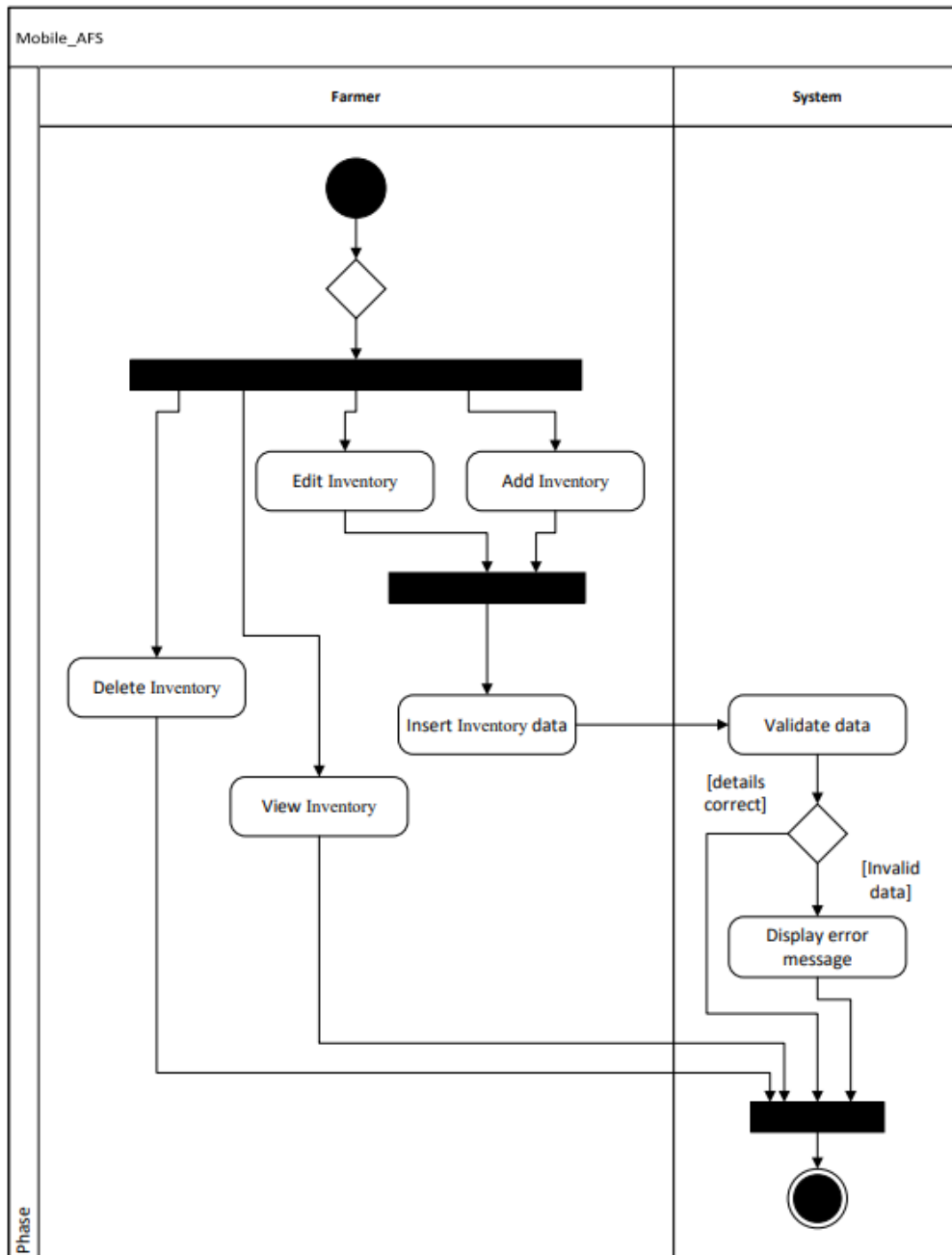


Figure 3.21 Activity diagram for Manage Inventory

### 3.4.4 Context Diagram

Context diagrams are used to illustrate specific situations and limitations of the demonstrated framework. The general questions asked are what the role of the system is and what are the linkages between the system and external entities.

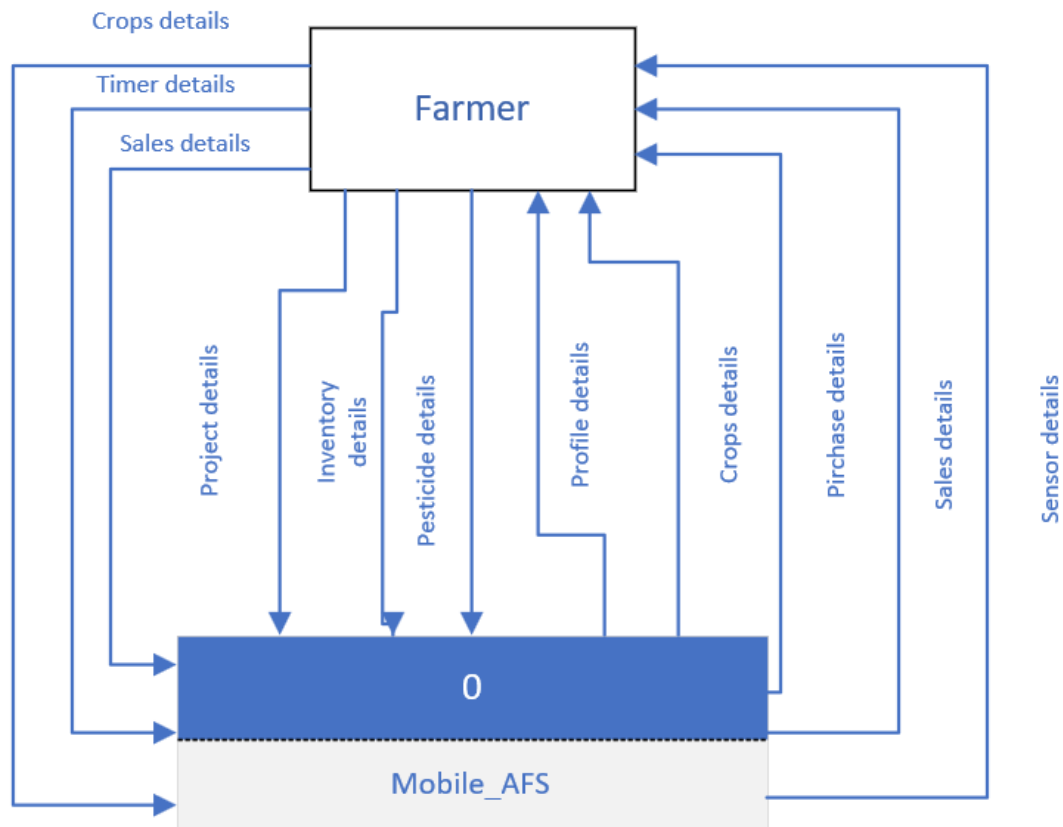


Figure 3.22 Context Diagram for Mobile\_AFS

Figure 3.22 shows a level 0 context diagram for the Mobile\_AFS. Farmer is the external entity that exists in this system. The farmer will have access to insert details like crops, timer, sales, project, inventory, and pesticide details. The system will return some details like profile, crops, purchase, sales, and sensor details to the farmer.

### 3.4.5 Interface design/ Storyboard

#### 3.4.5.1 Module Login

This interface shows the storyboard for Farmer Login, Register and Profile. The farmer will need to enter their correct email and password to login to the Main interface. Besides, farmer can click on the forget password to reset their password by sending verification email to their email.

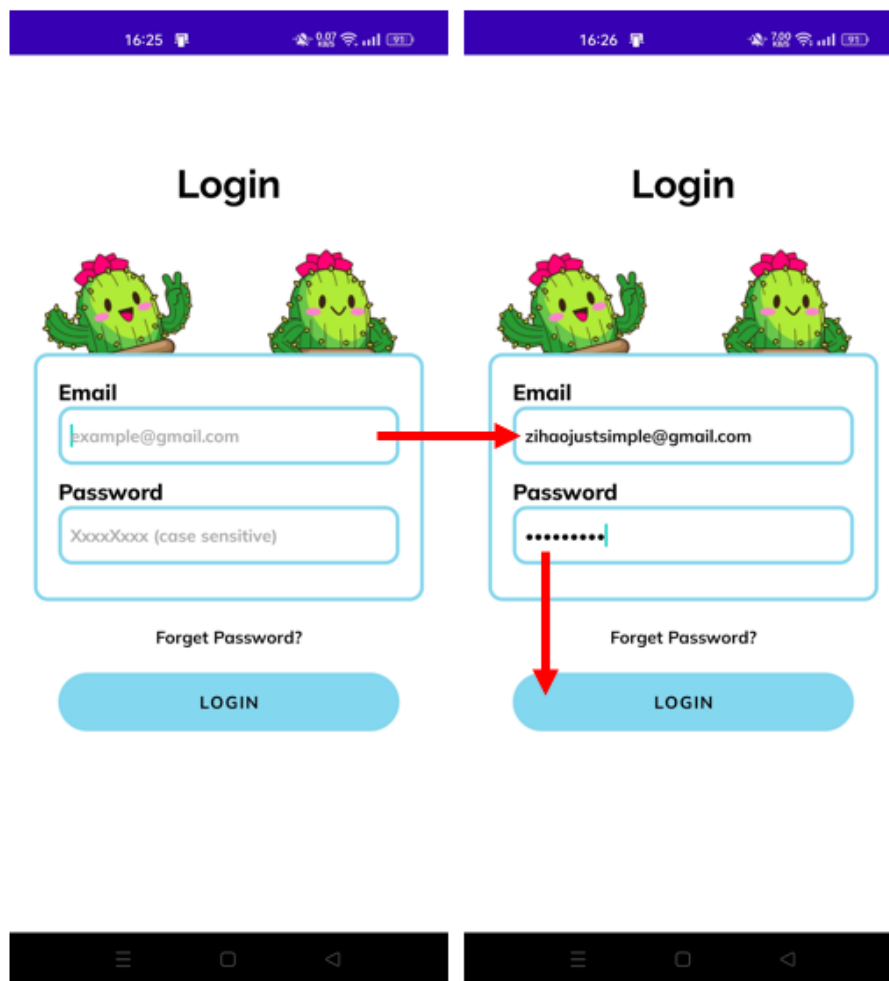


Figure 3.23 Storyboard for Farmer Login and Forget Password

### 3.4.5.2 Main

After farmer login, the main interface contains a lot of navigation to another module. At the bottom, we have the Manage Crops, Sensor, Timer, and Menu. The functions contain navigation to the Manage Pesticide, Sales, Schedule, Project, Purchase and Inventory.

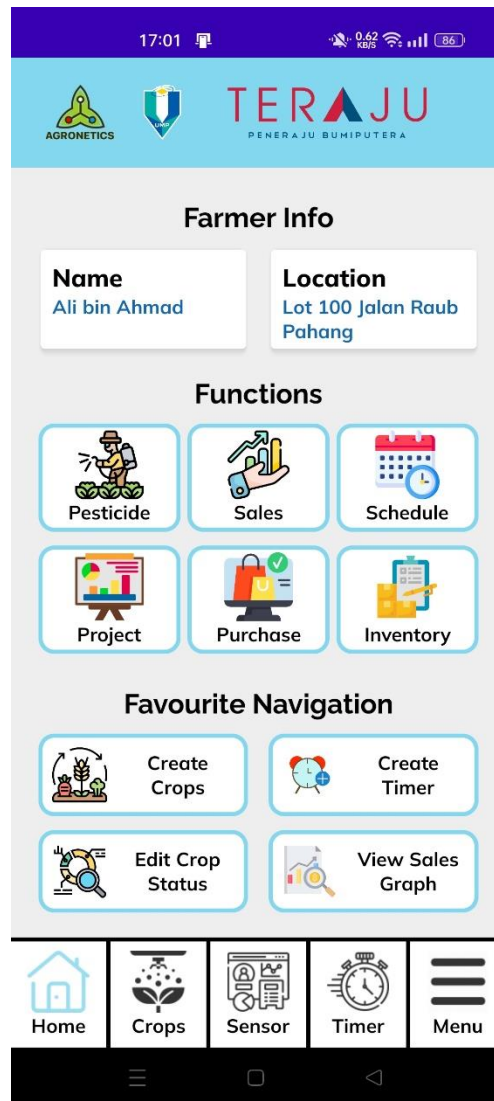


Figure 3.24 Storyboard for Main Interface

### 3.4.5.3 Module Profile

Farmer need to click on the menu on the bottom navigation at the Main interface. User can click the “Profile” to navigate to the Profile interface. The user can update any data in the textbox and click the update button. A success update Toast will appear if the data is updated at the database.

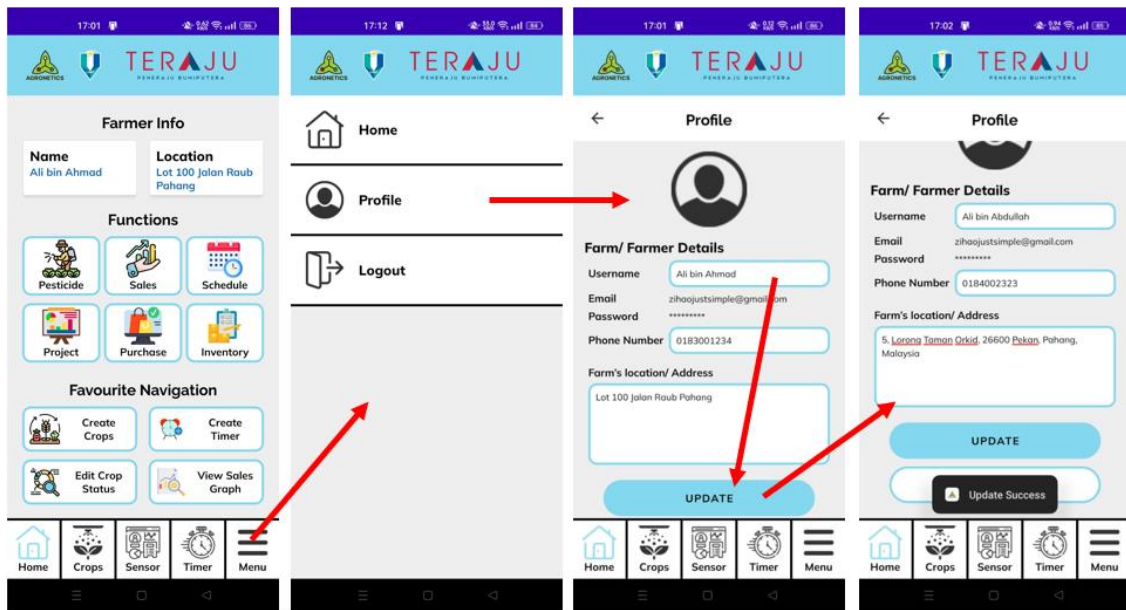


Figure 3.25 Storyboard for Manage Profile

### 3.4.5.4 Module Crops

This interface shows the storyboard for Manage Crops. In the Main menu, farmer can choose to Manage Crops or Edit Crops Status. If the farmer chooses to click Manage Crops, farmers can create, view, update and delete the crops. While the farmer chooses to Edit Crops Status, farmer can update the quantity of the crops and status.

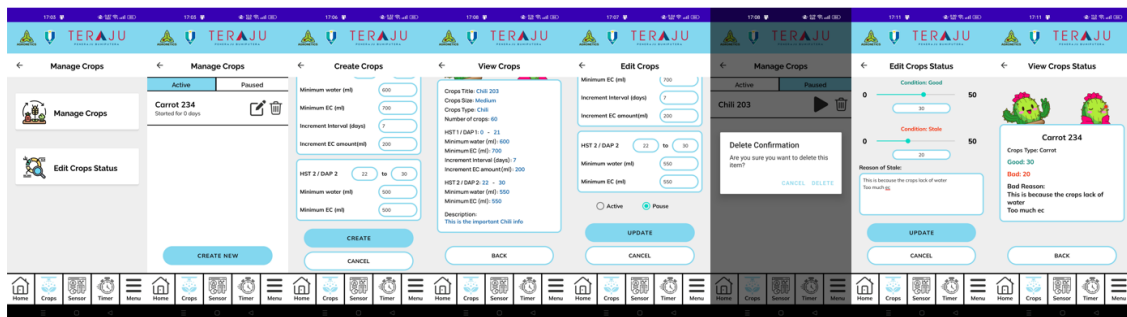


Figure 3.26 Storyboard for Manage Crops



### 3.4.5.5 Module Sensor

Farmer need to click on the” Sensor” to navigate to Sensor interface. All the details of the sensor will display to the farmer.

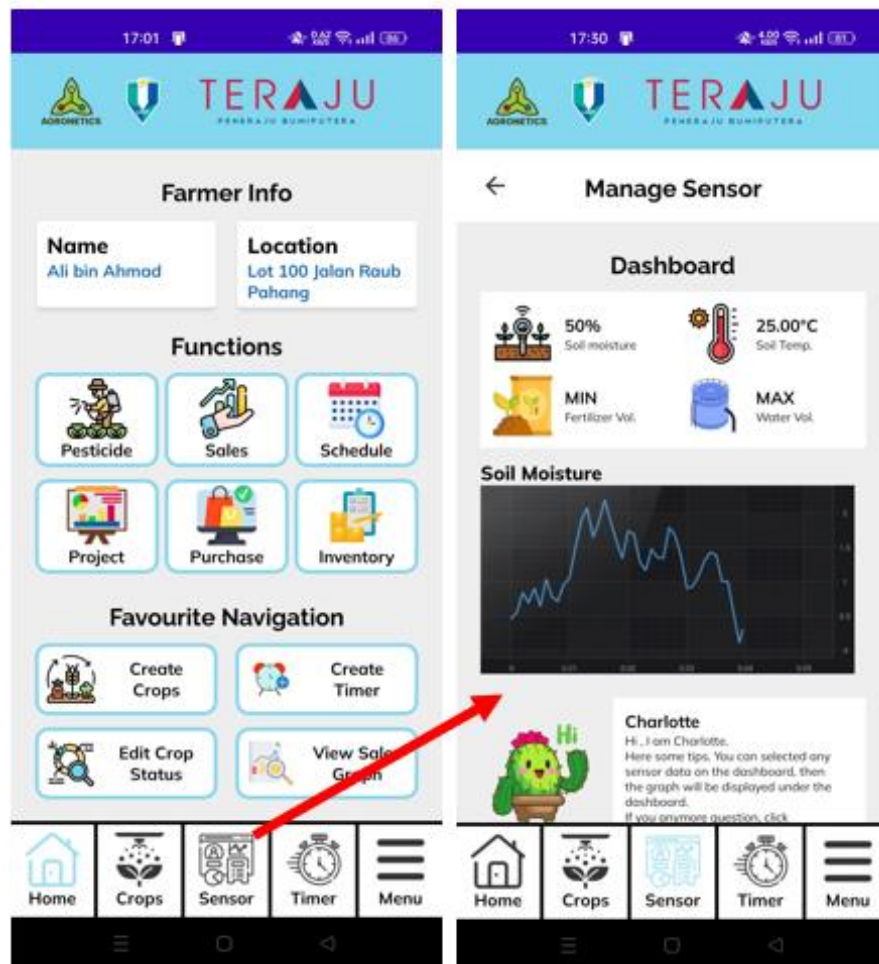


Figure 3.27 Storyboard for Manage Sensor

### 3.4.5.6 Module Timer

This interface shows the storyboard for Manage Timer. In the Main menu, farmer can choose to Add, View, Update and Delete timer.

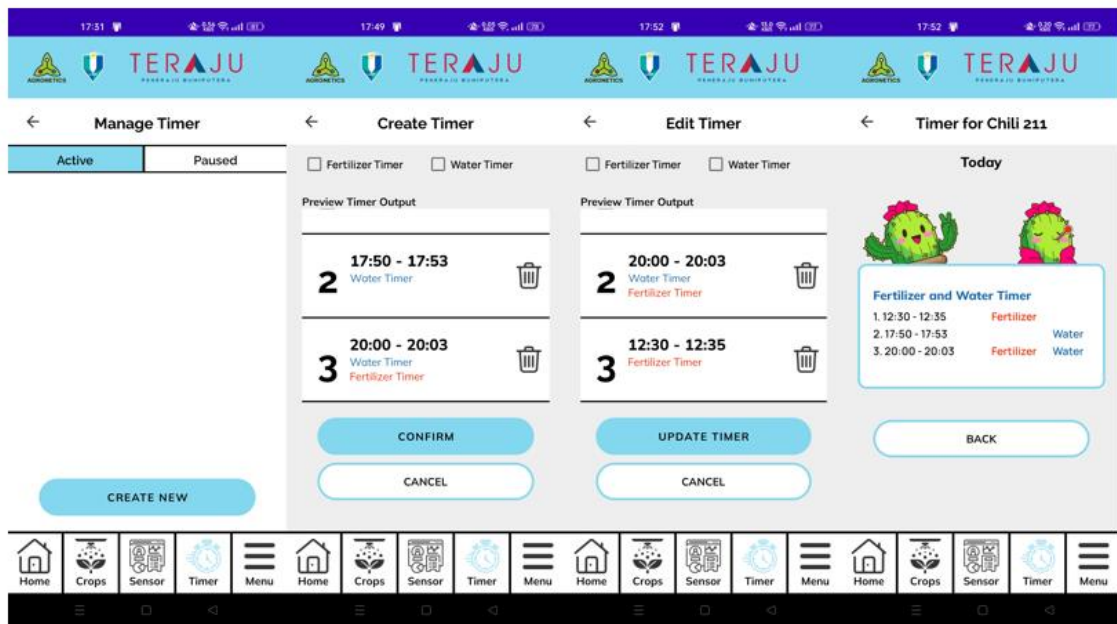


Figure 3.28 Storyboard for Manage Timer

### 3.4.5.7 Module Pesticide

This interface shows the storyboard for Manage Pesticide. In the Main menu, farmer can choose to Add, View, Update and Delete Pesticide.

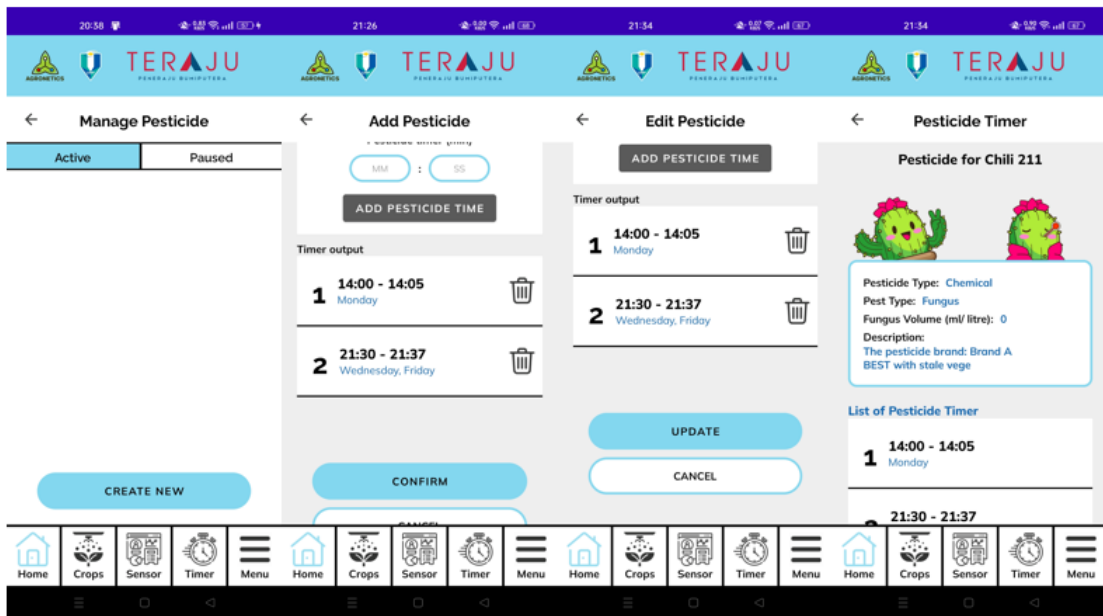


Figure 3.29 Storyboard for Manage Pesticide

### 3.4.5.8 Module Sales

This interface shows the storyboard for Manage Sales. In the Main menu, farmer can choose to Add and View Sales. For editing, the farmer does not have the authority to update or delete the sales. Therefore, they need to call the Teraju admin to edit or delete their sales data. Besides, in this module farmer can view their recent graph of sales. Daily sales indicate all the crops sales, while crops graph will filter the sales based on crops.

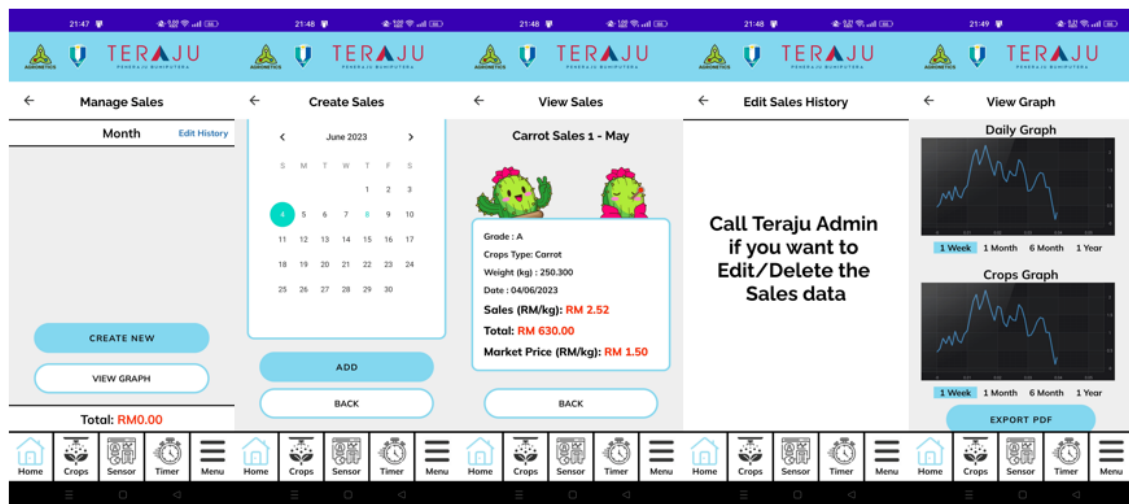


Figure 3.30 Storyboard for Manage Sales

### 3.4.5.9 Module Schedule

This interface shows the storyboard for Manage Schedule. In the Main menu, farmer can choose to Add, View, Edit and Delete Schedule.

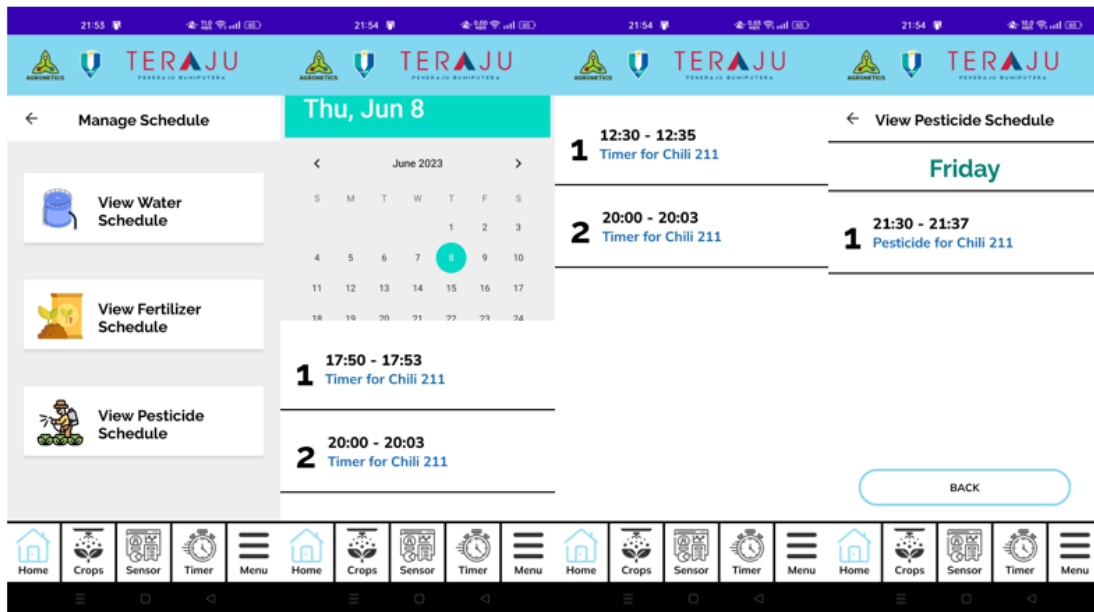


Figure 3.31 Storyboard for Manage Schedule

### 3.4.5.10 Module Project

This interface shows the storyboard for Manage Project. In the Main menu, farmer can choose to Add, View, Edit and Delete Project.



Figure 3.32 Storyboard for Manage Project

### 3.4.5.11 Module Purchase

This interface shows the storyboard for Manage Purchase. In the Main menu, farmer can choose to Add, View, Edit and Delete Purchase.

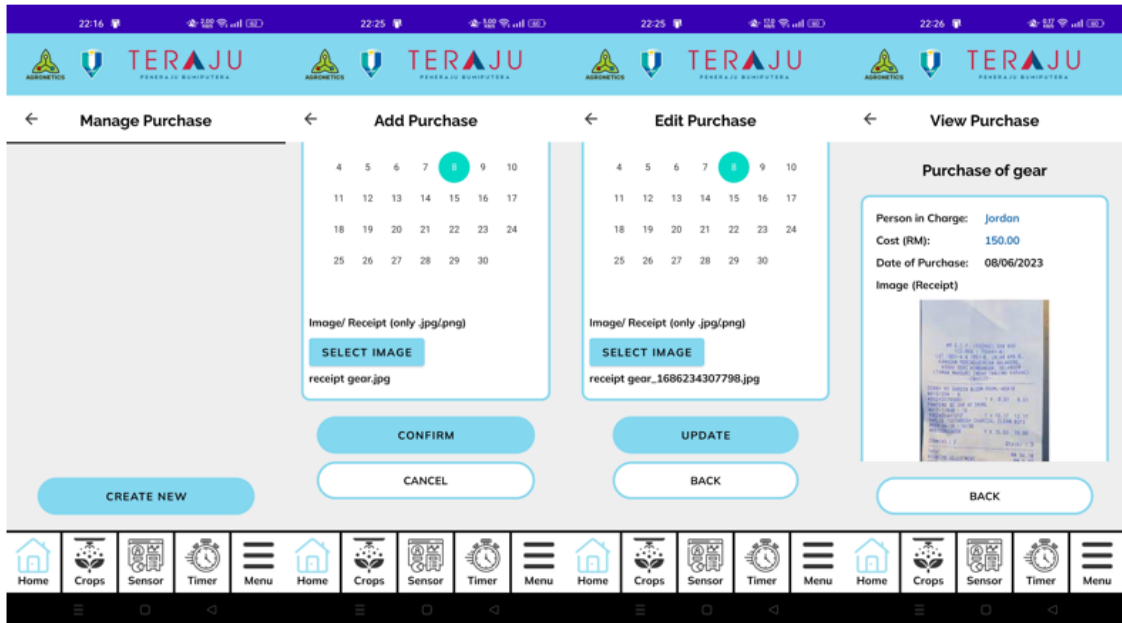


Figure 3.33 Storyboard for Manage Purchase

### 3.4.5.12 Module Inventory

This interface shows the storyboard for Manage Inventory. In the Main menu, farmer can choose to Add, View, Edit and Delete Inventory.



Figure 3.34 Storyboard for Manage Inventory



### 3.5 Data Design

#### 3.5.1 ERD



Figure 3.35 ERD for Mobile\_AFS

### 3.5.2 Data Dictionary

#### 3.5.2.1 Autotimer Table

Table 3.15 Data Dictionary of AutotimerTable

Field Name	Description	Data Type	Constraint
unique_id	Autotimer Identification ID	INTEGER	PK, Auto Increment
cropsID	Crops Identification ID	INTEGER	FK
type	Type of the autotimer	VARCHAR (15)	Not null
title	Autotimer title	VARCHAR (50)	Not null
isActive	The status of the timer is autotimer	BOOLEAN	Not null
schedule	The schedule time for the timer	TEXT	
water_flag	The flag that controls the water to start irrigating when 1	TEXT	
fertilizer_flag	The flag that controls the fertilizer to start irrigating when 1	TEXT	
last_modified	The date that last modified data	DATETIME	
waterTimers	The water timer's length	VARCHAR (100)	Not null
fertilizerTimers	The water timer's length	VARCHAR (100)	Not null
status	The status of the timer is active or not	BOOLEAN	Not null

dateCreated	The timestamp for creating the timer	BIGINT	Not null
-------------	--------------------------------------	--------	----------

### 3.5.2.2 Controltest Table

Table 3.16 Data Dictionary of Controltest Table

Field Name	Description	Data Type	Constraint
controltestID	ControlTest Identification ID	INTEGER	PK, Auto Increment
farmID	Farm Identification ID	INTEGER	FK
timestamp	The timestamp recording the sensor data	INTEGER	Not null
water	The water sensor status	VARCHAR (10)	Not null
fertilizer	The fertilizer sensor status	VARCHAR (10)	Not null
moisture	The soil moisture sensor status	INTEGER	Not null
ec	The ec sensor status	INTEGER	Not null
temperature	The soil temperature sensor status	FLOAT	Not null

### 3.5.2.3 Crops Table

Table 3.17 Data Dictionary of Crops Table

Field Name	Description	Data Type	Constraint
cropsID	Crops Identification ID	INTEGER	PK, Auto Increment
farmID	Farm Identification ID	INTEGER	FK
title	The title of the crops	VARCHAR (50)	Not null
size	The size of the crops	VARCHAR (15)	Not null
type	The type of the crops	VARCHAR (15)	Not null
description	The description of the crops	VARCHAR (255)	Not null
cropsNum	The number of the crops	INTEGER	Not null
HST1Start	The start day after planting stage 1 of the crops	INTEGER	Not null
HST1End	The end day after planting stage 1 of the crops	INTEGER	Not null
minWater1	The minimum water stage 1 of the crops	INTEGER	Not null
minEC1	The minimum ec stage 1 of the crops	INTEGER	Not null
incrementInterval	The increment day for planting stage 1 of the crops	INTEGER	Not null

incrementAmt	The increment volume for planting stage 1 of the crops	INTEGER	Not null
HST2Start	The start day after planting stage 2 of the crops	INTEGER	Not null
HST2End	The end day after planting stage 2 of the crops	INTEGER	Not null
minWater2	The minimum water stage 2 of the crops	INTEGER	Not null
minEC2	The minimum ec stage 2 of the crops	INTEGER	Not null
good	The number of the crops that is good in condition	INTEGER	Not null
bad	The number of the crops that is bad in condition	INTEGER	Not null
badReason	The reason that the crops is bad	VARCHAR (255)	
dateCreated	The timestamp of the date created of the crops	BIGINT	Not null
status	The status of the crops	BOOLEAN	Not null

### 3.5.2.4 Farm Table

Table 3.18 Data Dictionary of Farm Table

Field Name	Description	Data Type	Constraint
farmID	Farm Identification ID	INTEGER	PK, Auto Increment
farmerID	Farmer Identification ID	INTEGER	FK
title	Name of the farm	VARCHAR (30)	Not null
address	Address of the farm	VARCHAR (255)	Not null

### 3.5.2.5 Farmers Table

Table 3.19 Data Dictionary of Farmers Table

Field Name	Description	Data Type	Constraint
id	Farmer Identification ID	INTEGER	PK, Auto Increment
email	The email of the farmer	VARCHAR (255)	Not null
password	The password of the farmer	VARCHAR (20)	Not null
fmname	The username of the farmer	VARCHAR (255)	Not null
phone	The phone number of the farmer	VARCHAR (13)	Not null
location	The location of the farmer	VARCHAR (255)	Not null
reset_token	The reset password token for farmer	TEXT	

### 3.5.2.6 Inventorys Table

Table 3.20 Data Dictionary of Inventorys Table

Field Name	Description	Data Type	Constraint
inventoryID	Inventory Identification ID	INTEGER	PK, Auto Increment
purchaseID	Purchase Identification ID	INTEGER	FK
inventoryName	The inventory name	VARCHAR (50)	Not null
inventoryPIC	The Person in Charge for the inventory	VARCHAR (255)	Not null
quantity	The inventory quantity	INTEGER	Not null
price	The inventory price	DOUBLE	Not null
dateStored	The inventory storing date	DATE	Not null
status	The inventory status	VARCHAR (15)	Not null
lastModified	The inventory last modified date	DATE	Not null



### 3.5.2.7 Pesticide Table

Table 3.21 Data Dictionary of Pesticide Table

Field Name	Description	Data Type	Constraint
pesticideID	Pesticide Identification ID	INTEGER	PK, Auto Increment
cropsID	Crops Identification ID	INTEGER	FK
pesticideTitle	Title of the pesticide	VARCHAR (50)	Not null
pesticideType	Type of the pesticide	VARCHAR (10)	Not null
pestType	Type of the pest	VARCHAR (10)	Not null
fungusVolume	The volume of the fungus pesticide	DOUBLE	
nVolume	The volume of the nerve pesticide	DOUBLE	
eVolume	The volume of the energy pesticide	DOUBLE	
gVolume	The volume of the growth pesticide	DOUBLE	
description	Description of the pesticide	VARCHAR (255)	Not null
days	Day watering the pesticide	VARCHAR (255)	Not null
schedule	Schedule time of the pesticide	VARCHAR (255)	Not null
timer	The length of the timer for the pesticide	VARCHAR (255)	Not null
status	The status of the pesticide	BOOLEAN	Not null

dateCreated	The date created for the pesticide	BIGINT	Not null
-------------	------------------------------------	--------	----------

### 3.5.2.8 Project Table

Table 3.22 Data Dictionary of Project Table

Field Name	Description	Data Type	Constraint
projectID	Project Identification ID	INTEGER	PK, Auto Increment
farmID	Farm Identification ID	INTEGER	FK
title	The title for the project	VARCHAR (100)	Not null
budget	The budget for the project	DOUBLE	Not null
startDate	The start date for the project	DATE	Not null
endDate	The end date for the project	DATE	Not null
status	The status for the project	BOOLEAN	Not null

### 3.5.2.9 Purchase Table

Table 3.23 Data Dictionary of Purchase Table

Field Name	Description	Data Type	Constraint
purchaseID	Purchase Identification ID	INTEGER	PK, Auto Increment
projectID	Project Identification ID	INTEGER	FK
title	The title of the purchase	VARCHAR (100)	Not null
projectPIC	The Person In Charge of the purchase	VARCHAR (255)	Not null
price	The price of the purchase	DOUBLE	Not null
date	The date of the purchase	DATE	Not null
imageUri	The imag uri for the purchase	VARCHAR (255)	Not null

### 3.5.2.10 Sales Table

Table 3.24 Data Dictionary of Sales Table

Field Name	Description	Data Type	Constraint
salesID	Sales Identification ID	INTEGER	PK, Auto Increment
farmID	Farm Identification ID	INTEGER	FK
salesTitle	The title of the sales	VARCHAR (50)	Not null
salesGrade	The grade of the sales	VARCHAR (5)	Not null

cropsType	The crops type that is selling	VARCHAR (15)	Not null
salesWeight	The weight of the sales	DOUBLE	Not null
salesPrice	The sales price of the sales	DOUBLE	Not null
marketPrice	The market price of the sales	DOUBLE	Not null
salesDate	The sales date for the sales	DATE	Not null

### 3.5.2.11 Sales\_history Table

Table 3.25 Data Dictionary of Sales\_history Table

Field Name	Description	Data Type	Constraint
historyID	Sales History Identification ID	INTEGER	PK, Auto Increment
salesID	Sales Identification ID	INTEGER	FK
title	The title of the sales history	VARCHAR (50)	Not null
description	The description of the sales history	VARCHAR (255)	Not null
status	The status of the sales history	BOOLEAN	Not null
rejectReason	The reject reason of the sales history	VARCHAR (255)	

## 3.6 Testing/Validation Plan

### 3.6.1 Module Login

#### 3.6.1.1 Test Case Description for Login

Table 3.26 Test Case for Login

Test Case ID	TC-01-01
Description	To test the users can login to the main interface.
Pre-Condition	Users had registered an account.
Test Input	<ol style="list-style-type: none"><li>1. The system displays login form to fill in.</li><li>2. Users fill in the email address, and password.</li><li>3. Click &lt;&lt;Login&gt;&gt;.</li></ol>
Expected Test Result	System directs the user to the main interface.

#### 3.6.1.2 Test Case Description for Forget Password

Table 3.27 Test Case for Forget Password

Test Case ID	TC-01-02
Description	To test the users can reset their password when forget password.
Pre-Condition	Users had registered an account.
Test Input	<ol style="list-style-type: none"><li>1. Users click &lt;&lt;Forget Password&gt;&gt; button.</li><li>2. The system displays success sent reset password link to the user email.</li></ol>
Expected Test Result	The user will receive the reset password verification email.

### 3.6.2 Module Profile

#### 3.6.2.1 Test Case Description for Profile

Table 3.28 Test Case for Profile Farmer

Test Case ID	TC-02-01
Description	To test the user can edit their profile details.
Pre-Condition	User had login to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt; Profile&gt;&gt; button.</li><li>2. The system displays profile details.</li><li>3. User update in the username, phone, or address.</li><li>4. User clicks &lt;&lt;Update&gt;&gt;.</li></ol>
Expected Test Result	The updated profile data is saved to the database.

### 3.6.3 Manage Crops

#### 3.6.3.1 Test Case Description for Add Crops

Table 3.29 Test Case for Add Crops

Test Case ID	TC-03-01
Description	To determine whether the user can add new crops to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display crops form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new crop is added.</li><li>• System direct user to Crops Menu.</li></ul>

#### 3.6.3.2 Test Case Description for View Crops

Table 3.30 Test Case for View Crops

Test Case ID	TC-03-02
Description	To determine whether the user can view crops details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all crops details.</li></ol>
Expected Test Result	The crops details are correctly display to the user.

### 3.6.3.3 Test Case Description for Update Crops

Table 3.31 Test Case for Update Crops

Test Case ID	TC-03-03
Description	To determine whether the user can update crops details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all crops details.</li><li>3. User updated the crops details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated crops details are updated to the database.</li><li>• System direct user to Crops Menu.</li></ul>

### 3.6.3.4 Test Case Description for Delete Crops Details

Table 3.32 Test Case for Delete crops details

Test Case ID	TC-03-04
Description	To determine whether the user can delete crops from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion crops.</li></ol>
Expected Test Result	The selected crops deleted from the database.



### 3.6.3.5 Test Case Description for Edit Crops Status

Table 3.33 Test Case for Edit Crops Status

Test Case ID	TC-03-05
Description	To determine whether the user can update the crops details.
Pre-Condition	<ul style="list-style-type: none"><li>• The user has already log in to the system.</li><li>• At least of 1 crop is created in the farm</li></ul>
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Edit&gt;&gt; button.</li><li>2. The system displays the crops status.</li><li>3. User updated the crops status.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	The IOT device is successfully connected to the crops.

### 3.6.4 Manage Timer

#### 3.6.4.1 Test Case Description for Add Timer

Table 3.34 Test Case for Add Timer

Test Case ID	TC-04-01
Description	To determine whether the user can add new Timer to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Timer form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Timer is added.</li><li>• System direct user to Timer Menu.</li></ul>

#### 3.6.4.2 Test Case Description for View Timer

Table 3.35 Test Case for View Timer

Test Case ID	TC-04-02
Description	To determine whether the user can view Timer details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Timer details.</li></ol>
Expected Test Result	The Timer details are correctly display to the user.

### 3.6.4.3 Test Case Description for Update Timer

Table 3.36 Test Case for Update Timer

Test Case ID	TC-04-03
Description	To determine whether the user can update Timer details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all Timer details.</li><li>3. User updated the Timer details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated Timer details are updated to the database.</li><li>• System direct user to Timer Menu.</li></ul>

### 3.6.4.4 Test Case Description for Delete Timer

Table 3.37 Test Case for Delete Timer

Test Case ID	TC-04-04
Description	To determine whether the user can delete Timer from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion.</li></ol>
Expected Test Result	The selected Timer deleted from the database.

### 3.6.5 Manage Pesticide

#### 3.6.5.1 Test Case Description for Add Pesticide

Table 3.38 Test Case for Add Pesticide

Test Case ID	TC-05-01
Description	To determine whether the user can add new Pesticide to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Pesticide form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Pesticide is added.</li><li>• System direct user to Pesticide Menu.</li></ul>

#### 3.6.5.2 Test Case Description for View Pesticide

Table 3.39 Test Case for View Pesticide

Test Case ID	TC-05-02
Description	To determine whether the user can view Pesticide details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Pesticide details.</li></ol>
Expected Test Result	The Pesticide details are correctly display to the user.

### 3.6.5.3 Test Case Description for Update Pesticide

Table 3.40 Test Case for Update Pesticide

Test Case ID	TC-05-03
Description	To determine whether the user can update Pesticide details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all Pesticide details.</li><li>3. User updated the Pesticide details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated Pesticide details are updated to the database.</li><li>• System direct user to Pesticide Menu.</li></ul>

### 3.6.5.4 Test Case Description for Delete Pesticide

Table 3.41 Test Case for Delete Pesticide

Test Case ID	TC-05-04
Description	To determine whether the user can delete Pesticide from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion.</li></ol>
Expected Test Result	The selected Pesticide deleted from the database.

### 3.6.6 Manage Sales

#### 3.6.6.1 Test Case Description for Add Sales

Table 3.42 Test Case for Add Sales

Test Case ID	TC-06-01
Description	To determine whether the user can add new Sales to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Sales form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Sales is added.</li><li>• System direct user to Sales Menu.</li></ul>

#### 3.6.6.2 Test Case Description for View Sales

Table 3.43 Test Case for View Sales

Test Case ID	TC-06-02
Description	To determine whether the user can view Sales details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Sales details.</li></ol>
Expected Test Result	The Sales details are correctly display to the user.

### 3.6.7 Manage Project

#### 3.6.7.1 Test Case Description for Add Project

Table 3.44 Test Case for Add Project

Test Case ID	TC-07-01
Description	To determine whether the user can add new Project to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Project form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Project is added.</li><li>• System direct user to Project Menu.</li></ul>

#### 3.6.7.2 Test Case Description for View Project

Table 3.45 Test Case for View Project

Test Case ID	TC-07-02
Description	To determine whether the user can view Project details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Project details.</li></ol>
Expected Test Result	The Project details are correctly display to the user.

### 3.6.7.3 Test Case Description for Update Project

Table 3.46 Test Case for Update Project

Test Case ID	TC-07-03
Description	To determine whether the user can update Project details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all Project details.</li><li>3. User updated the Project details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated Project details are updated to the database.</li><li>• System direct user to Project Menu.</li></ul>

### 3.6.7.4 Test Case Description for Delete Project

Table 3.47 Test Case for Delete Project

Test Case ID	TC-07-04
Description	To determine whether the user can delete Project from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion.</li></ol>
Expected Test Result	The selected Project deleted from the database.



### 3.6.8 Manage Purchase

#### 3.6.8.1 Test Case Description for Add Purchase

Table 3.48 Test Case for Add Purchase

Test Case ID	TC-08-01
Description	To determine whether the user can add new Purchase to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Purchase form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Purchase is added.</li><li>• System direct user to Purchase Menu.</li></ul>

#### 3.6.8.2 Test Case Description for View Purchase

Table 3.49 Test Case for View Purchase

Test Case ID	TC-08-02
Description	To determine whether the user can view Purchase details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Purchase details.</li></ol>
Expected Test Result	The Purchase details are correctly display to the user.

### 3.6.8.3 Test Case Description for Update Purchase

Table 3.50 Test Case for Update Purchase

Test Case ID	TC-05-03
Description	To determine whether the user can update Purchase details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all Purchase details.</li><li>3. User updated the Purchase details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated Purchase details are updated to the database.</li><li>• System direct user to Purchase Menu.</li></ul>

### 3.6.8.4 Test Case Description for Delete Purchase

Table 3.51 Test Case for Delete Purchase

Test Case ID	TC-05-04
Description	To determine whether the user can delete Purchase from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion.</li></ol>
Expected Test Result	The selected Purchase deleted from the database.

### 3.6.9 Manage Inventory

#### 3.6.9.1 Test Case Description for Add Inventory

Table 3.52 Test Case for Add Inventory

Test Case ID	TC-09-01
Description	To determine whether the user can add new Inventory to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Create new&gt;&gt; button.</li><li>2. The system display Inventory form.</li><li>3. The user enters all the required details.</li><li>4. The user clicks &lt;&lt;Create&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The new Inventory is added.</li><li>• System direct user to Inventory Menu.</li></ul>

#### 3.6.9.2 Test Case Description for View Inventory

Table 3.53 Test Case for View Inventory

Test Case ID	TC-09-02
Description	To determine whether the user can view Inventory details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;View&gt;&gt; button.</li><li>2. The system displays all Inventory details.</li></ol>
Expected Test Result	The Inventory details are correctly display to the user.

### 3.6.9.3 Test Case Description for Update Inventory

Table 3.54 Test Case for Update Inventory

Test Case ID	TC-09-03
Description	To determine whether the user can update Inventory details to the system.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Update&gt;&gt; button.</li><li>2. The system displays all Inventory details.</li><li>3. User updated the Inventory details.</li><li>4. User clicks &lt;&lt;Update&gt;&gt; button.</li></ol>
Expected Test Result	<ul style="list-style-type: none"><li>• The updated Inventory details are updated to the database.</li><li>• System direct user to Inventory Menu.</li></ul>

### 3.6.9.4 Test Case Description for Delete Inventory

Table 3.55 Test Case for Delete Inventory

Test Case ID	TC-09-04
Description	To determine whether the user can delete Inventory from database.
Pre-Condition	The user has already log in to the system.
Test Input	<ol style="list-style-type: none"><li>1. User clicks &lt;&lt;Delete&gt;&gt; button.</li><li>2. The system displays alert box.</li><li>3. The user selects &lt;&lt;Yes&gt;&gt; to confirm deletion.</li></ol>
Expected Test Result	The selected Inventory deleted from the database.

### 3.7 Potential Use of Proposed Solution



Figure 3.36 Figure for prototype farm

Above show the prototype farm at Faculty Manufacturing which represent the 50 crops in real time. All the crops will be irrigating by water and fertilizer through the pipe connected to it.



Figure 3.37 Figure for control panel and water tank

Above show the control panel and the water tanks for the system. From the control panel we can know that it can control the distribution of water and fertilizer to the crops. Besides, it can fill the water and fertilizer tanks also. It can indicate the water tank and fertilizer tank is in low or high by light the LED light at the control panel.



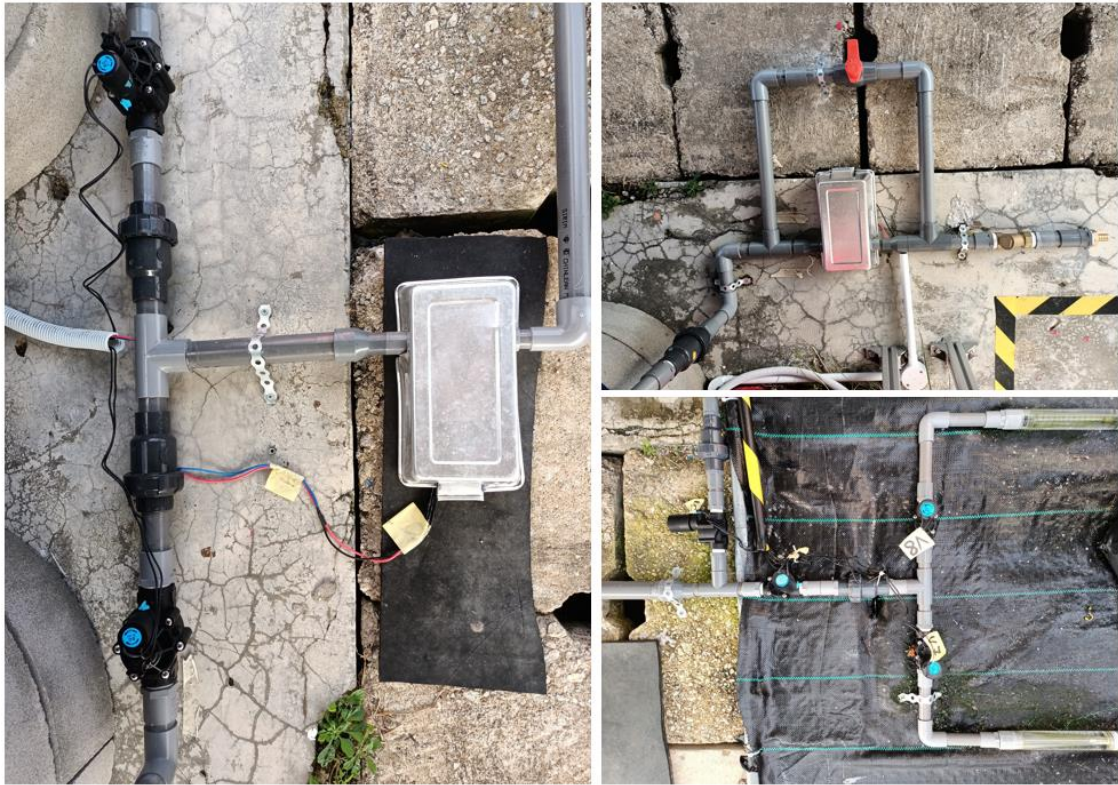


Figure 3.38 Figure for valve and piping

The valves can be either manually operated by hand or opened and closed remotely via a control panel to control the flow of water into the crop.

Therefore, the potential use of the proposed solution is we going to develop a mobile system to replace or enhance the capability of the control panel. Previously, the control panel needed a person to control and monitor in front of it. Now, the person in charge can just sit back and relax in the office and get know to the status of all water tank and crops just using a fingertip.

Farmers can now monitor or understand the status of their crops by looking at the Mobile\_AFS dashboard. Details of sensor data such as soil moisture, soil temperature, tank and volume in the tank can now be updated to the mobile app in real-time.

After using the mobile app for a long time, farmers will know the exact amount and timing of watering for different types of crops. This is because they can edit or manipulate the length of the timer to get the best conditions for crop growth.

The mobile app also removes paperwork for recording gross sales of crops. Farmers can now record their sales after selling their crops at fairs or markets. The price will be recorded along with the market price so farmers will know if their sale was profitable by dividing the total weight of the crop by the profit they earned. Farmers can also view their sales graphs and see how their revenue is growing.

Finally, Mobile\_AFS also solves one of the biggest headaches for all farmers, their incomplete inventory system. Farmers can now see the quantity of items and the status of items in stock. Farmers can also track the prices of the items they buy to see if they are buying more than they have budgeted for the project.



### 3.8 Gantt Chart

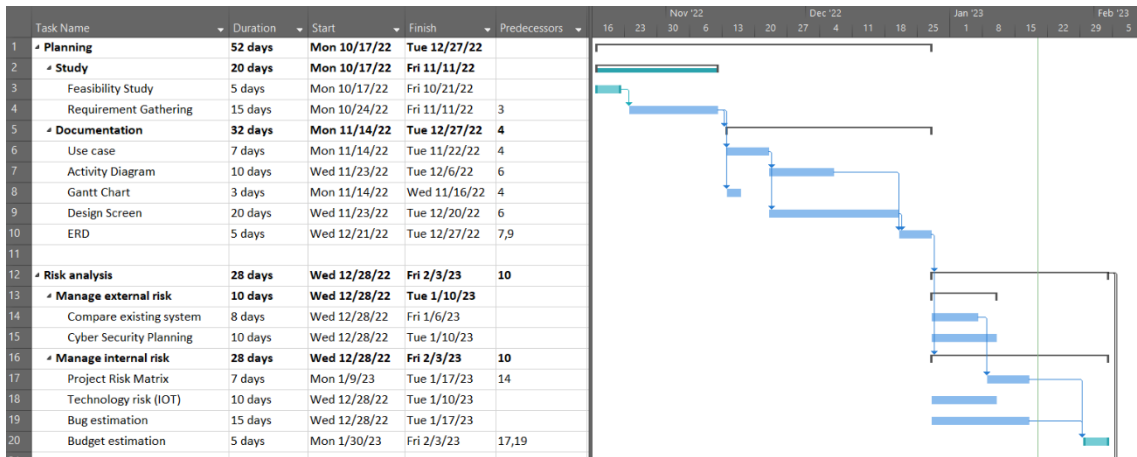


Figure 3.39 Gantt Chart 1

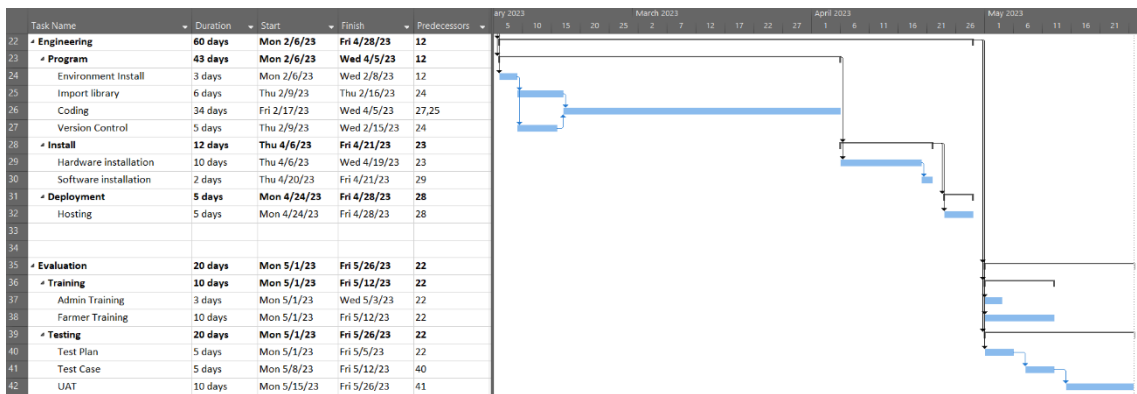


Figure 3.40 Gantt Chart 2

## **CHAPTER 4**

### **IMPLEMENTATION AND TESTING**

#### **4.1 Introduction**

Implementation and testing are the most important phase for developing a mobile application.

We all know that every mobile application can easily cause bugs. The reason occurring bugs is not because of the level of coding of developers but the complexity of the system. This chapter discusses the testing of the solution and implementation methodologies.

## 4.2 Input/ Output Design Implementation

Below show the screenshots of some input and example of output interfaces by Mobile\_AFS.

Below figure shows the Create Crops interface, which the farmer can add the crops. The user has to fill in all the textbox and the drop-down menu to create the new crops.

The figure displays two screenshots of the 'Create Crops' interface in the Mobile\_AFS application. Both screenshots show the top navigation bar with the time, signal strength, and battery level, along with the 'AGRONETICS' and 'TERAJU PENERAJU BUKHUPUTERA' logos.

The left screenshot shows the 'Create Crops' form with the following fields:

- Crops Title:** Eggplant 321
- Crops Size:** Medium
- Crops Type:** Eggplant (dropdown menu)
- Description:** This is the description of the eggplant
- Number of crops:** 50
- HST 1 / DAP 1:** 0 to 25
- Minimum water (ml):** 1000

The right screenshot shows the 'Create Crops' form with the following fields:

- HST 1 / DAP 1:** 0 to 25
- Minimum water (ml):** 1000
- Minimum EC (ml):** 1500
- Increment Interval (days):** 7
- Increment EC amount(ml):** 100
- HST 2 / DAP 2:** 26 to 50
- Minimum water (ml):** 500
- Minimum EC (ml):** 1000
- CREATE** button

Both screenshots show a bottom navigation bar with icons for Home, Crops, Sensor, Timer, and Menu.

Figure 4.1 Input design screen of Mobile\_AFS

In Figure 4.2, there are 2 screenshots as an example of the output design of the Mobile\_AFS. After the crops successfully created, the system will Toast the “Success: Crops Created” message to the farmer to let them know the crops success created. The farmer can also click on the specific crops and view the details in View Crops interface.

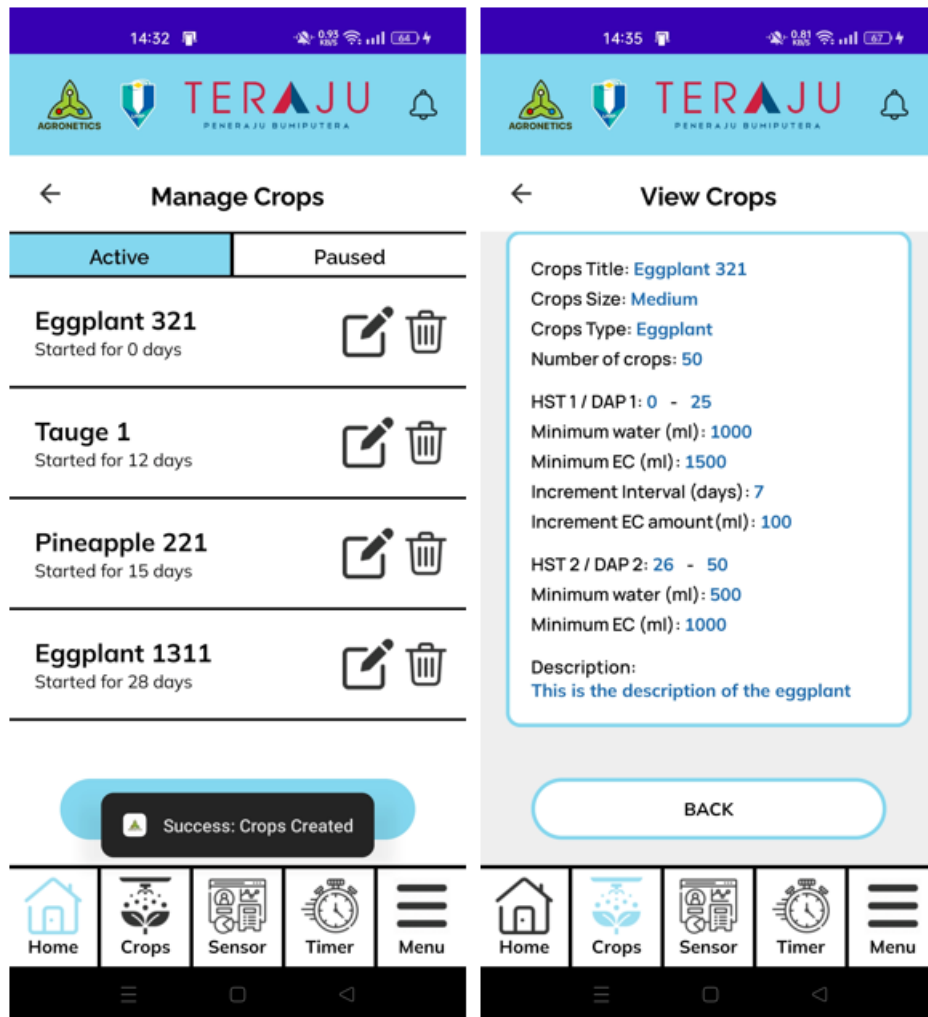
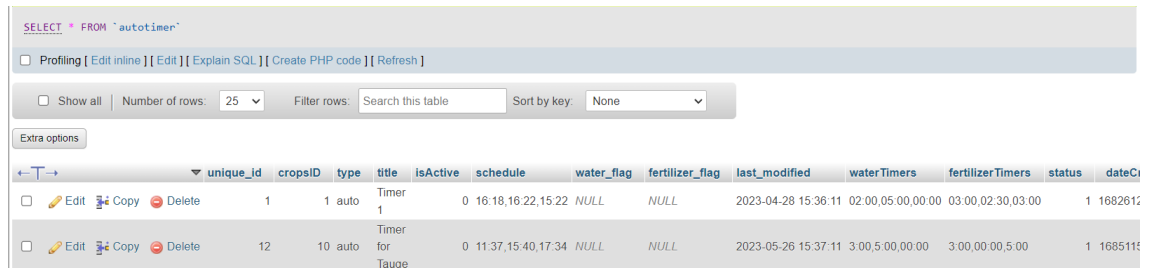


Figure 4.2 Output design screen of Mobile\_AFS

## 4.3 Database Implementation

### 4.3.1 Autotimer Table

The Autotimer table has primary key of unique\_id with the foreign key of cropsID. The other attributes are type, title, isActive, schedule, water\_flag, fertilizer\_flag, last\_modified, waterTimers, fertilizerTimers, status, dateCreated.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	unique_id	int(11)			No	None		AUTO_INCREMENT
2	cropsID	int(11)			No	None		
3	type	varchar(15)	utf8mb4_general_ci		No	auto		
4	title	varchar(50)	utf8mb4_general_ci		No	None		
5	isActive	tinyint(1)			No	0		
6	schedule	text	utf8mb4_general_ci		Yes	NULL		
7	water_flag	text	utf8mb4_general_ci		Yes	NULL		
8	fertilizer_flag	text	utf8mb4_general_ci		Yes	NULL		
9	last_modified	datetime			Yes	NULL		
10	waterTimers	varchar(100)	utf8mb4_general_ci		No	0,0,0,0,0,0,0,0		
11	fertilizerTimers	varchar(100)	utf8mb4_general_ci		No	0,0,0,0,0,0,0,0		
12	status	tinyint(1)			No	1		
13	dateCreated	bigint(20)			No	None		

Figure 4.3 Database of Autotimer table

### 4.3.2 Controltest Table

The Controltest table has primary key of controltestID with the foreign key of farmID. The other attributes are timestamp, water, fertilizer, moisture, ec and temperature.

`SELECT * FROM `controltest``

Profiling [ [Edit inline](#) ] [ [Edit](#) ] [ [Explain SQL](#) ] [ [Create PHP code](#) ] [ [Refresh](#) ]

Show all | Number of rows:  Filter rows:  Sort by key:

Extra options

	controltestID	farmID	timestamp	water	fertilizer	moisture	ec	temperature
<input type="checkbox"/> Edit Copy Delete	1	1	1678438421	EMPTY	EMPTY	65	1636	20
<input type="checkbox"/> Edit Copy Delete	2	1	1678438426	MIN	EMPTY	58	1307	23
<input type="checkbox"/> Edit Copy Delete	3	1	1678438431	MIN	MIN	52	1284	20
<input type="checkbox"/> Edit Copy Delete	4	1	1678438436	MAX	MIN	50	1684	25

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	<b>controltestID</b>	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/> 2	<b>farmID</b>	int(11)			No	None		
<input type="checkbox"/> 3	<b>timestamp</b>	int(15)			No	None		
<input type="checkbox"/> 4	<b>water</b>	varchar(10)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 5	<b>fertilizer</b>	varchar(10)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 6	<b>moisture</b>	int(11)			No	None		
<input type="checkbox"/> 7	<b>ec</b>	int(11)			No	None		
<input type="checkbox"/> 8	<b>temperature</b>	float			No	None		

Figure 4.4 Database of Controltest table

### 4.3.3 Crops Table

The Crops table has primary key of cropsID with the foreign key of farmID. The other attributes are title, size, type, description, cropsNum, HST1Start, HST1End, minWater1, minEC1, incrementInterval, incrementAmt, HST2Start, HST2End, minWater2, minEC2, good, bad, badReason, dateCreated and status.

The screenshot shows a database management interface with the 'crops' table selected. The top part displays a list of rows with columns: cropsID, farmID, title, size, type, description, cropsNum, HST1Start, HST1End, minWater1, minEC1, incrementInterval, incrementAmt, and H. Below this, a schema table lists the attributes and their properties.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	cropsID	int(11)			No	None		AUTO_INCREMENT
2	farmID	int(11)			No	None		
3	title	varchar(50)	utf8mb4_general_ci		No	None		
4	size	varchar(15)	utf8mb4_general_ci		No	None		
5	type	varchar(15)	utf8mb4_general_ci		No	None		
6	description	varchar(255)	utf8mb4_general_ci		No	None		
7	cropsNum	int(11)			No	None		
8	HST1Start	int(11)			No	None		
9	HST1End	int(11)			No	None		
10	minWater1	int(11)			No	None		
11	minEC1	int(11)			No	None		
12	incrementInterval	int(11)			No	None		
13	incrementAmt	int(11)			No	None		
14	HST2Start	int(11)			No	None		
15	HST2End	int(11)			No	None		
16	minWater2	int(11)			No	None		
17	minEC2	int(11)			No	None		
18	good	int(11)			No	None		
19	bad	int(11)			No	None		
20	badReason	varchar(255)	utf8mb4_general_ci		Yes	NULL		
21	dateCreated	bigint(20)			No	None		

Figure 4.5 Database of Crops table

### 4.3.4 Farm Table

The Farm table has primary key of farmID with the foreign key of farmerID. The other attributes are title and address.

The screenshot shows a database management interface. At the top, there is a SQL query editor with the text `SELECT * FROM `farm``. Below the editor are several control buttons: `Profiling`, `Edit inline`, `Edit`, `Explain SQL`, `Create PHP code`, and `Refresh`. A control bar includes `Show all`, `Number of rows: 25` (with a dropdown arrow), and `Filter rows: Search this table` (with an input field). An `Extra options` button is also present. The main area displays a table with columns `farmID`, `farmerID`, `title`, and `address`. A single row is visible with values `1`, `1`, `Farm 1`, and `3, Lorong Taman Orkid, 26600 Pekan, Pahang, Malays...`. Below the table is a detailed schema view with columns: `#`, `Name`, `Type`, `Collation`, `Attributes`, `Null`, `Default`, `Comments`, and `Extra`. The schema details are as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	farmID	int(11)			No	None		AUTO_INCREMENT
2	farmerID	int(11)			No	None		
3	title	varchar(30)	utf8mb4_general_ci		No	None		
4	address	varchar(255)	utf8mb4_general_ci		No	None		

Figure 4.6 Database of Farm table



### 4.3.5 Farmers Table

The Farmers table has primary key of id. The other attributes are password, fmname, phone, location and reset\_token.

The screenshot shows a database management interface for the 'Farmers' table. It displays a list of two records and a detailed view of the table's schema.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b>	int(11)			No	None		AUTO_INCREMENT
2	<b>email</b>	varchar(255)	utf8mb4_general_ci		No	None		
3	<b>password</b>	varchar(20)	utf8mb4_general_ci		No	None		
4	<b>fmname</b>	varchar(255)	utf8mb4_general_ci		No	None		
5	<b>phone</b>	varchar(13)	utf8mb4_general_ci		No	None		
6	<b>location</b>	varchar(255)	utf8mb4_general_ci		No	None		
7	<b>reset_token</b>	text	utf8mb4_general_ci		Yes	NULL		

Figure 4.7 Database of Farmers table

### 4.3.6 Inventorys Table

The Inventorys table has primary key of inventoryID with the foreign key of purchaseID. The other attributes are inventoryName, inventoryPIC, quantity, price, dateStored, status and lastModified.

`SELECT * FROM `inventorys``

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	inventoryID	purchaseID	inventoryName	inventoryPIC	quantity	price	dateStored	status	lastModified
<input type="checkbox"/> Edit Copy Delete	2	1	Raticide	Aden Jackson	32	350.00	2023-06-02	In Stock	2023-06-03
<input type="checkbox"/> Edit Copy Delete	3	1	Raticide	John	20	100.00	2023-06-03	In Stock	2023-06-03
<input type="checkbox"/> Edit Copy Delete	4	1	Bird Pesticide	James Holmes	20	25.00	2023-06-03	In Stock	2023-06-03
<input type="checkbox"/> Edit Copy Delete	5	3	Raticide	Ali bin Ahmad	30	600.00	2023-06-03	In Stock	2023-06-03

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	<b>inventoryID</b>	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/> 2	<b>purchaseID</b>	int(11)			No	None		
<input type="checkbox"/> 3	<b>inventoryName</b>	varchar(50)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 4	<b>inventoryPIC</b>	varchar(255)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 5	<b>quantity</b>	int(11)			No	None		
<input type="checkbox"/> 6	<b>price</b>	double(8,2)			No	None		
<input type="checkbox"/> 7	<b>dateStored</b>	date			No	None		
<input type="checkbox"/> 8	<b>status</b>	varchar(15)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 9	<b>lastModified</b>	date			No	None		

Figure 4.8 Database of Inventorys table

### 4.3.7 Pesticide Table

The Pesticide table has primary key of pesticideID with the foreign key of cropsID. The other attributes are pesticideTitle, pesticideType, pestType, fungusVolume, nVolume, eVolume, gVolume, description, days, schedule, timer, status and dateCreated.

The screenshot shows a database management interface with the following components:

- Query Editor:** A text area containing the SQL query: `SELECT * FROM `pesticide``. Below it are links for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh.
- Table Controls:** A row of controls including 'Show all', 'Number of rows: 25', 'Filter rows: Search this table', and 'Sort by key: None'.
- Table Data:** A table with columns: pesticideID, cropsID, pesticideTitle, pesticideType, pestType, fungusVolume, nVolume, eVolume, gVolume, description, days, and schedule. It contains two rows of data:
 

pesticideID	cropsID	pesticideTitle	pesticideType	pestType	fungusVolume	nVolume	eVolume	gVolume	description	days	schedule
9	1	Pesticide for Eggplant 1	Organic	Fungus	NULL	NULL	NULL	NULL	This is the fungus type eggplant	Monday, Wednesday-Monday, Wednesday, Thursday, Fri...	19 05,11:33
10	9	Pineapple 1	Chemical	Insects	NULL	200	150	150	This is the pineapple description	Tuesday, Thursday-Thursday, Saturday	12 24,14:00
- Table Schema:** A table listing the columns and their properties:
 

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	pesticideID	int(11)			No	None		AUTO_INCREMENT
2	cropsID	int(11)			No	None		
3	pesticideTitle	varchar(50)	utf8mb4_general_ci		No	None		
4	pesticideType	varchar(10)	utf8mb4_general_ci		No	None		
5	pestType	varchar(10)	utf8mb4_general_ci		No	None		
6	fungusVolume	double			Yes	NULL		
7	nVolume	double			Yes	NULL		
8	eVolume	double			Yes	NULL		
9	gVolume	double			Yes	NULL		
10	description	varchar(255)	utf8mb4_general_ci		No	None		
11	days	varchar(255)	utf8mb4_general_ci		No	None		
12	schedule	varchar(255)	utf8mb4_general_ci		No	None		
13	timer	varchar(255)	utf8mb4_general_ci		No	None		
14	status	tinyint(1)			No	1		
15	dateCreated	bigint(20)			No	None		

Figure 4.9 Database of Pesticide table

### 4.3.8 Project Table

The Project table has primary key of projectID with the foreign key of farmID. The other attributes are title, budget, startDate, endDate and status.

The screenshot displays a database management tool interface. At the top, there is a SQL query editor with the text `SELECT * FROM `project``. Below the editor are several control buttons:  Profiling, [\[ Edit inline \]](#), [\[ Edit \]](#), [\[ Explain SQL \]](#), [\[ Create PHP code \]](#), and [\[ Refresh \]](#). A toolbar below contains  Show all, Number of rows: 25 (dropdown), Filter rows: Search this table (input), and Sort by key: None (dropdown). An 'Extra options' button is also present.

The main data table has the following columns: projectID, farmID, title, budget, startDate, endDate, and status. It contains two rows of data:

	projectID	farmID	title	budget	startDate	endDate	status
<input type="checkbox"/>	1	1	Project Carrot	5500.00	2023-05-29	2023-08-31	1
<input type="checkbox"/>	2	1	Eggplant Rich	5300.00	2023-05-10	2023-08-31	1

Below the data table is a table structure overview with the following columns: #, Name, Type, Collation, Attributes, Null, Default, Comments, and Extra.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	projectID	int(11)			No	None		AUTO_INCREMENT
2	farmID	int(11)			No	None		
3	title	varchar(100)	utf8mb4_general_ci		No	None		
4	budget	double(8,2)			No	None		
5	startDate	date			No	None		
6	endDate	date			No	None		
7	status	tinyint(1)			No	1		

Figure 4.10 Database of Project table

### 4.3.9 Purchase Table

The Purchase table has primary key of purchaseID with the foreign key of projectID. The other attributes are title, projectPIC, price, date and imageUri.

SELECT \* FROM `purchase`

Profiling [ [Edit inline](#) ] [ [Edit](#) ] [ [Explain SQL](#) ] [ [Create PHP code](#) ] [ [Refresh](#) ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	purchaseID	projectID	title	projectPIC	price	date	imageUri
<input type="checkbox"/> Edit Copy Delete	1	1	Purchase pesticide	James Holmes	500.39	2023-05-28	receipt2_1685637160516.png
<input type="checkbox"/> Edit Copy Delete	2	1	Gear Purchase for Carrot	James Charles	265.30	2023-06-02	receipt4_1685719895634.jpg
<input type="checkbox"/> Edit Copy Delete	3	2	Purchase in Mr.DIY	Ali bin Ahmad	600.00	2023-06-03	receipt3_1685804484153.png

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	purchaseID	int(11)			No	None		AUTO_INCREMENT
2	projectID	int(11)			No	None		
3	title	varchar(100)	utf8mb4_general_ci		No	None		
4	projectPIC	varchar(255)	utf8mb4_general_ci		No	None		
5	price	double(8,2)			No	None		
6	date	date			No	None		
7	imageUri	varchar(255)	utf8mb4_general_ci		No	None		

Figure 4.11 Database of Purchase table

### 4.3.10 Sales Table

The Sales table has primary key of salesID with the foreign key of farmID. The other attributes are salesTitle, salesGrade, cropsType, salesWeight, salesPrice, marketPrice and salesDate.

SELECT \* FROM `sales`

Profiling [\[ Edit inline \]](#) [\[ Edit \]](#) [\[ Explain SQL \]](#) [\[ Create PHP code \]](#) [\[ Refresh \]](#)

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	salesID	farmID	salesTitle	salesGrade	cropsType	salesWeight	salesPrice	marketPrice	salesDate
<input type="checkbox"/> Edit Copy Delete	1	1	Carrot Sales 1	B	Carrot	350.12	5008.80	120.00	2023-04-19
<input type="checkbox"/> Edit Copy Delete	2	1	Carrot Sales 2	A	Carrot	356.00	348.00	20.00	2023-04-19
<input type="checkbox"/> Edit Copy Delete	3	1	Polato Sales 1	A	Polato	721.32	48217.50	100.00	2023-04-29
<input type="checkbox"/> Edit Copy Delete	4	1	Chili Sales 1	B	Chili	56236.35	999999.99	200.00	2023-04-29
<input type="checkbox"/> Edit Copy Delete	7	1	Eggplant 131 - Jan Sales	A	Eggplant	561.20	15221.50	45.30	2023-05-15
<input type="checkbox"/> Edit Copy Delete	8	1	Pineapple Sales - May	B	Pineapple	150.00	2005.30	4.45	2023-05-23

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	salesID	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/> 2	farmID	int(11)			No	None		
<input type="checkbox"/> 3	salesTitle	varchar(50)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 4	salesGrade	varchar(5)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 5	cropsType	varchar(15)	utf8mb4_general_ci		No	None		
<input type="checkbox"/> 6	salesWeight	double(7,2)			No	None		
<input type="checkbox"/> 7	salesPrice	double(8,2)			No	None		
<input type="checkbox"/> 8	marketPrice	double(8,2)			No	None		
<input type="checkbox"/> 9	salesDate	date			No	None		

Figure 4.12 Database of Sales table

### 4.3.11 Sales\_history Table

The Sales\_history table has primary key of historyID with the foreign key of salesID. The other attributes are title, description, status and rejectReason.

The screenshot shows a database management interface with the following components:

- SQL Query:** `SELECT * FROM `sales_history``
- Table Data:**

historyID	adminID	salesID	title	description	status	rejectReason
1	1	3	Change sales title for salesID = 2	Change salesID=2 title from "Hhd" to "Carrot Sale..."	1	NULL
2	1	1	Request for changing price	Change price to RM1000.00	0	Not enough evidence to prove
- Table Structure:**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	historyID	int(11)			No	None		AUTO_INCREMENT
2	adminID	int(11)			No	None		
3	salesID	int(11)			No	None		
4	title	varchar(50)	utf8mb4_general_ci		No	None		
5	description	varchar(255)	utf8mb4_general_ci		No	None		
6	status	tinyint(1)			No	None		
7	rejectReason	varchar(255)	utf8mb4_general_ci		Yes	NULL		

Figure 4.13 Database of Sales\_history table

## 4.4 User manual

### 4.4.1 Module Login

Farmer need to enter the correct email and password then click on the “Login” button to navigate to the Main interface.

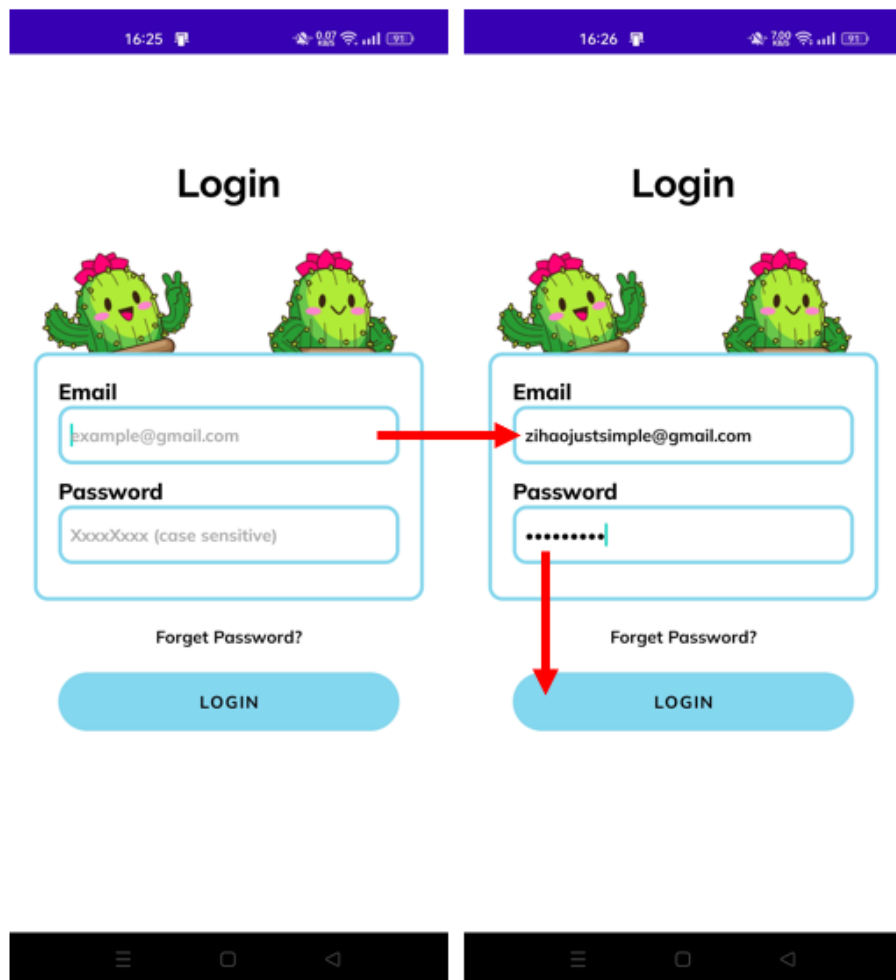


Figure 4.14 Login User Manual of Mobile\_AFS



Farmer need to enter the correct email then click on the “Reset Password” button to receive the reset password email from server. The farmer can open their email and click on the reset password link to reset their password in the browser.

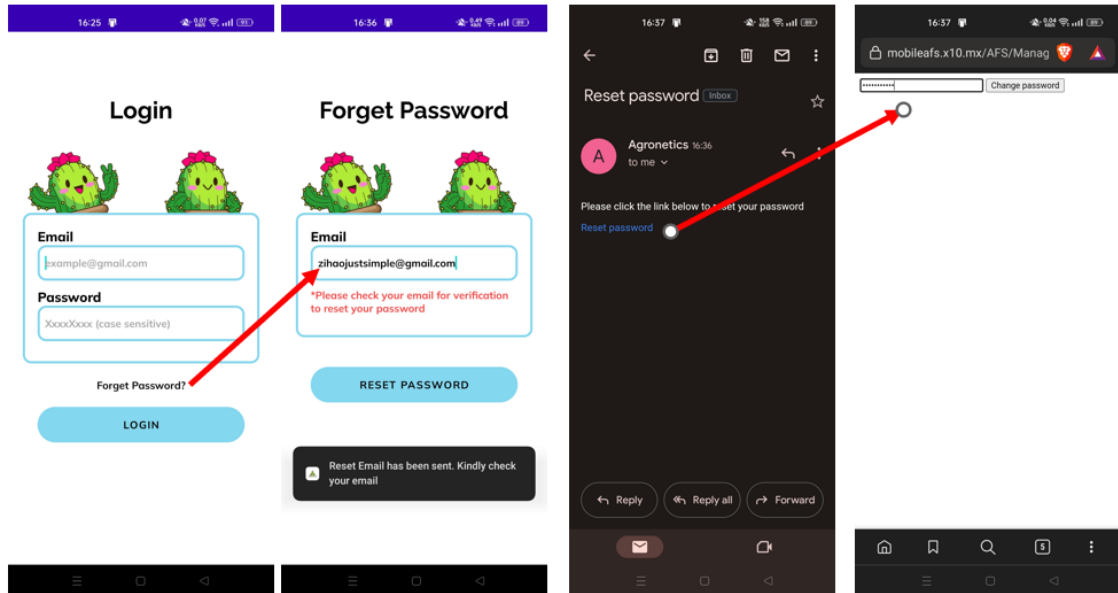


Figure 4.15 Forget Password User Manual of Mobile\_AFS

#### 4.4.2 Main

After farmer login, the main interface contains a lot of navigation to another module. At the bottom, we have the Manage Crops, Sensor, Timer, and Menu. The functions contain navigation to the Manage Pesticide, Sales, Schedule, Project, Purchase, and Inventory.

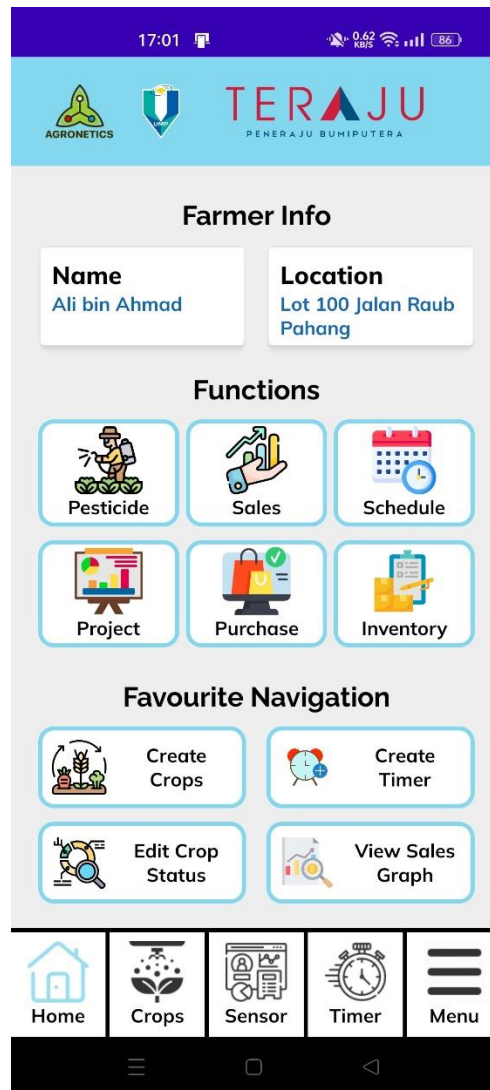


Figure 4.16 Main Interface User Manual of Mobile\_AFS

### 4.4.3 Module Profile

Farmer need to click on the menu on the bottom navigation at the Main interface. User can click the “Profile” to navigate to the Profile interface. The user can update any data in the textbox and click the update button. A success update Toast will appear if the data is updated at the database.

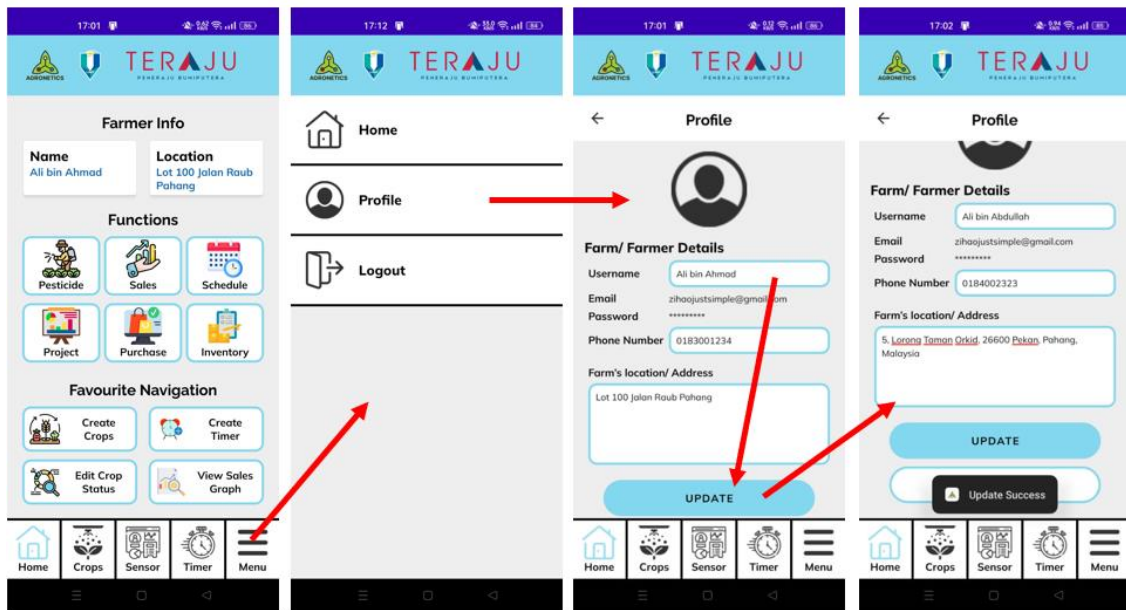


Figure 4.17 Edit Profile User Manual of Mobile\_AFS

#### 4.4.4 Module Crops

Farmer need to click on the “Crops” button to navigate to the main menu of crops. The farmer must press the “Manage Crops” button to open the Manage Crops menu interface.

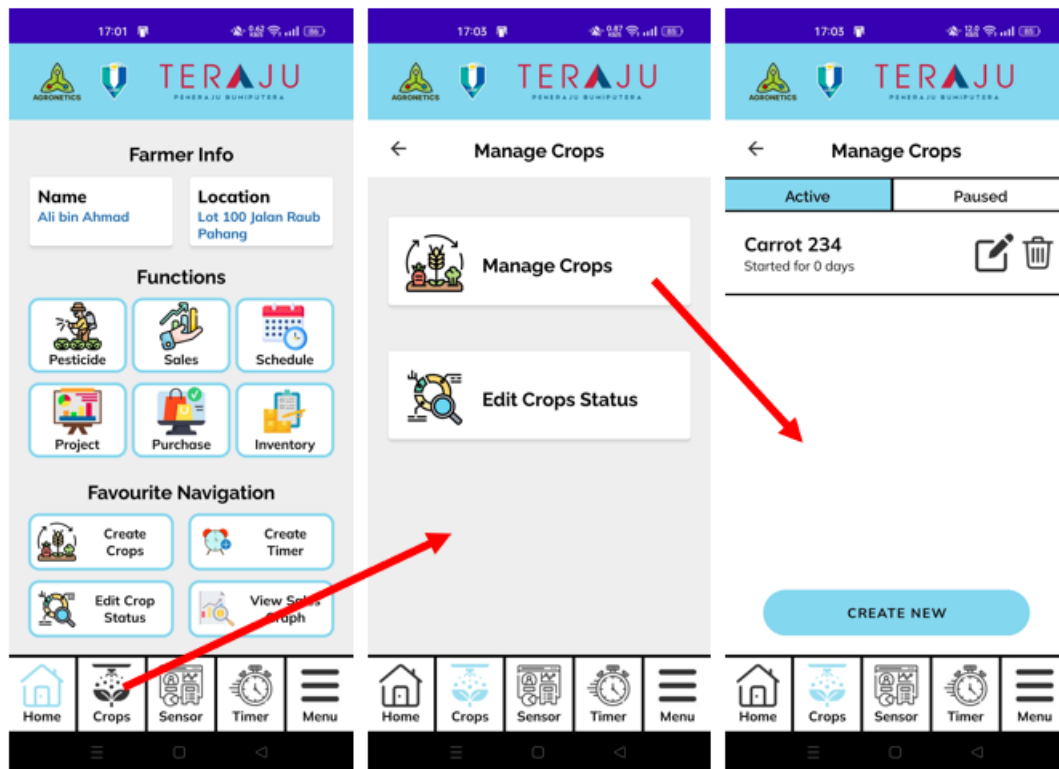


Figure 4.18 Manage Crops Menu User Manual of Mobile\_AFS

Farmer need to click on the “CREATE NEW” button to navigate to the Add Crops interface. Farmer must enter all the info needed for the crops to create the new crops. The created crops will appear at the menu.



Figure 4.19 Add Crops User Manual of Mobile\_AFS

Farmer need to click on the “EDIT” button to navigate to the Edit Crops interface. Farmer can enter the updated info to update the crops. The updated crops will appear at the menu.

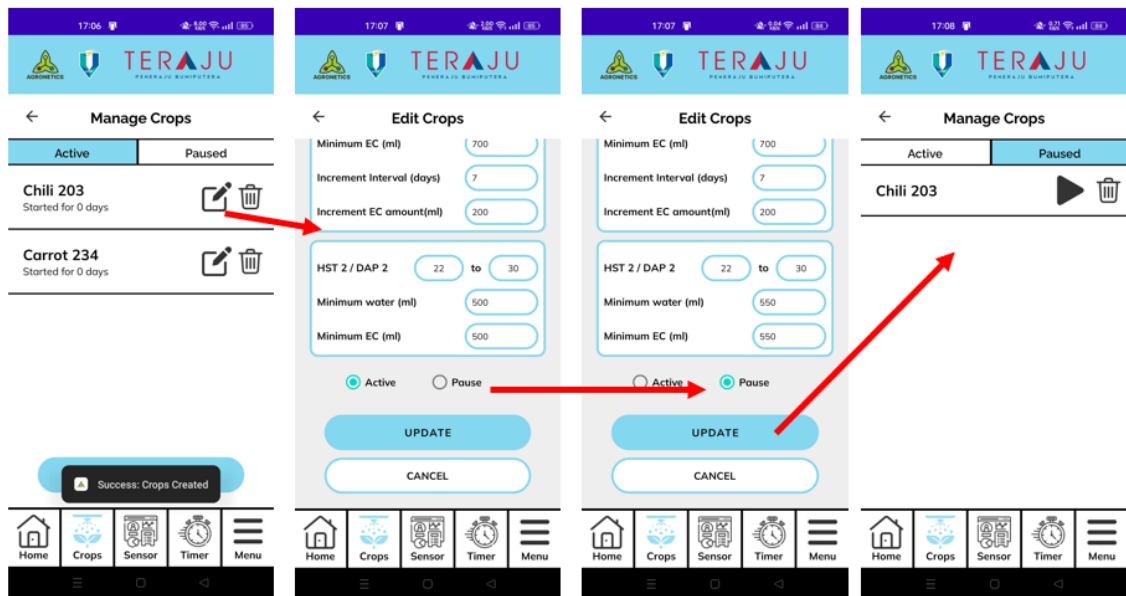


Figure 4.20 Edit Crops User Manual of Mobile\_AFS

Farmer need to click on the “VIEW button to navigate to the View Crops interface. Farmer can view the details of the crops.

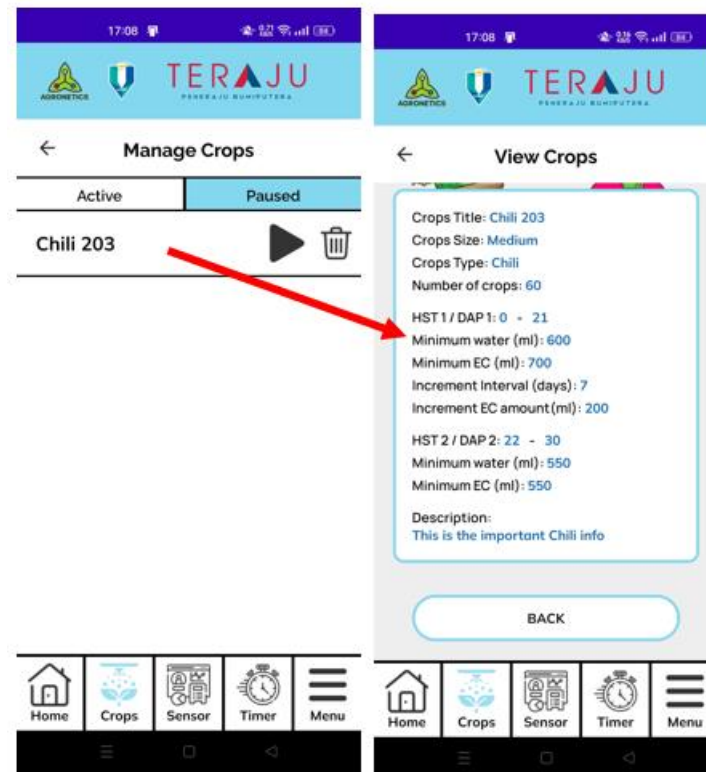


Figure 4.21 View Crops User Manual of Mobile\_AFS

Farmer need to click on the “DELETE button to navigate to delete the crops from the database.

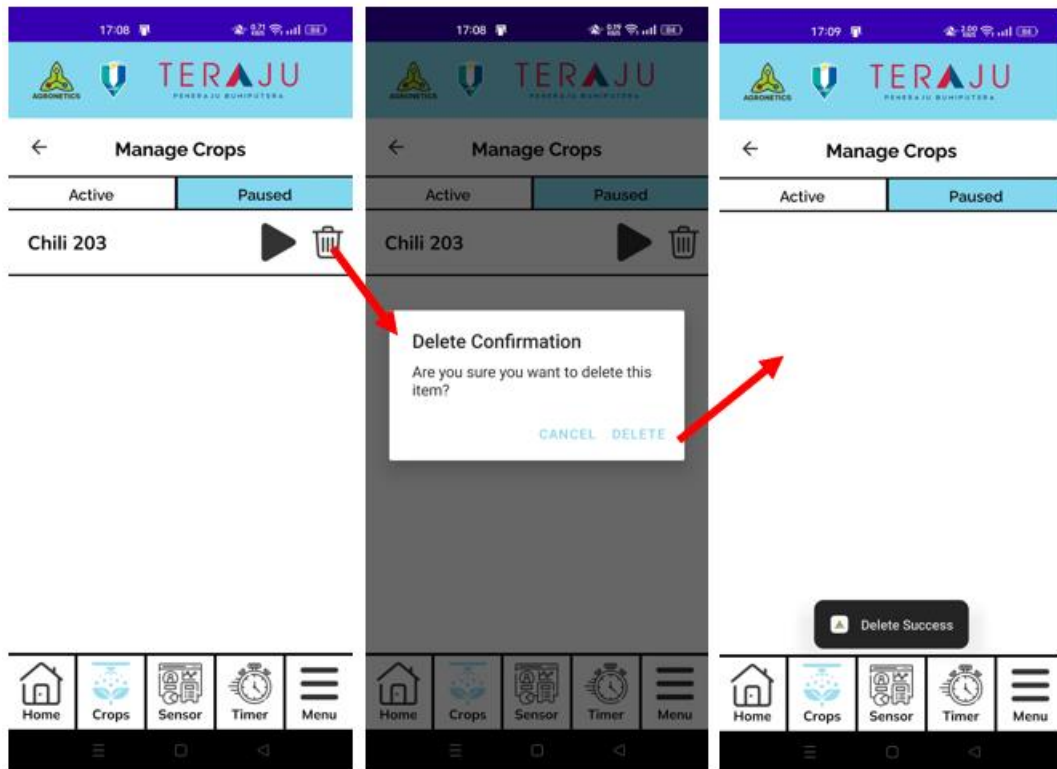


Figure 4.22 Delete Crops User Manual of Mobile\_AFS



Farmer need to click on the “Edit Crops Status” button to navigate to Edit Crops Status Menu interface.

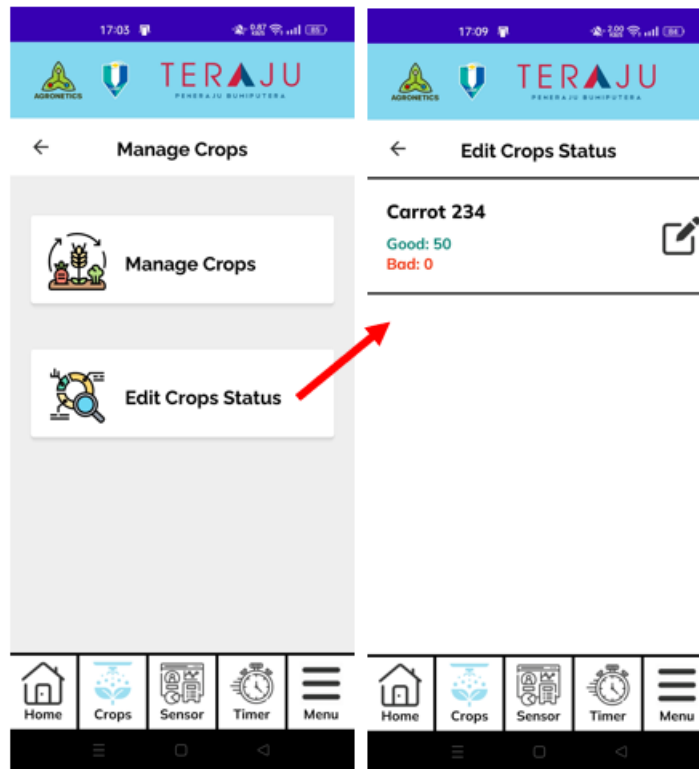


Figure 4.23 Edit Crops Status Menu User Manual of Mobile\_AFS

Farmer need to click on the “Edit” button to navigate to Edit Crops Status interface. The farmer can drag the status bar or directly enter the value in the textbox. After all the details is fill, farmer need to click the update button to update the data to database. The updated crops will display at the menu.

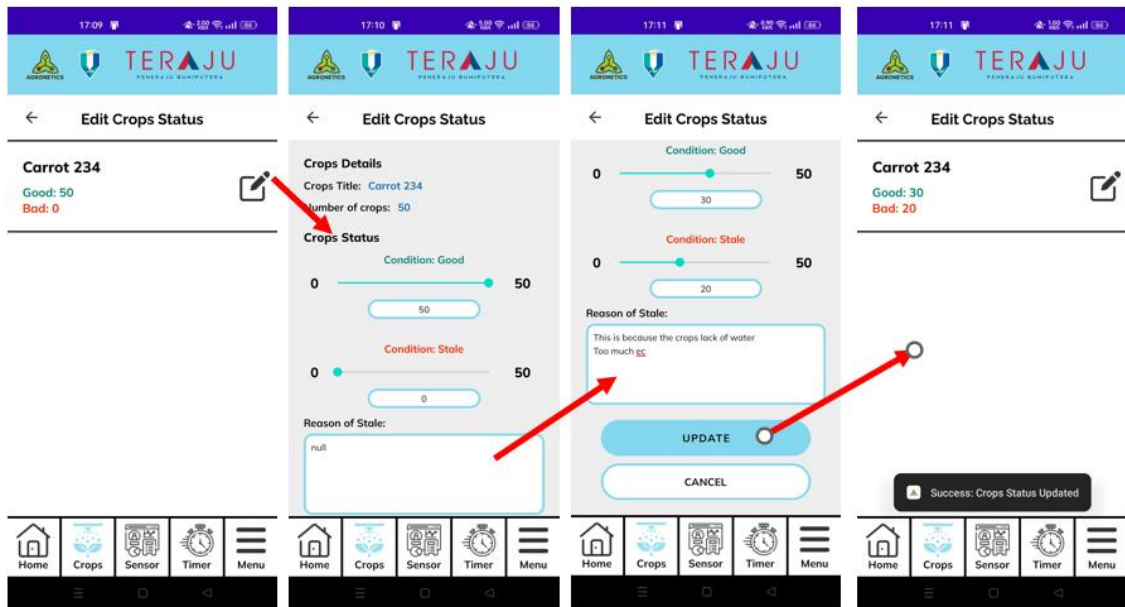


Figure 4.24 Edit Crops Status User Manual of Mobile\_AFS

Farmer need to click on the empty space in the specific crops to navigate to View Crops Status interface. The interface will display all the details about the crop's status.

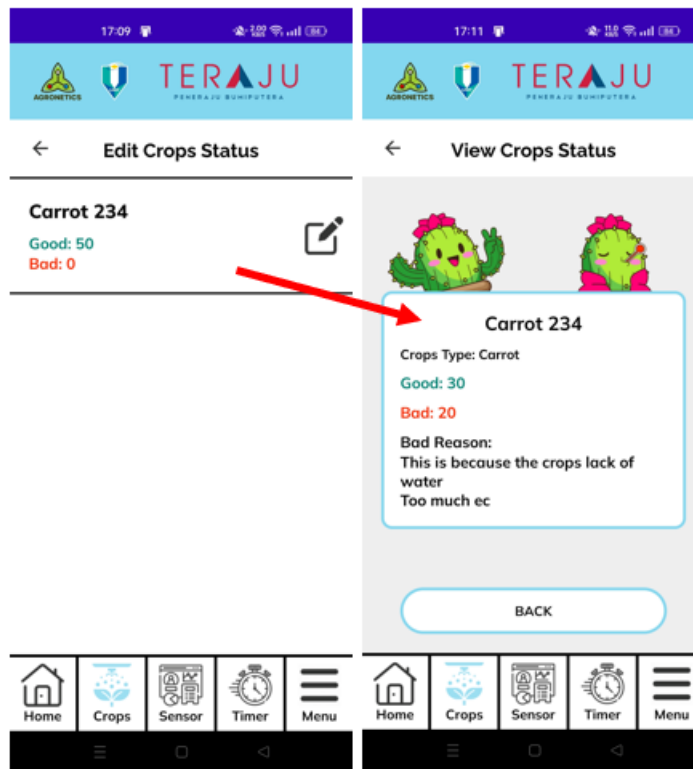


Figure 4.25 View Crops Status User Manual of Mobile\_AFS

#### 4.4.5 Module Sensor

Farmer need to click on the” Sensor” to navigate to Sensor interface. All the details of the sensor will display to the farmer.

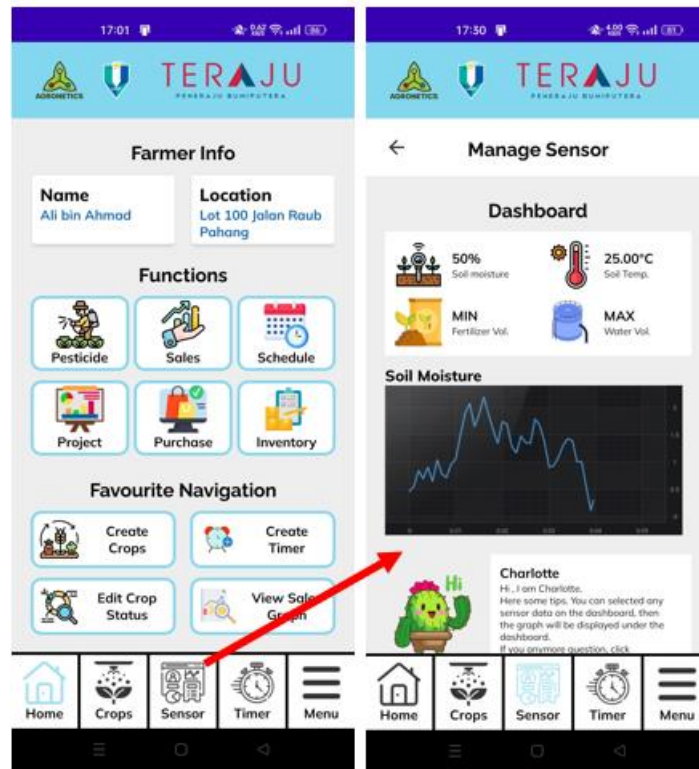


Figure 4.26 Manage Sensor User Manual of Mobile\_AFS

#### 4.4.6 Module Timer

Farmer need to click on the” Timer” to navigate to Manage Timer Menu interface.

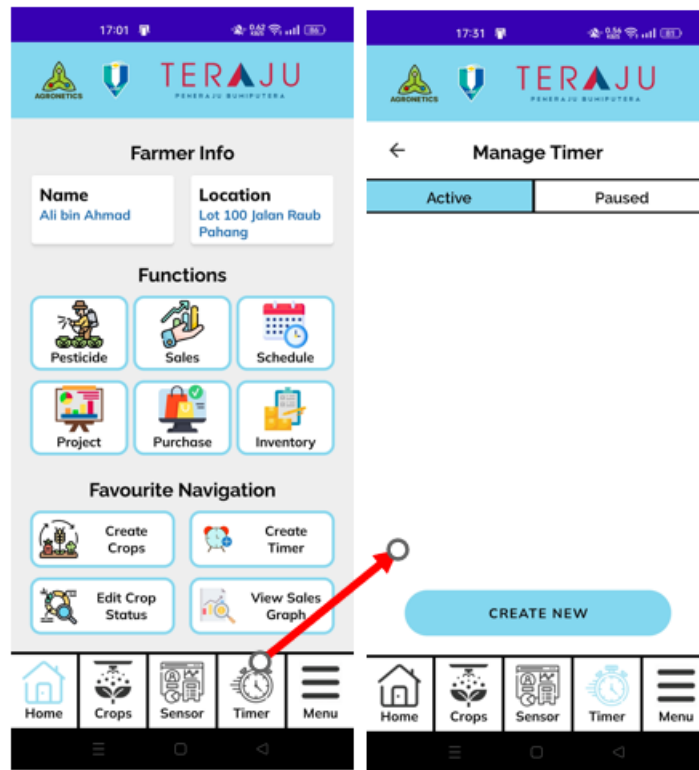


Figure 4.27 Manage Timer Menu User Manual of Mobile\_AFS

The farmer needs to choose the crops title (Crops) from the drop-down box to connect with the timer.



Figure 4.28 Add Timer 1 User Manual of Mobile\_AFS

The farmer needs to choose to tick the checkbox. In example, we checked the fertilizer timer, which only will set timer for fertilizer only. The timer should have time and duration of irrigate. The farmer will click the ADD button after filling the details.

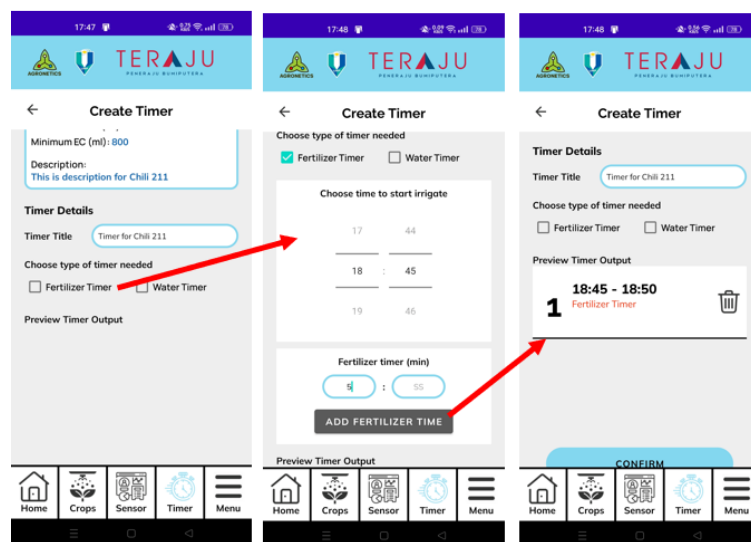


Figure 4.29 Add Timer 2 User Manual of Mobile\_AFS

In 2<sup>nd</sup> example, we checked the water timer, which only will set timer for water only. The farmer also needs to click the ADD button after filling the details.

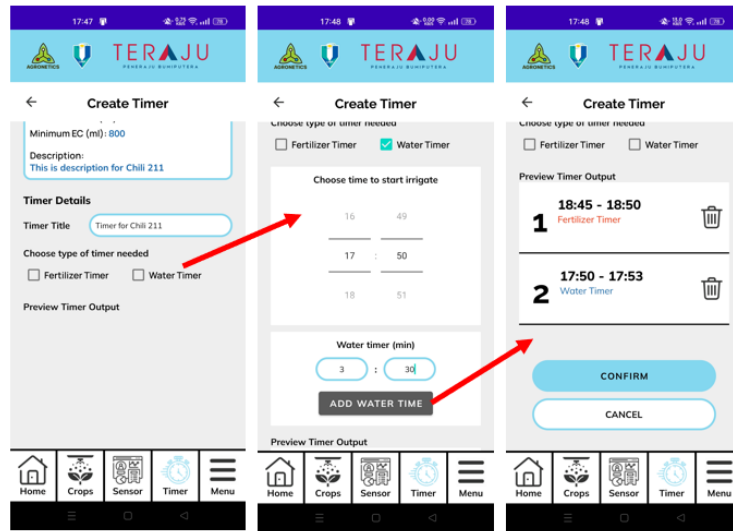


Figure 4.30 Add Timer 3 User Manual of Mobile\_AFS

In 3<sup>rd</sup> example, we checked the both the fertilizer and water timer. The farmer can proceed to save all the timer data by click on the “Confirm” button.

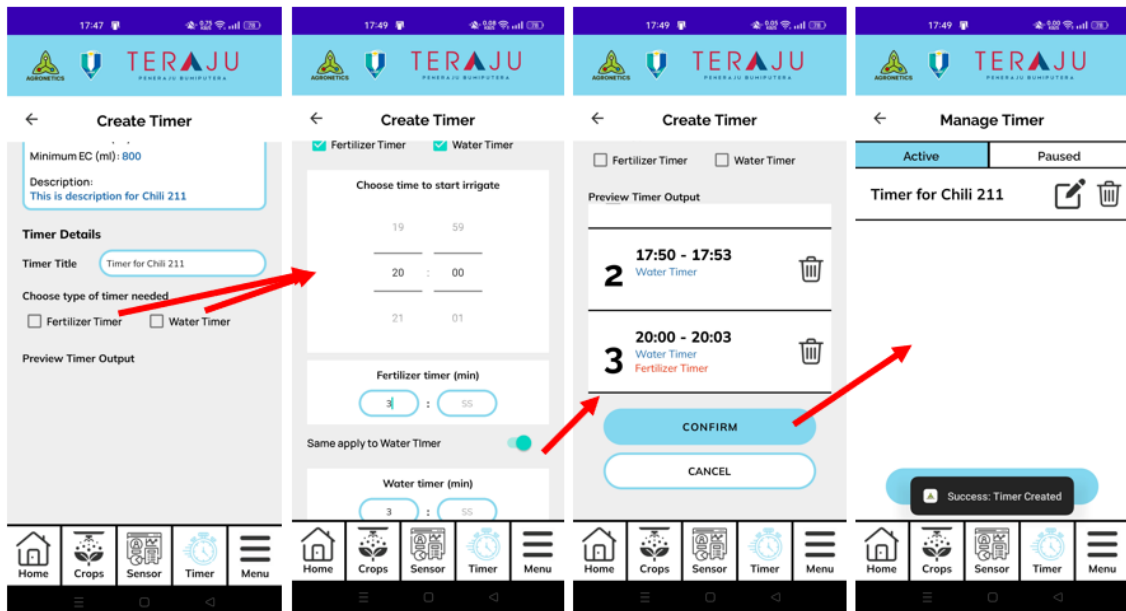


Figure 4.31 Add Timer 4 User Manual of Mobile\_AFS

Farmer need to click on the” Edit” button to navigate to Edit Timer interface. In this interface, farmer has the option to update every timer details. For example, we edit the timer output by delete it. We can see the timer is deleted in on the third screen.

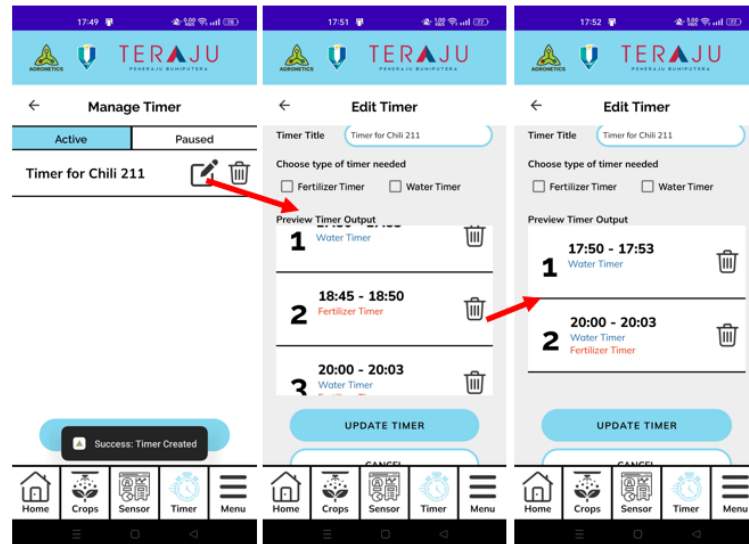


Figure 4.32 Edit Timer 1 User Manual of Mobile\_AFS

In the example, we add new timer output. The farmer can click on the “Update Timer” button to update the latest timer details to the database.

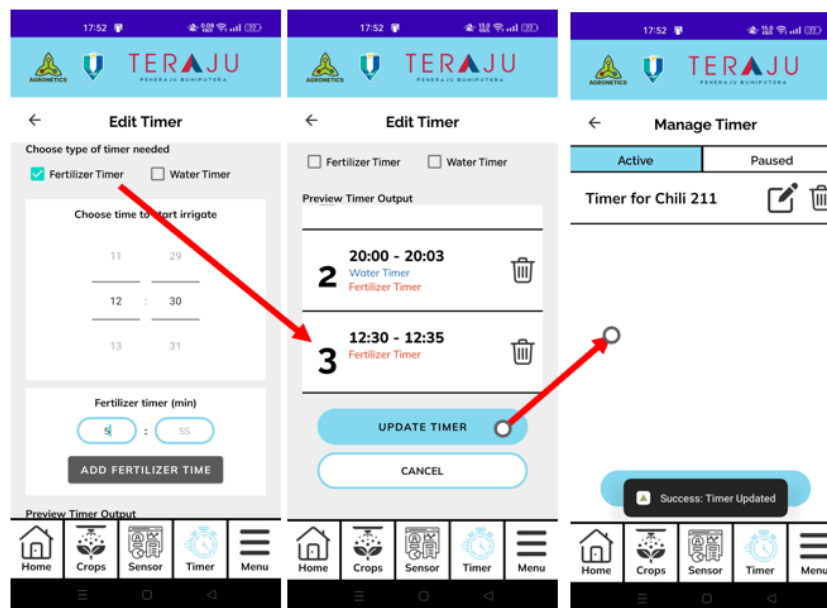


Figure 4.33 Edit Timer 2 User Manual of Mobile\_AFS



Farmer need to click on the empty white space of the specific timer to navigate to View Timer interface. All the fertilizer and water timer details will be displayed.



Figure 4.34 View Timer User Manual of Mobile\_AFS

#### 4.4.7 Module Pesticide

Farmer need to click on the” Pesticide” button to navigate to Manage Pesticide Menu interface.

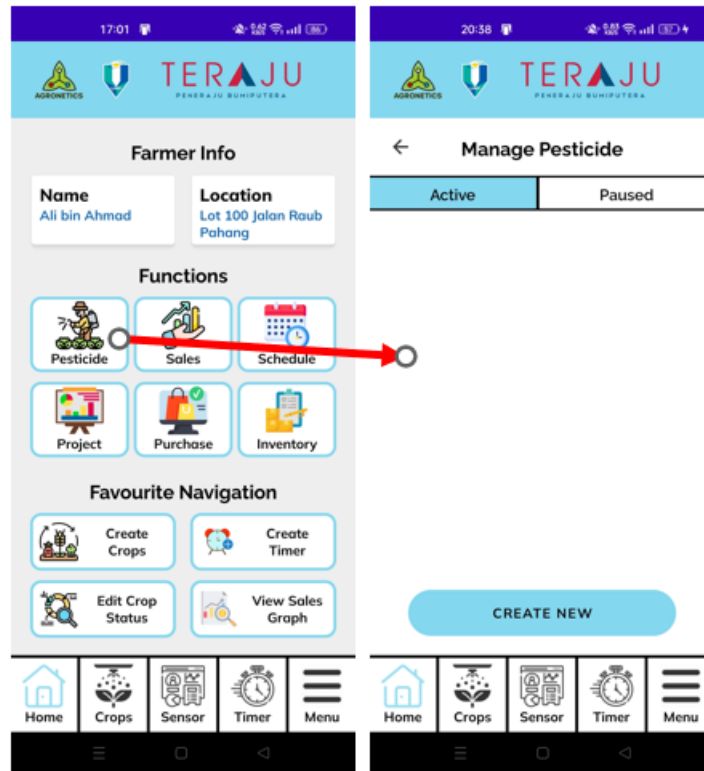


Figure 4.35 Manage Pesticide Menu User Manual of Mobile\_AFS

Farmer need to click on the” Create New” button to navigate to Add Pesticide interface. Farmer need to choose the Crops that need to spray the pesticide. The farmer needs to fill in the pesticide details by insert all the textbox and choose the pesticide and pest type using the radio button.

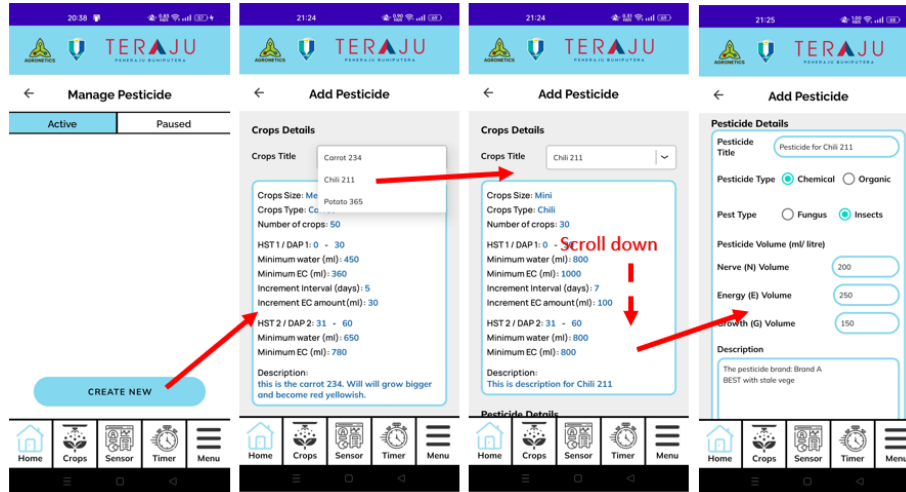


Figure 4.36 Add Pesticide 1 User Manual of Mobile\_AFS

In the example, we choose Monday (Mon), then farmer need to choose the schedule time and duration to spray the pesticide. Then the farmer needs to click the “Add pesticide” button to add the timer to the timer output.

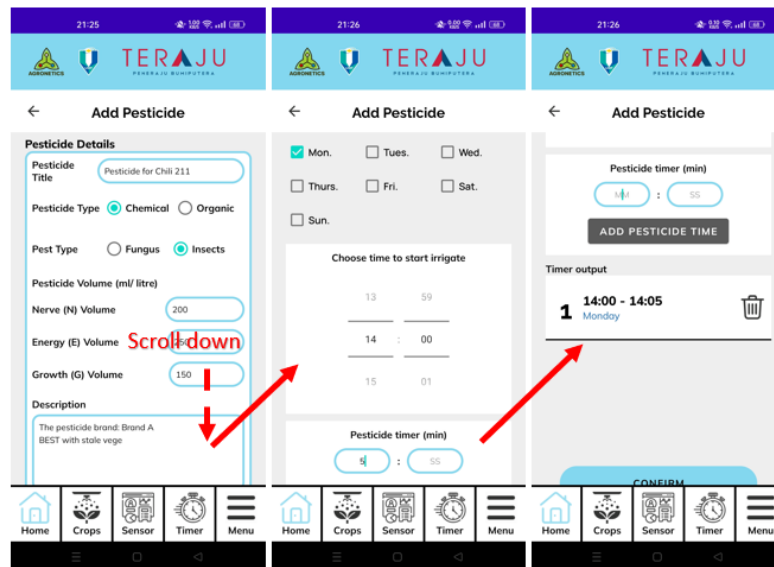


Figure 4.37 Add Pesticide 2 User Manual of Mobile\_AFS

In the example, we choose two days which are Wednesday (Wed) and Friday (Fri). The farmer needs to fill the date and duration of the timer also. After all the data is filled, the pesticide list also will be displayed.

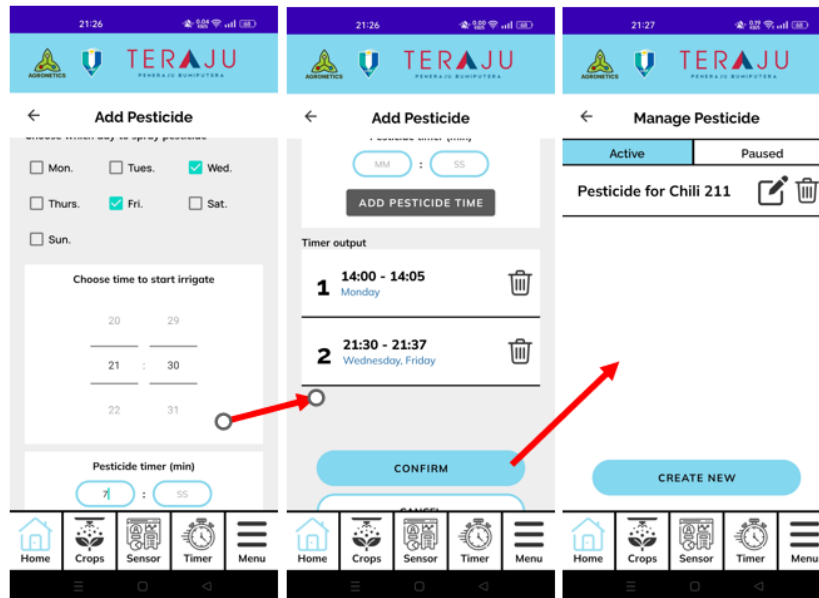


Figure 4.38 Add Pesticide 3 User Manual of Mobile\_AFS

The farmer needs to “Edit” button to navigate to the Edit Pesticide interface. After the farmer fill all the updated pesticide details, he need to press the “Update” button to updated the details to the database. The success update message will prompt to the farmer.

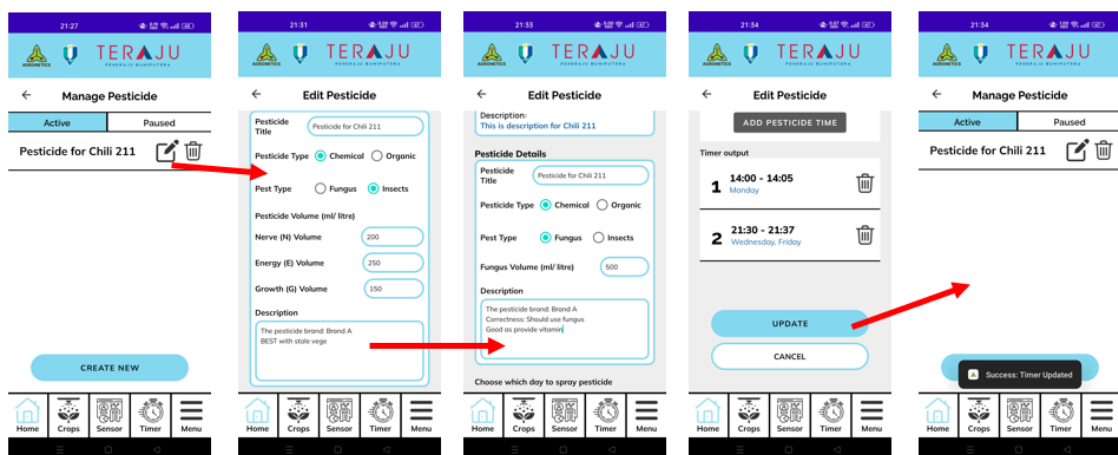


Figure 4.39 Edit Pesticide User Manual of Mobile\_AFS

Farmer need to click on the empty white space of the specific pesticide to navigate to View Pesticide interface. The details of the pesticide will be display.



Figure 4.40 View Pesticide Menu User Manual of Mobile\_AFS

#### 4.4.8 Module Sales

Farmer need to click on the” Sales” button to navigate to Manage Sales Menu interface.

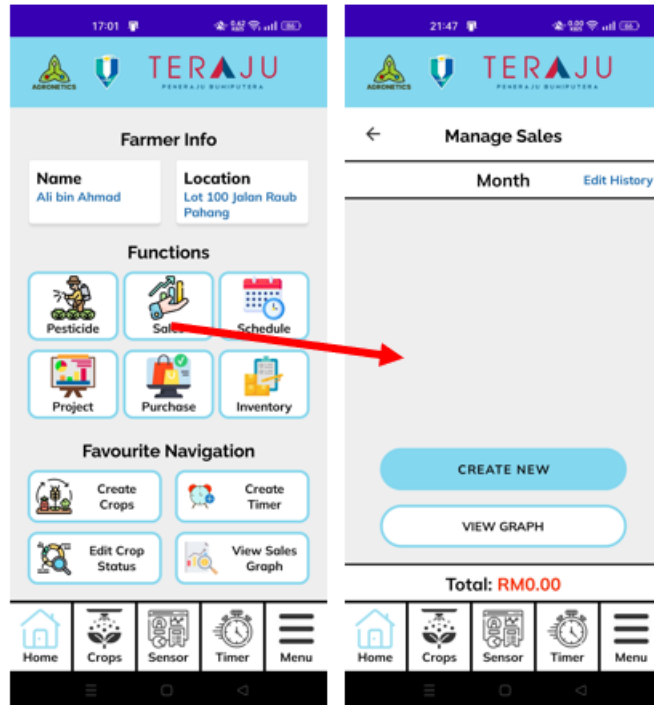


Figure 4.41 Manage Sales Menu User Manual of Mobile\_AFS

Farmer need to click on the” Create New” button to navigate to Add Sales interface. Farmer need to fill in the sales details by insert all the textbox and choose the crops type using the drop-down box. The sales date will choose by the date Picker. After all details is filled. Farmer need to click “ADD” button to create new dales to database.

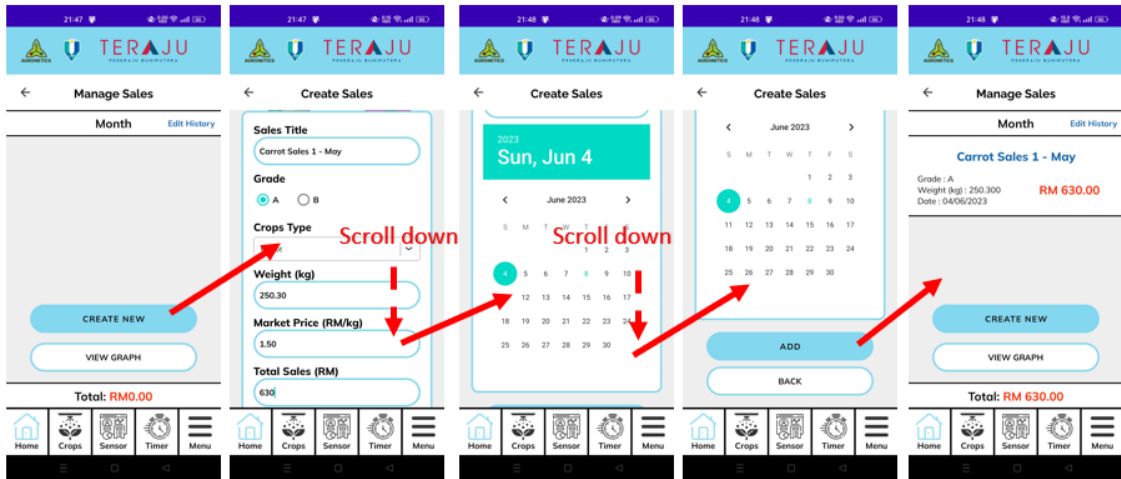


Figure 4.42 Add Sales User Manual of Mobile\_AFS

Farmer need to click the specific sales in the menu to navigate to the View Sales interface. The details of the sales will be display to the farmer.

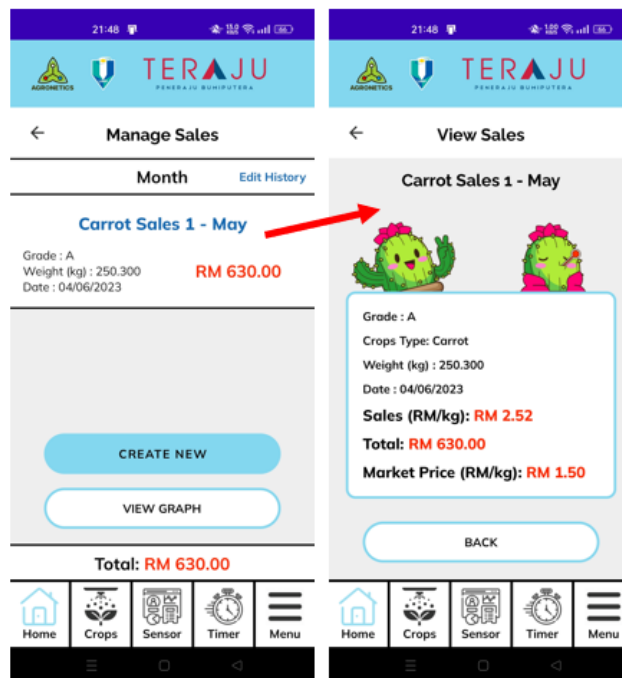


Figure 4.43 View Sales User Manual of Mobile\_AFS

Farmer need to click on the” Edit history” button to navigate to Edit Sales History interface. In this interface will display the edit request whether approve change request or not form the admin. However, in this example the farmer hasn’t make any edit request before.

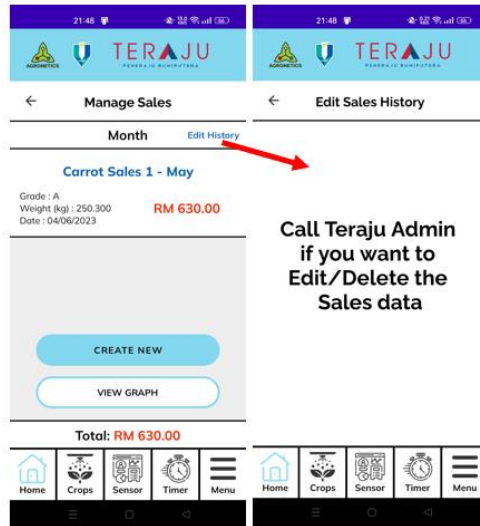


Figure 4.44 Edit Sales History User Manual of Mobile\_AFS

Farmer need to click the “View Graph” button to navigate to the View Sales Graph interface. The daily and crops graph of the sales will be display to the farmer.



Figure 4.45 View Sales Graph Manual of Mobile\_AFS



#### 4.4.9 Module Schedule

Farmer need to click on the” Schedule” button to navigate to Manage Schedule Menu interface.

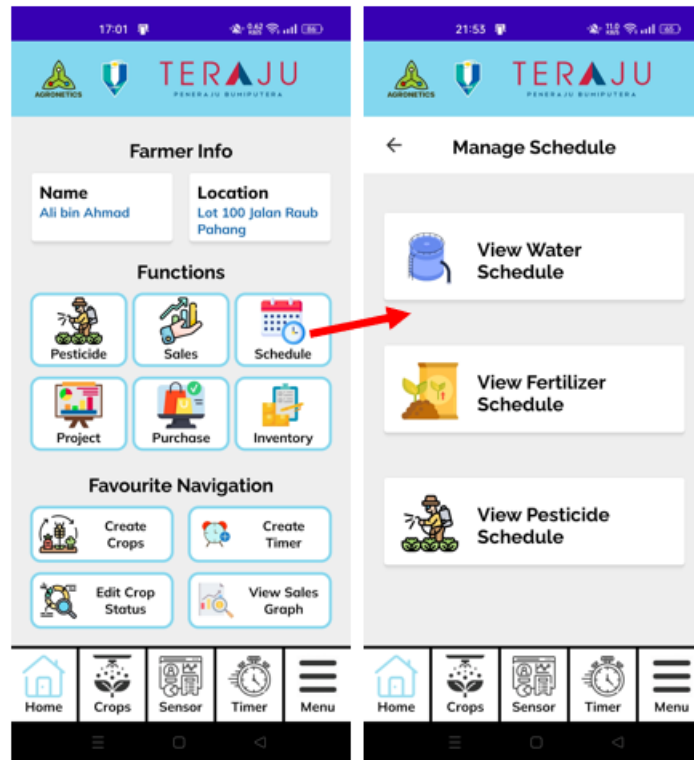


Figure 4.46 Manage Schedule Menu User Manual of Mobile\_AFS

Farmer need to click on the” View Water Schedule” button to navigate to View Water Schedule interface. Farmer can choose the date on the calendar and the list of water timer on that day will be displayed. Farmer can scroll down to have the full view of the timer.

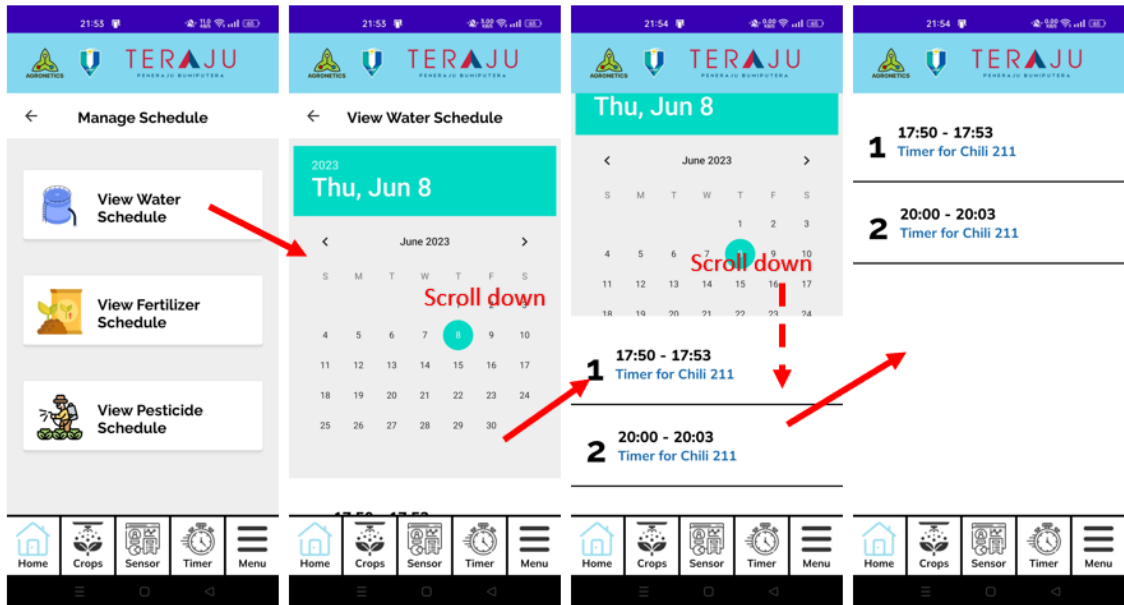


Figure 4.47 View Water Schedule User Manual of Mobile\_AFS

Farmer need to click on the” View Fertilizer Schedule” button to navigate to View Fertilizer Schedule interface. Farmer can choose the date on the calendar and the list of fertilizer timer on that day will be displayed. Farmer can scroll down to have the full view of the timer.

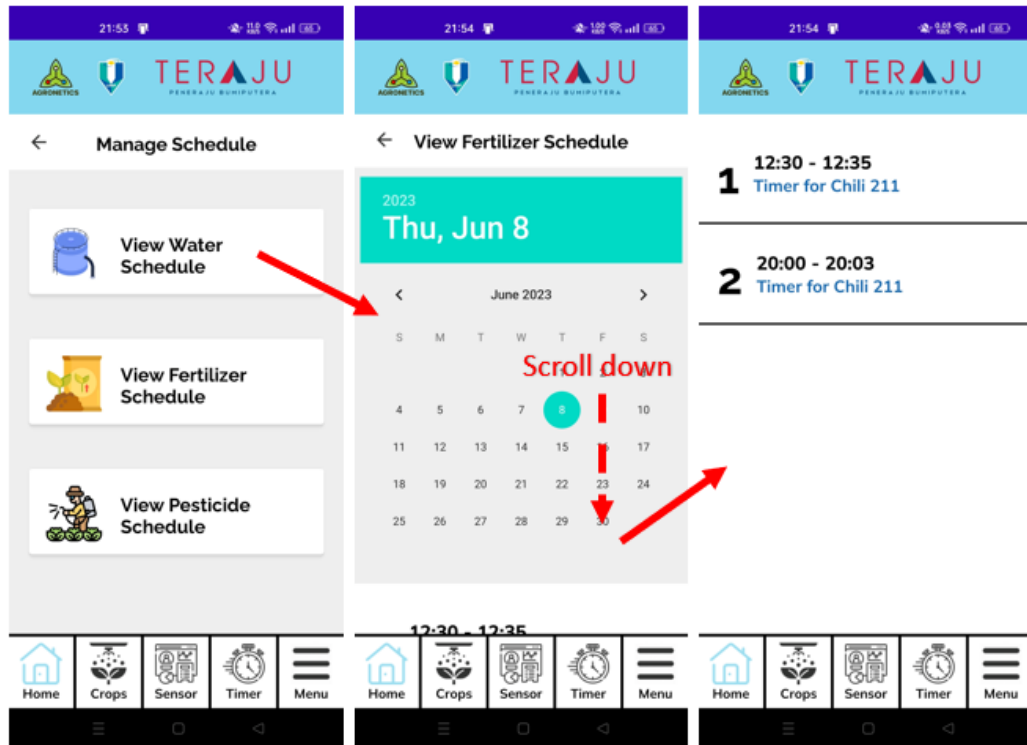


Figure 4.48 View Fertilizer Schedule User Manual of Mobile\_AFS

Farmer need to click on the” View Pesticide Schedule” button to navigate to View Pesticide Schedule interface. The first screen will be the day of current day. In the example, we do the testing on Thursday, therefore Thursday will display first. When the farmer pulls the screen from right to left (Pull from right). The day will switch form Thursday to Friday (increment by 1 day). On Friday, there exist the pesticide timer, therefore the timer will be display. As so on to the Monday.

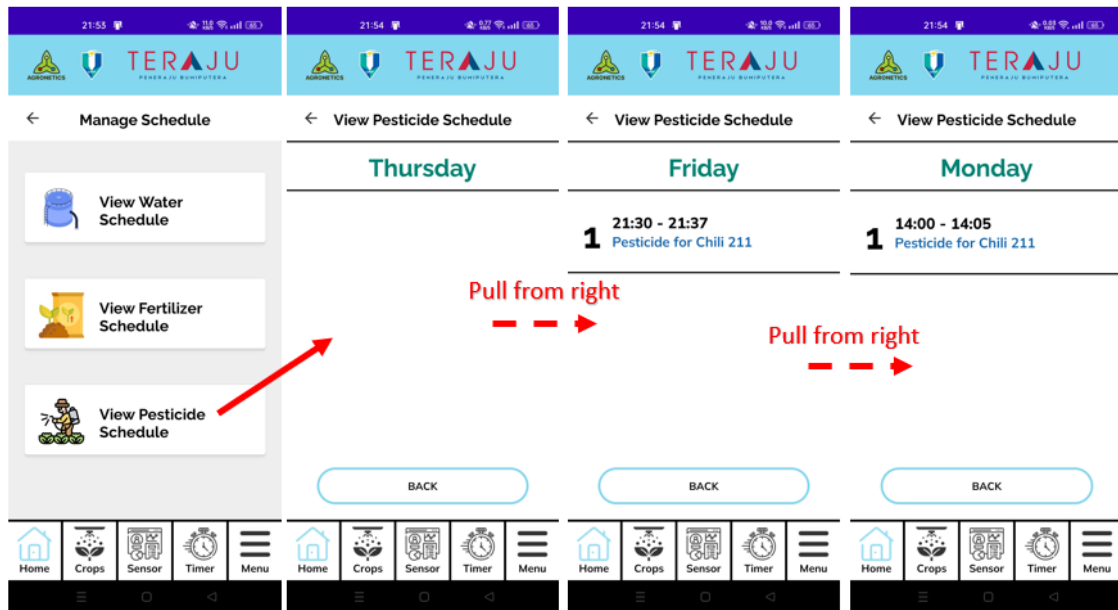


Figure 4.49 View Pesticide Schedule User Manual of Mobile\_AFS

#### 4.4.10 Module Project

Farmer need to click on the” Project” button to navigate to Manage Project Menu interface.

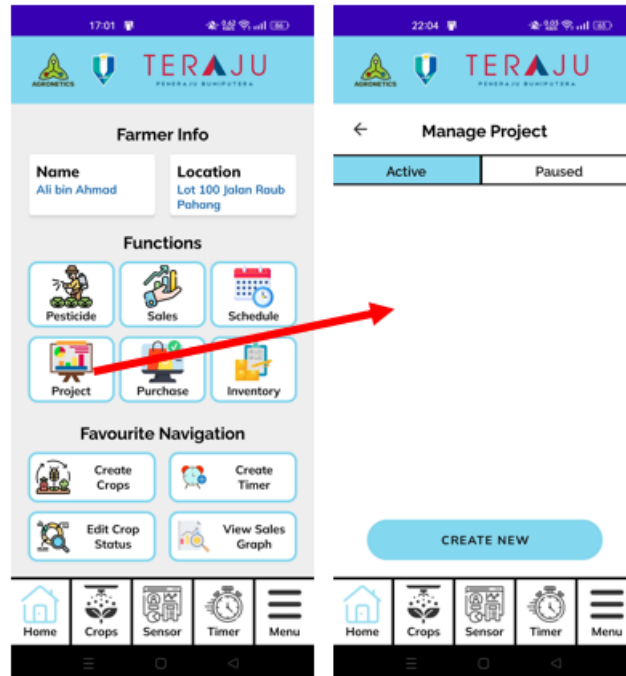


Figure 4.50 Manage Project Menu User Manual of Mobile\_AFS

Farmer need to click on the” Create New” button to navigate to Create Project interface.  
Farmer need to fill in all the textbox and choose the date on the date Picker.

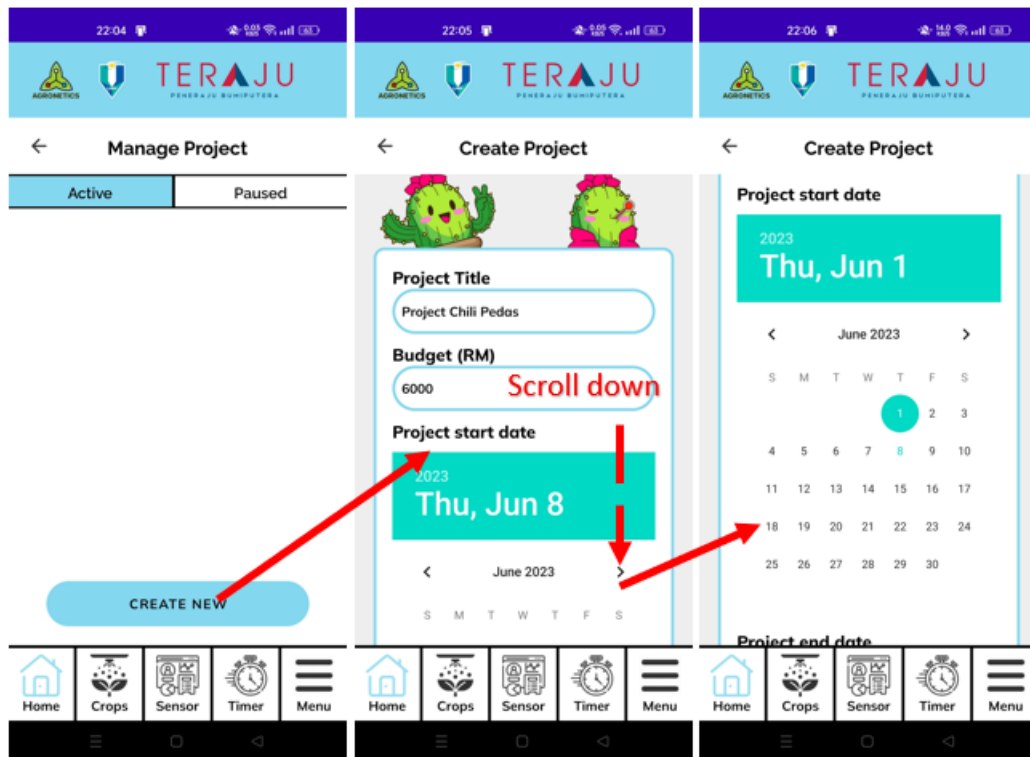


Figure 4.51 Create Project 1 User Manual of Mobile\_AFS

Farmer need to scroll down and choose the project end date using the date Picker also. Farmer need to choose the status of the project using radio button. After all, the farmer need to click on the “ADD” button to create new project to database.

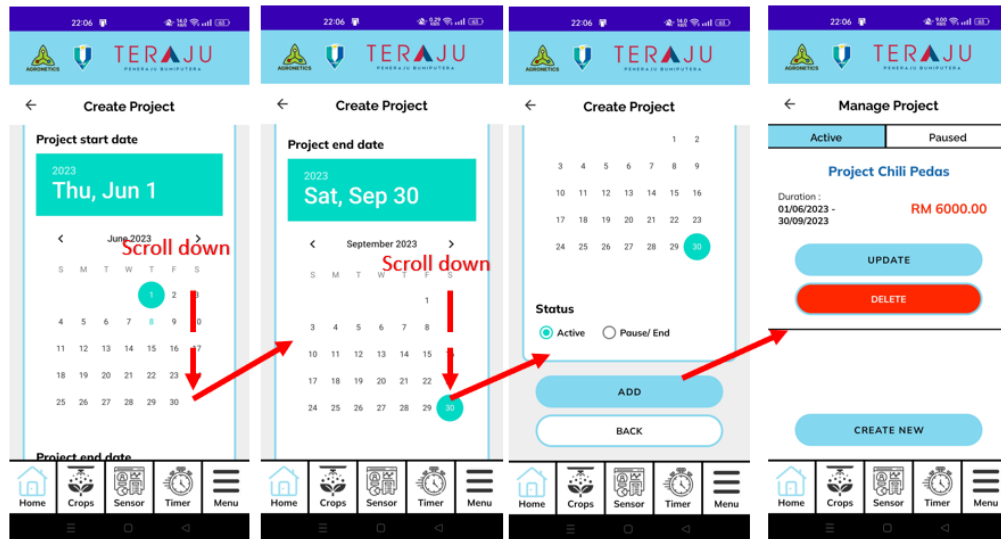


Figure 4.52 Create Project 2 User Manual of Mobile\_AFS

Farmer need to click on the” Update” button to navigate to Edit Project interface. In the example, the farmer updated the budget, then he clicks the update button. We can see the budget of the project increased.

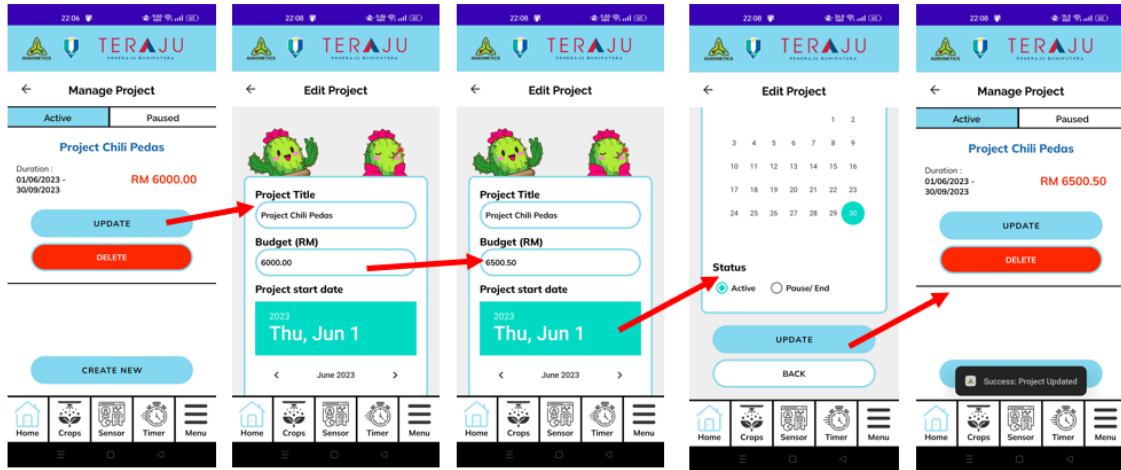


Figure 4.53 Edit Project User Manual of Mobile\_AFS



Farmer need to click on the white empty space on the specific project to open the View Project interface. The details of the project will be displayed to the farmer including the expenses from the purchase also.

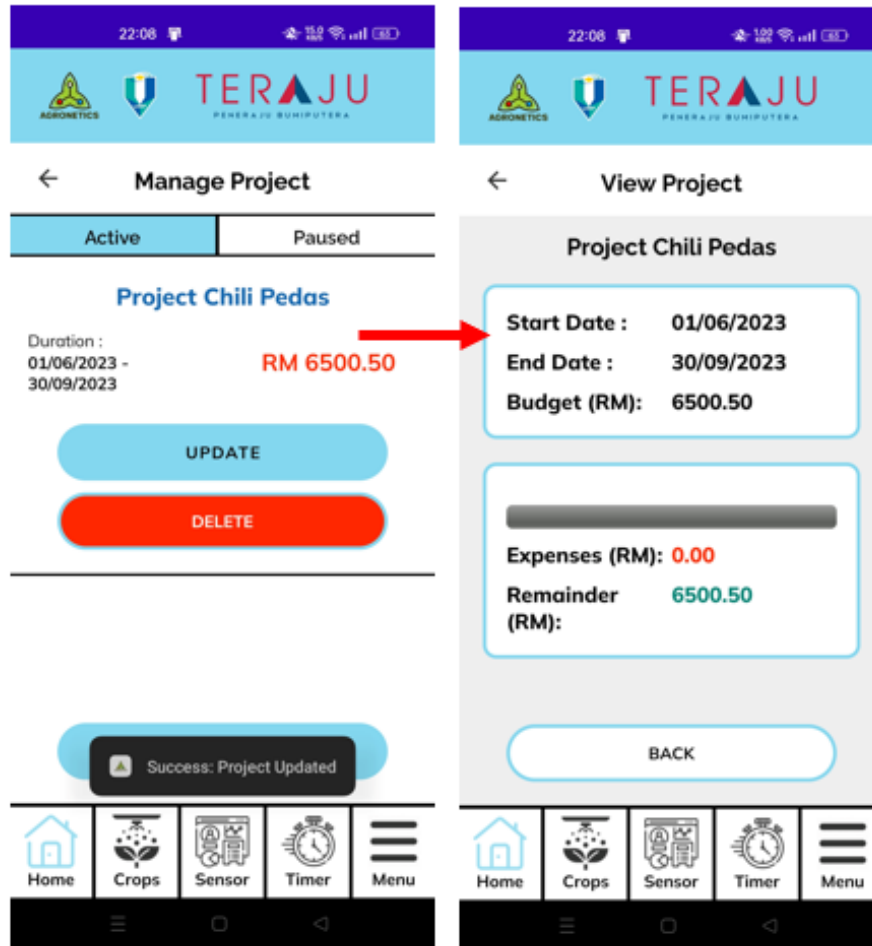


Figure 4.54 View Project User Manual of Mobile\_AFS

#### 4.4.11 Module Purchase

Farmer need to click on the” Purchase” button to navigate to Manage Purchase Menu interface.



Figure 4.55 Manage Purchase Menu User Manual of Mobile\_AFS

Farmer need to click on the” Create New” button to navigate to Purchase interface. Farmer need to choose the Project that need to add the Purchase (Receipt) using the drop-down box. Next, farmer need to fill in title and choose date of Purchase on the date Picker.

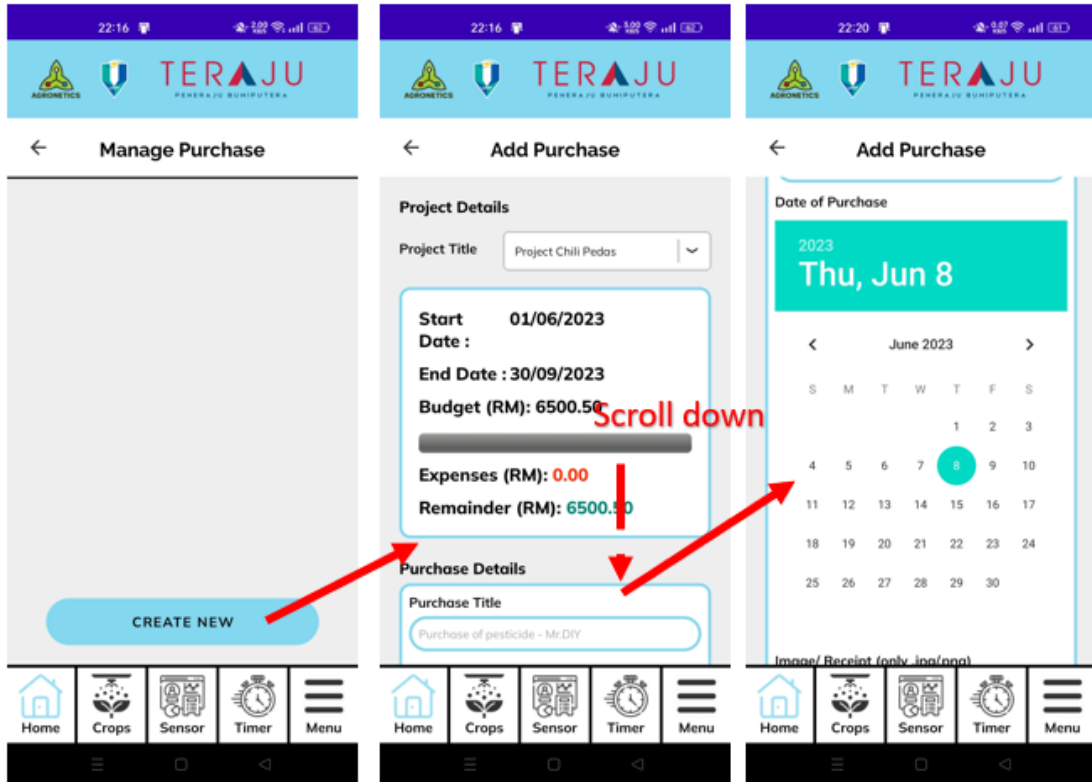


Figure 4.56 Create Purchase 1 User Manual of Mobile\_AFS

Farmer need to click on the “Select Image” button which navigate the user to gallery to choose the image (receipt) to upload. After the image is selected, the image name will be displayed so that farmer will know the image is chosen. After all, farmer will click the “Confirm” button to upload the data to database.

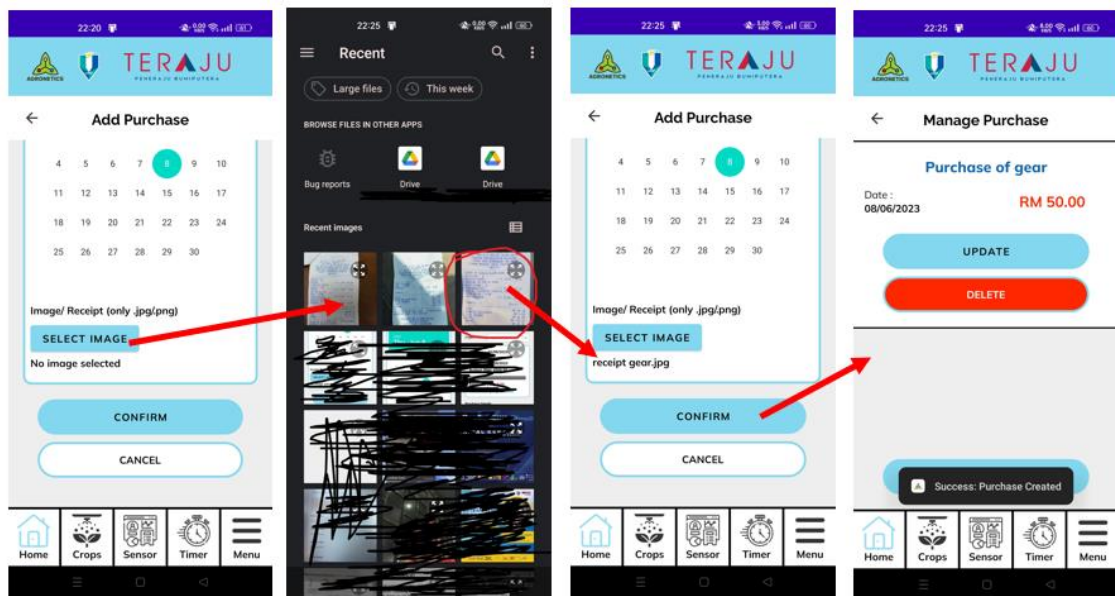


Figure 4.57 Create Project 2 User Manual of Mobile\_AFS

Farmer need to click on the” Update” button to navigate to Edit Purchase interface. In the example, the farmer updated the price, then he clicks the update button. We can see the price of the purchase increased to RM150.

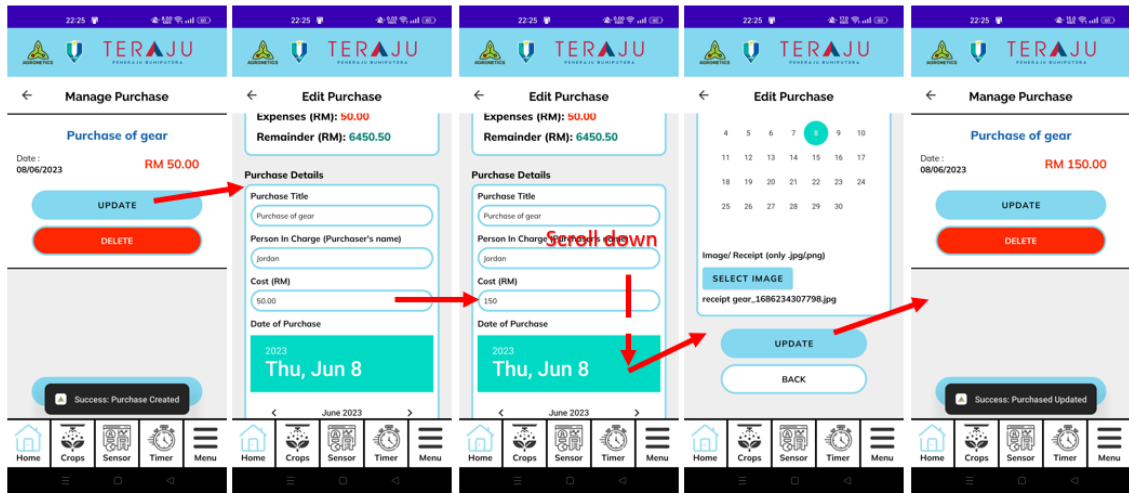


Figure 4.58 Edit Purchase User Manual of Mobile\_AFS

Farmer need to click on the white empty space on the specific purchase to open the View Purchase interface. The details of the purchase will be displayed to the farmer including the image of the purchase also.

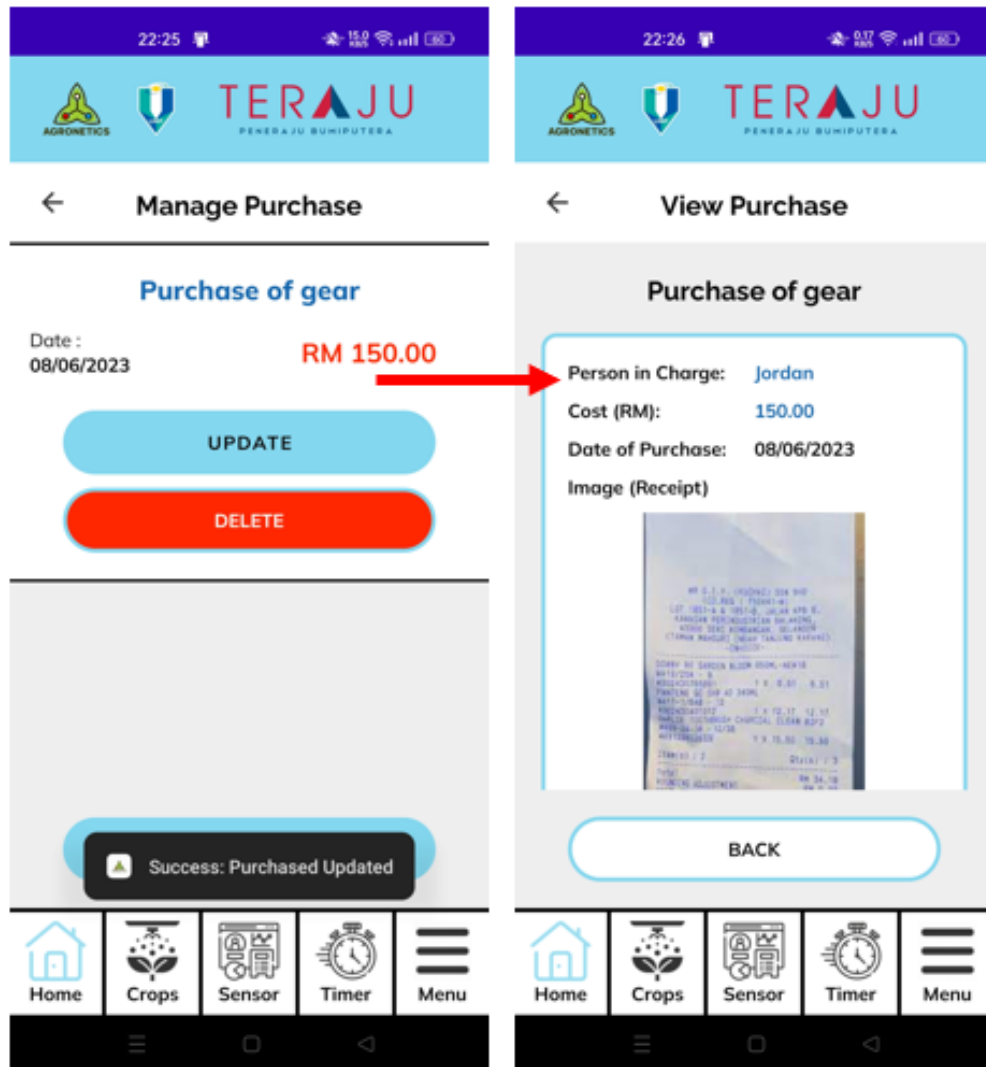


Figure 4.59 View Purchase User Manual of Mobile\_AFS

#### 4.4.12 Module Inventory

Farmer need to click on the” Inventory” button to navigate to Manage Inventory Main Menu interface. Then the farmer can click on the “Manage Stock Status” button to navigate to the Manage Stock Status Menu.

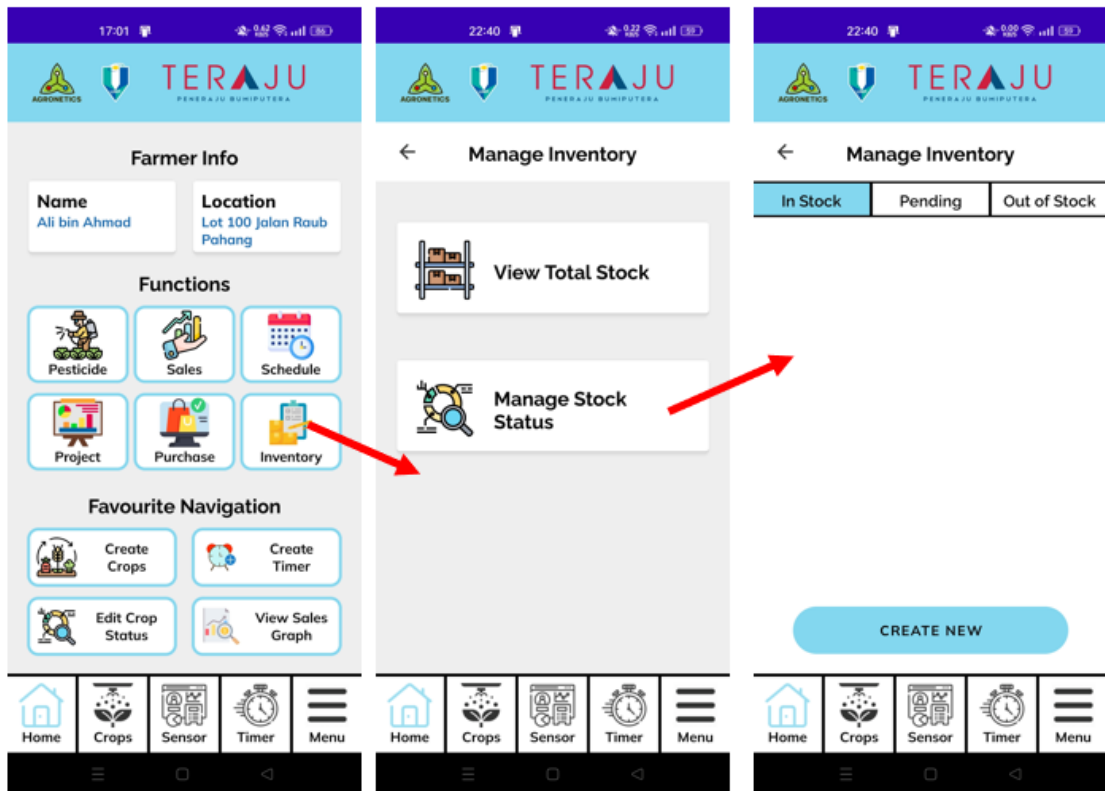


Figure 4.60 Manage Sotck Status Menu User Manual of Mobile\_AFS

Farmer need to click on the” Create New” button to navigate to Inventory interface. Farmer need to choose the Purchase that related to the Inventory item using the drop-down box. Next, farmer need to fill in the inventory details.

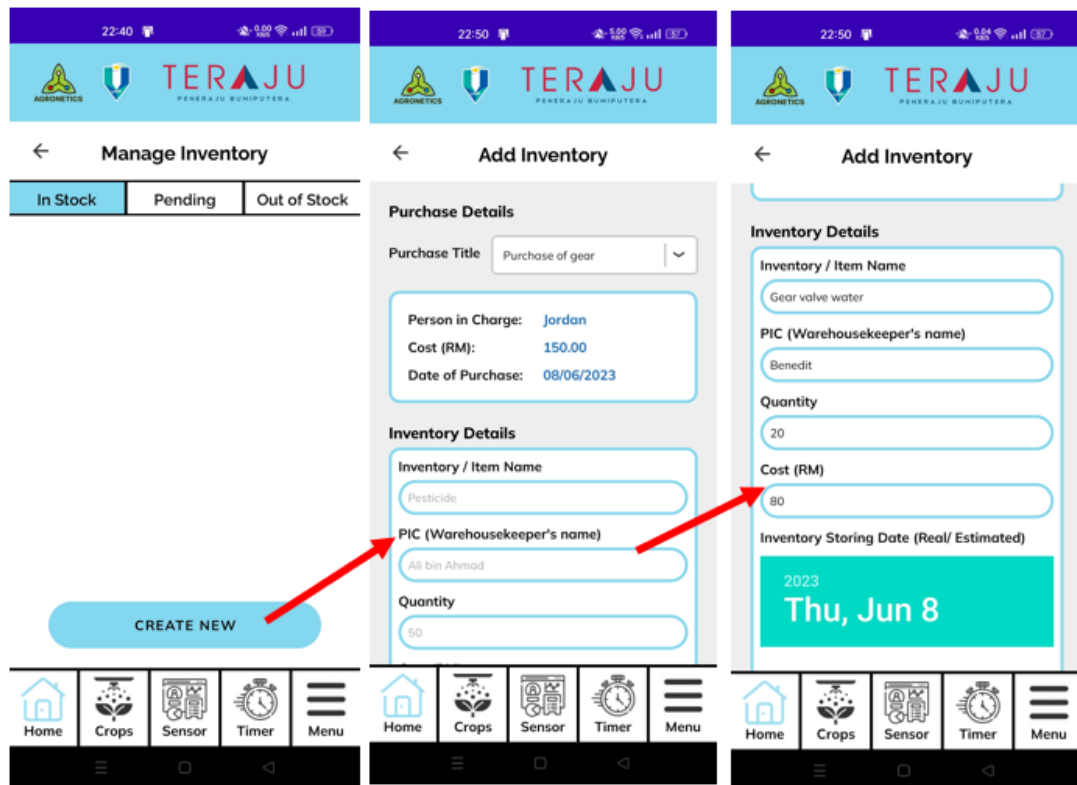


Figure 4.61 Create Inventory 1 User Manual of Mobile\_AFS



Farmer to choose the estimated arrive date for the inventory using the Date Picker. Then, the farmer needs to choose the status of the inventory using the radio button. After all, the farmer can click on the “Confirm” button to upload the details to the database.

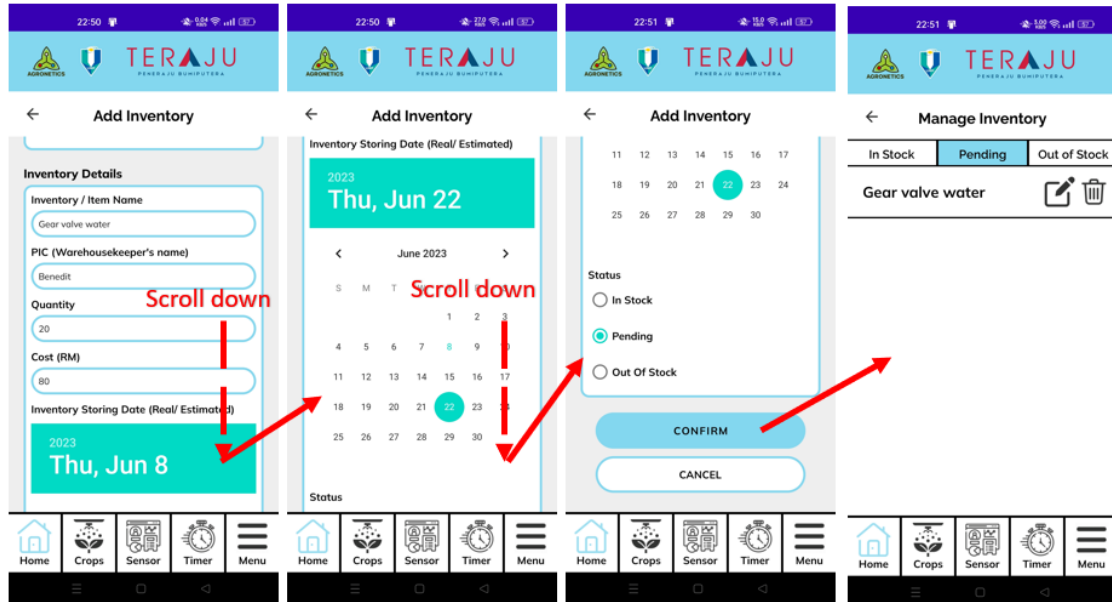


Figure 4.62 Create Inventory 2 User Manual of Mobile\_AFS

Farmer need to click on the” Update” button to navigate to Edit Inventory interface. In the example, the farmer updated the quantity either using the minus button or directly enter value using the textbox. The status also been updated. After all, he click the “Update” button and success update message display.

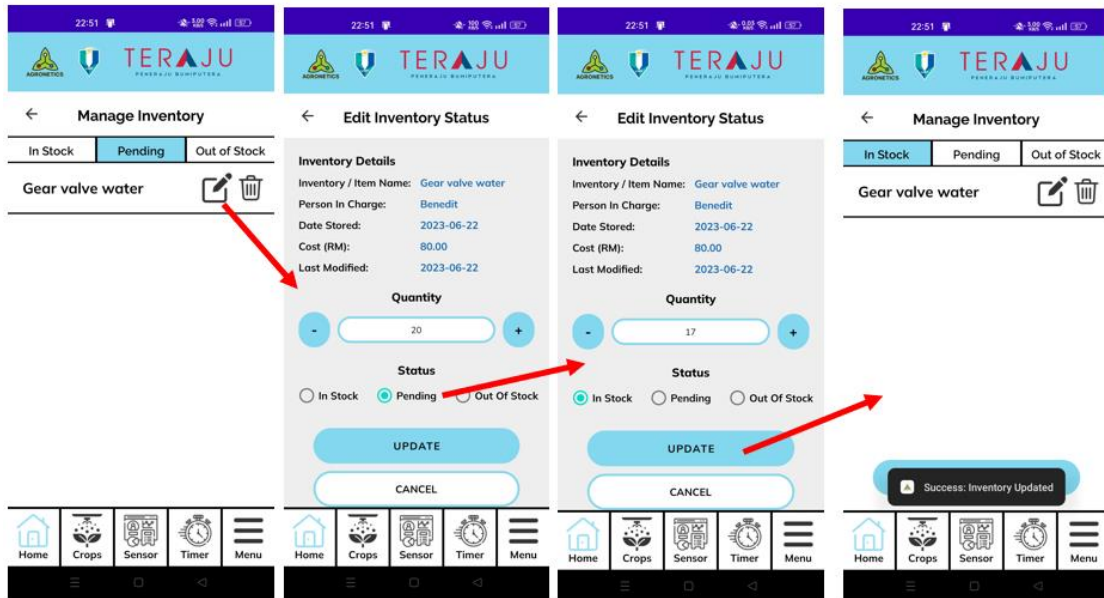


Figure 4.63 Edit Inventory User Manual of Mobile\_AFS

Farmer need to click on the white empty space on the specific inventory to open the View Inventory interface. The details of the inventory will be displayed to the farmer.

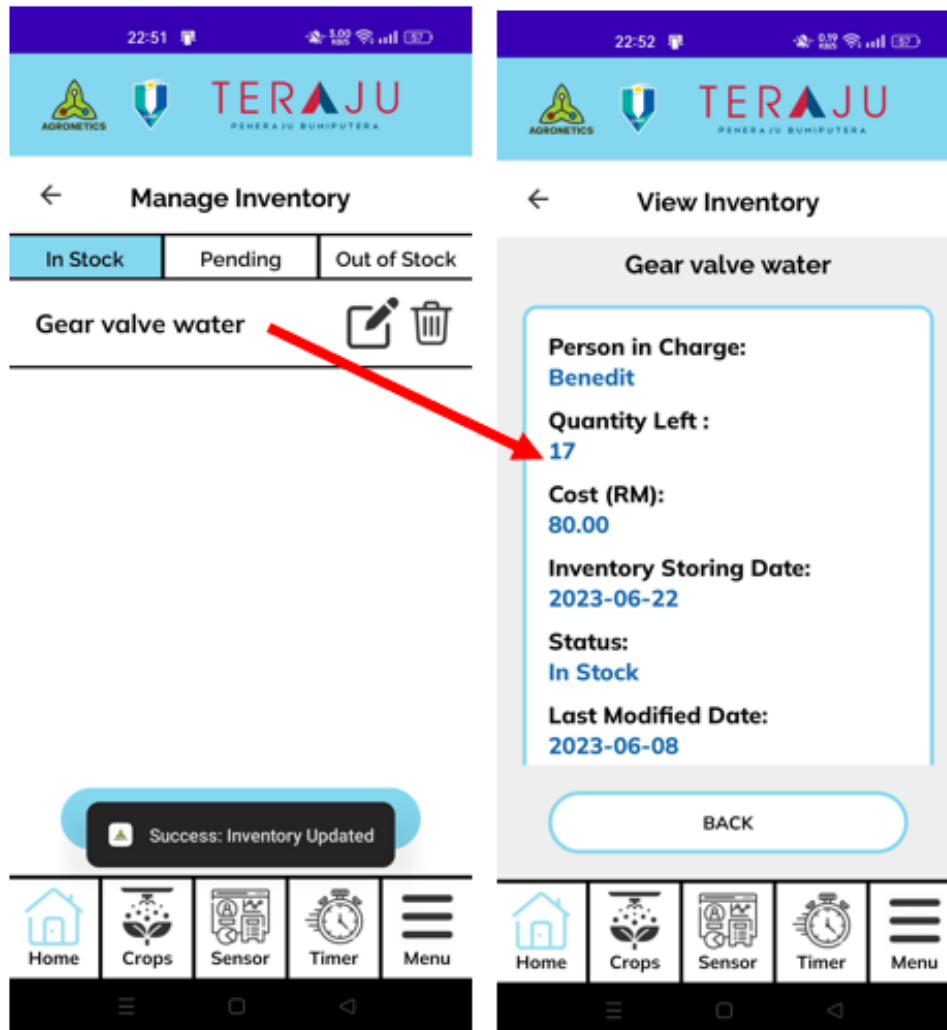


Figure 4.64 View Inventory User Manual of Mobile\_AFS

Farmer need to click “View Total Stock” button to navigate to the Total Stock Menu interface. In the interface, all the in-stock item will be display. In the example, we only have the Gear valve water only. Therefore, when farmer click on the item, the View Total Stock interface is opened. The details of the inventory item will be displayed to the farmer.

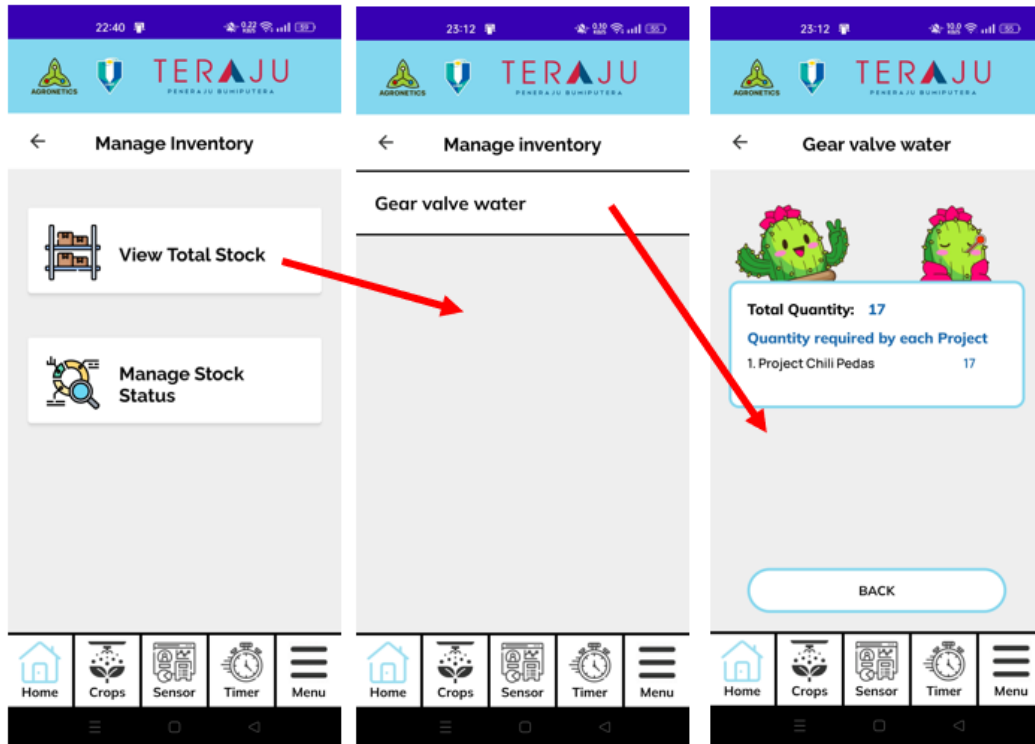


Figure 4.65 View Total Stock User Manual of Mobile\_AFS

## 4.5 Coding implementation

### 4.5.1 Login

By using Volley library, the URL to the login.php is given from the Android Studio.

```
RequestQueue queue = Volley.newRequestQueue(getApplicationContext());  
//get Ip address from string.xml  
String ipAddress = "mobileafs.x10.mx";  
String url = "https://" + ipAddress + "/AFS/Manage_Login/Login.php";
```

Figure 4.66 Login code 1

The email and password of Farmer object is sent to the specific URL which contain the PHP files for validate the data in the database.

```
    }  
    }  
    protected Map<String, String> getParams(){  
        Map<String, String> paramV = new HashMap<>();  
        paramV.put(k "email", farmer.getEmail());  
        paramV.put(k "password", farmer.getPassword());  
        return paramV;  
    }  
};  
queue.add(stringRequest);
```

Figure 4.67 Login code 2

In PHP, we extract the email and password from Farmer table and verify. The farmID is selected and return to the Android Studio for the Session used.

```
/*
1. check for the email and password
2. join table Farmer and Farm on the same FarmerID
*/
$sql = "SELECT F.id, F.email, F.password, Farm.farmID
FROM (
    SELECT id, email, password
    FROM Farmers
    WHERE email = '$email' AND password = '$password'
) F
INNER JOIN Farm ON F.id = Farm.farmerID";

$result= mysqli_query($conn,$sql);

//if got valid email & password
if(mysqli_num_rows($result) !=0){
    $farmerArr = array();
    $i=0;
    while($row = mysqli_fetch_assoc($result)){
        $farmerArr[$i] = $row;
        $i++;
    }

    // Send response as JSON
    header('Content-Type: application/json');
    echo json_encode($farmerArr, JSON_PRETTY_PRINT);
}
else{
    echo "Invalid email or password" ;
}
}
```

Figure 4.68 Login code 3

## 4.5.2 Forget Password (Sent Email)

We using the PHPMailer to help us sent the email to the client.

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require '../..//PHPMailer/src/Exception.php';
require '../..//PHPMailer/src/PHPMailer.php';
require '../..//PHPMailer/src/SMTP.php';
```

Figure 4.69 Forget Password code 1

When farmer enter their email address, the email is POST and the reset\_token is setup for that farmer. The farmer will receive the email to reset the password inside of the content in the email.

```
if (isset($_POST['email'])) {
    $email= $_POST["email"];

    $sql = "SELECT * FROM farmers WHERE email = '$email'";
    $result= mysqli_query($conn,$sql);
    if (mysqli_num_rows($result) !=0)
    {
        $reset_token = time() . md5($email);

        $token_sql = "UPDATE farmers SET reset_token='$reset_token' WHERE email='$email'";
        mysqli_query($conn, $token_sql);

        $message = "<p>Please click the link below to reset your password</p>";
        $message .= "<a href='https://mobileafs.x10.mx/AFS/Manage_Login/reset-password.php?e"
        $message .= "Reset password";
        $message .= "</a>";

        send_mail($email, "Reset password", $message);
    }
    else
    {
        echo "Email does not exists";
    }
}
```

Figure 4.70 Forget Password code 2

### 4.5.3 Create Operation (Create Crops)

In this figure, we can see the url to the create\_crop.php which to the server side. The Toast will be display after the process in the server side completed.

```
String url = "https://" + ipAddress + "/AFS/Manage_Crops/create_crop.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            if(response.equals("Create Success")){
                Toast.makeText(context, text: "Success: Crops Created", Toast.LENGTH_SHORT).s

                ft= getFragmentManager().beginTransaction();
                ManageCropsMenuActive manageCropsMenuActive= new ManageCropsMenuActive();
                ft.replace(R.id.fragment_place,manageCropsMenuActive);
                ft.commit();
            }
            //failed update data
            else{
                Toast.makeText(context, response, Toast.LENGTH_SHORT).show();
                btnCreate.setClickable(true);
                progressBar.setVisibility(View.GONE);
            }
        }
    }
);
```

Figure 4.71 Create crops Android Studio code 1



All the details of the Crops data are sent to the PHP files. When JSON response is received back from the PHP file.

```
//sent the Crop class data to the PHP
protected Map<String, String> getParams(){
    Map<String, String> paramV = new HashMap<>();
    paramV.put(k "farmID", farmID);
    paramV.put(k "title", crop.getTitle());
    paramV.put(k "size", crop.getSize());
    paramV.put(k "type", crop.getType());
    paramV.put(k "description", crop.getDescription());
    paramV.put(k "cropsNum", Integer.toString(crop.getCropsNum()));
    paramV.put(k "HST1Start", Integer.toString(crop.getHST1Start()));
    paramV.put(k "HST1End", Integer.toString(crop.getHST1End()));
    paramV.put(k "minWater1", Integer.toString(crop.getMinWater1()));
    paramV.put(k "minEC1", Integer.toString(crop.getMinEC1()));
    paramV.put(k "incrementInterval", Integer.toString(crop.getIncrementInterval()));
    paramV.put(k "incrementAmt", Integer.toString(crop.getIncrementAmt()));
    paramV.put(k "HST2Start", Integer.toString(crop.getHST2Start()));
    paramV.put(k "HST2End", Integer.toString(crop.getHST2End()));
    paramV.put(k "minWater2", Integer.toString(crop.getMinWater2()));
    paramV.put(k "minEC2", Integer.toString(crop.getMinEC2()));
    //good= crops num, because all crops estimate to be good when create
    paramV.put(k "good", Integer.toString(crop.getCropsNum()));
    paramV.put(k "bad", Integer.toString(i: 0));
    paramV.put(k "dateCreated", Long.toString( System.currentTimeMillis()));
    return paramV;
}
```

Figure 4.72 Create crops Android Studio code 2

When the details of crops received, the SQL is written to create the new crops into the Crops Table.

```
$sql= "INSERT INTO `Crops`
(`farmID`, `title`, `size`, `type`,
`description`, `cropsNum`, `HST1Start`,
`HST1End`, `minWater1`, `minEC1`,
`incrementInterval`, `incrementAmt`, `HST2Start`,
`HST2End`, `minWater2`, `minEC2`,
`good`, `bad`, `dateCreated`)
VALUES
('$farmID', '$title', '$size', '$type',
'$description', '$cropsNum', '$HST1Start',
'$HST1End', '$minWater1', '$minEC1',
'$incrementInterval', '$incrementAmt', '$HST2Start',
'$HST2End', '$minWater2', '$minEC2',
'$good', '$bad', '$dateCreated');";

if ($conn->query($sql) === TRUE)
{
    echo "Create Success";
}
else
{
    echo "Failed: Error Creating Crops. Try Again. \n";
    echo $conn->error;
}
```

Figure 4.73 Create crops Visual Studio code

#### 4.5.4 View Operation (View Crops)

In this figure, we can see the url to the view\_crop.php which to the server side. All the received data will be saved to the local variable in order to display to the interface.

```
String url ="https://" + ipAddress + "/AFS/Manage_Crops/view_crops.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try{
                JSONArray jsonArray= new JSONArray(response);
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject jsonObject= jsonArray.getJSONObject(i);
                    int farmID = jsonObject.getInt( name: "farmID");
                    String cropsTitle= jsonObject.getString( name: "title");
                    String cropsSize= jsonObject.getString( name: "size");
                    String cropsType= jsonObject.getString( name: "type");
                    String cropsDescription= jsonObject.getString( name: "description");
                    int cropsNum = jsonObject.getInt( name: "cropsNum");
                    int HST1Start = jsonObject.getInt( name: "HST1Start");
                    int HST1End = jsonObject.getInt( name: "HST1End");
                    int minWater1 = jsonObject.getInt( name: "minWater1");
                    int minEC1 = jsonObject.getInt( name: "minEC1");
                    int incrementInterval = jsonObject.getInt( name: "incrementInterval");
                    int incrementAmt = jsonObject.getInt( name: "incrementAmt");
                    int HST2Start = jsonObject.getInt( name: "HST2Start");
                    int HST2End = jsonObject.getInt( name: "HST2End");
                    int minWater2 = jsonObject.getInt( name: "minWater2");
                    int minEC2 = jsonObject.getInt( name: "minEC2");
```

Figure 4.74 View crops Android Studio code 1

The cropsID is sent to the PHP files and all the details of the crops is retrieved and sent to the here.

```
protected Map<String, String> getParams(){
    Map<String, String> paramV = new HashMap<>();
    paramV.put( k: "cropsID", Integer.toString(cropsID));
    return paramV;
}
```

Figure 4.75 View crops Android Studio code 2

The SQL will look through the Crops table until found the cropsID that same with the received cropsID from the client side. The details of the crops will then sent back to the client.

```
if (isset($_POST['cropsID'])) {  
  
    $cropsID= $_POST["cropsID"];  
  
    $sql = "SELECT  
farmID, title, size, type,  
description, cropsNum, HST1Start, HST1End,  
minWater1, minEC1, incrementInterval, incrementAmt,  
HST2Start, HST2End, minWater2, minEC2  
    FROM crops  
    WHERE cropsID = '$cropsID'  
    LIMIT 1";  
  
    $result= mysqli_query($conn,$sql);  
  
    if(mysqli_num_rows($result) !=0){  
        $data = array();  
        $i = 0;  
        while($row = mysqli_fetch_assoc($result)){  
            $data[$i] = $row;  
            $i++;  
        }  
        echo json_encode($data, JSON_PRETTY_PRINT);  
    }  
    else{  
        //display error msg  
        echo "Error: " . $conn->error;  
    }  
}
```

Figure 4.76 View crops Visual Studio code

#### 4.5.5 Update Operation (Update Crops)

In this figure, we can see the URL to the update\_crop.php which to the server side. The Toast will be display after the process in the server side completed.

```
String url = "https://" + ipAddress + "/AFS/Manage_Crops/update_crops.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {

            if(response.equals("Update Success")){
                Toast.makeText(context, text: "Success: Crops Updated", Toast.LENGTH_SHORT)

                ft= getFragmentManager().beginTransaction();
                ManageCropsMenuActive manageCropsMenuActive= new ManageCropsMenuActive();
                ft.replace(R.id.fragment_place,manageCropsMenuActive);
                ft.commit();
            }
            //failed update data
            else{
                Toast.makeText(context, response, Toast.LENGTH_SHORT).show();
                btnUpdate.setClickable(true);
                progressBar.setVisibility(View.GONE);
            }
        }
    }
}
```

Figure 4.77 Update crops Android Studio code 1

All the updated Crops data are sent through here inside the HashMap to the PHP files.

```
//sent the Crop class data to the PHP
protected Map<String, String> getParams(){
    Map<String, String> paramV = new HashMap<>();
    paramV.put(k: "cropsID", Integer.toString(cropsID));
    paramV.put(k: "title", crop.getTitle());
    paramV.put(k: "size", crop.getSize());
    paramV.put(k: "type", crop.getType());
    paramV.put(k: "description", crop.getDescription());
    paramV.put(k: "cropsNum", Integer.toString(crop.getCropsNum()));
    paramV.put(k: "HST1Start", Integer.toString(crop.getHST1Start()));
    paramV.put(k: "HST1End", Integer.toString(crop.getHST1End()));
    paramV.put(k: "minWater1", Integer.toString(crop.getMinWater1()));
    paramV.put(k: "minEC1", Integer.toString(crop.getMinEC1()));
    paramV.put(k: "incrementInterval", Integer.toString(crop.getIncrementInterval()));
    paramV.put(k: "incrementAmt", Integer.toString(crop.getIncrementAmt()));
    paramV.put(k: "HST2Start", Integer.toString(crop.getHST2Start()));
    paramV.put(k: "HST2End", Integer.toString(crop.getHST2End()));
    paramV.put(k: "minWater2", Integer.toString(crop.getMinWater2()));
    paramV.put(k: "minEC2", Integer.toString(crop.getMinEC2()));
    //if status is true
    if(crop.getStatus()){
        paramV.put(k: "status", v: "1");
    }
    else{
        paramV.put(k: "status", v: "0");
    }
}
```

Figure 4.78 Update crops Android Studio code 2

When the details of crops received, the SQL is written to update the crops details into the Crops Table where the cropsID is match.

```
$sql="UPDATE crops SET
`title`='$title', `size`='$size', `type`='$type',
`description`='$description', `cropsNum`='$cropsNum', `HST1Start`='$HST1Start',
`HST1End`='$HST1End', `minWater1`='$minWater1', `minEC1`='$minEC1',
`incrementInterval`='$incrementInterval', `incrementAmt`='$incrementAmt', `HST2Start`='$HST2Start',
`HST2End`='$HST2End', `minWater2`='$minWater2', `minEC2`='$minWater2',
`status`='$status'
WHERE `cropsID`='$cropsID';

if ($conn->query($sql) === TRUE)
{
    echo "Update Success";
}
else
{
    echo "Failed: Error Updating Crops Data. Try Again.";
}
```

Figure 4.79 Update crops Visual Studio code

When the details of crops received, the SQL is written to update the crops details into the Crops Table where the cropsID is match.

#### 4.5.6 Delete Operation (Delete Crops)

The url will direct the request to the delete\_crops.php. After server side there complete the delete operation, the recycle view adapter will remove the existing specific crops from the array list and notify update the recycle view.

```
//Button 'Delete' Action Here
viewHolder.btnDelete.setOnClickListener(view -> {
    //Alert Box for Delete
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setTitle("Delete Confirmation")
        .setMessage("Are you sure you want to delete this item?")
        .setPositiveButton(text: "Delete", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String url = "https://" + ipAddress + "/AFS/Manage_Crops/delete_crops.php";

                StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String response) {

                            if(response.equals("Delete Success")){
                                // Perform delete operation here
                                idList.remove(position);
                                lblCropsTitleList.remove(position);
                                lblCropsActiveDaysList.remove(position);
                                //notify the changes is done
                                notifyItemRemoved(position);
                                notifyItemRangeChanged(position, idList.size());
                            }
                            //Toast Text (Success/ Failed response)
                            Toast.makeText(context, response, Toast.LENGTH_SHORT).show()
                        }
                    }
                );
            }
        });
}
```

Figure 4.80 Delete crops Android Studio code 1



Delete operation is normally done in the recycle view adapter. Therefore, the passing data will need to search for the array list of cropsID and get the position, so that it knows which crops will be deleted.

```
//sent the Crop class data to the PHP
protected Map<String, String> getParams(){
    Map<String, String> paramV = new HashMap<>();
    paramV.put(k "cropsID", Integer.toString(idList.get(position)))
    return paramV;
}
```

Figure 4.81 Delete crops Android Studio code 2

Due to the cropsID is the reference key or foreign key to the Autotimer and Pesticide table, therefore if the all the data that exist the cropsID will also be removed.

```
if (isset($_POST['cropsID'])) {
    $cropsID= $_POST["cropsID"];

    //remove all the related things with cropsID
    $sql = "DELETE crops, autotimer, pesticide
    FROM crops
    LEFT JOIN autotimer ON autotimer.cropsID = crops.cropsID
    LEFT JOIN pesticide ON pesticide.cropsID = crops.cropsID
    WHERE crops.cropsID = $cropsID";

    mysqli_query($conn, $sql);

    if ($conn->query($sql) === TRUE)
    {
        echo "Delete Success";
    }
    else
    {
        echo "Error: " . $conn->error;
    }
}
```

Figure 4.82 Delete crops Visual Studio code

#### 4.5.7 Schedule Operation (Pesticide Schedule)

Retrieve the all the timer for pesticide schedule from server. The data will then save to the array list of Pesticide Class.

```
String url ="https://" + ipAddress + "/AFS/Manage_Schedule/display_pesticide_schdule.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try{
                JSONArray jsonArray= new JSONArray(response);
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject jsonObject= jsonArray.getJSONObject(i);
                    String pesticideTitle= jsonObject.getString( name: "pesticideTitle");
                    String days= jsonObject.getString( name: "days");
                    String schedule= jsonObject.getString( name: "schedule");
                    String timer = jsonObject.getString( name: "timer");

                    //add to arrylisy
                    pesticideList.add(new Pesticide(pesticideTitle, days, schedule, timer));
                    //add all the data to list
                    preloadPesticideTimer(pesticideTitle, days, schedule, timer);
                }
            }
        }
    });
```

Figure 4.83 Schedule Pesticide Android Studio code – Retrieve from db

The function is used to display the selected day only pesticide schedule data. Let's say the selected day is Friday, then only the schedule pesticide timer on Friday will display only.

```
private boolean displaySelectedDayOnly(String daysSelected){
    //set the header
    lblHeaderDay.setText(dayOfWeekString);

    return dayOfWeekString.equals(daysSelected);
}
```

Figure 4.84 Schedule Pesticide Android Studio code – Display selected day only

All the timer of the pesticide will be sort according using the bubble sort algorithm.

```
private void sortScheduleTimeASC(){
    for (int i = 0; i < scheduleList.size()-1; i++) {
        for (int j = 0; j < scheduleList.size()-i-1; j++) {
            //swap time is j+1 < j/ j > j+1
            if(scheduleList.get(j).compareTo(scheduleList.get(j+1)) > 0){
                //swap j+1 with j
                Collections.swap(titleList, j, j+1);
                Collections.swap(daysList, j, j+1);
                Collections.swap(scheduleList, j, j+1);
                Collections.swap(timerList, j, j+1);
            }
        }
    }
}
```

Figure 4.85 Schedule Pesticide Android Studio code – Sort timer

This interface implements the Gesture Listener which will know the direction of farmer swiping.

```
private final class GestureListener extends GestureDetector.SimpleOnGestureListener {
```

Figure 4.86 Schedule Pesticide Android Studio code – Sort timer 1

The diff x and diff y is use to detect the movement horizontal and vertical respectively.

```
float diffY = e2.getY() - e1.getY();
float diffX = e2.getX() - e1.getX();
if (Math.abs(diffX) > Math.abs(diffY)) {
    if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) > SWIPE_VELOCITY_THRE
        if (diffX > 0) {
            onSwipeRight();
        } else {
            onSwipeLeft();
        }
        result = true;
    }
}
else if (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) > SWIPE_VELOCITY_THRE
    if (diffY > 0) {
        onSwipeBottom();
    } else {
        onSwipeTop();
    }
    result = true;
}
```

Figure 4.87 Schedule Pesticide Android Studio code – Sort timer 2

If the user swipe from right to left (on swipe left), the day will increment one, from Friday -> Saturday, and vice versa for the on swipe right.

```
public void onSwipeRight() {
    // Deduct one day here
    calendar.add(Calendar.DAY_OF_MONTH, 1, -1);
    int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
    changeDateToString(dayOfWeek);

    //reload the recycle view
    reloadRecyclerView();
    sortScheduleTimeASC();
    // Notify the RecyclerView adapter that the data has changed
    adapterManageScheduleViewDaily.notifyDataSetChanged();
}

public void onSwipeLeft() {
    // Add one day here
    calendar.add(Calendar.DAY_OF_MONTH, 1, 1);
    int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
    changeDateToString(dayOfWeek);
    //reload the recycle view
    reloadRecyclerView();
    sortScheduleTimeASC();
    // Notify the RecyclerView adapter that the data has changed
    adapterManageScheduleViewDaily.notifyDataSetChanged();
}
```

Figure 4.88 Schedule Pesticide Android Studio code – Swipe Action

### 4.5.8 Validation operation

If the farmer enters empty value in the Edittext or Textbox, the error will prompt to the farmer tell them to fill the empty field.

```
private void emptyValidation(String projectTitle,String budgetString){  
  
    if (projectTitle.equals("")) {  
        txtProjectTitle.setError("Please fill in PROJECT TITLE field");  
        txtProjectTitle.requestFocus();  
    }  
    else if (budgetString.equals("")) {  
        txtBudget.setError("Please fill in BUDGET field without empty spacing");  
        txtBudget.requestFocus();  
    }  
}
```

Figure 4.89 Empty Validation

In Manage Login, the email pattern will be check and Boolean value true will be return if the email pattern is match.

```
//check the valid email address  
private boolean isValidEmail(CharSequence target) {  
    return !TextUtils.isEmpty(target) && android.util.Patterns.EMAIL_ADDRESS.matcher(target).matches();  
}
```

Figure 4.90 Email Validation

In Manage Login, the password pattern will be check. The email must be more than 8 characters & at least 1 number & contain uppercase, lower case and special characters.

```
//check the valid password
private boolean isValidPassword(CharSequence target) {
    if (TextUtils.isEmpty(target)) {
        return false;
    } else {
        /*
         * regex pattern of check Anything with
         * more than eight characters AND
         * al least 1 numbers AND
         * Uppercase AND
         * lowercase AND
         * special characters.
         */
        String regexPattern = "^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z])(?=.*[#?!@$%^&*~]).{8,}$";
        return target.toString().matches(regexPattern);
    }
}
```

Figure 4.91 Password Validation

The decimal validation will validate the value in Edittext whether have the correct digit of the decimal place. Error will prompt if decimal place more than the logic decimal.

```
private boolean decimalValidation(String value, int decimalPlace, EditText editText){
    String[] splitter = value.split(regex: "\\.");
    /*
     * splitter.length ==2; means the total got xx.xx
     * EG: RM 12.34
     * splitter[0] = 12
     * splitter[1] = 34
     * splitter[1].length(); is the number of decimal point Count
     */
    if(splitter.length==2 && splitter[1].length() > decimalPlace){
        editText.setError("Maximum " + decimalPlace+ " decimal place is allow");
        editText.requestFocus();
        return false;
    }

    return true;
}
```

Figure 4.92 Decimal Validation

In Manage Project, the date is validated which the project Start Date must be smaller than the End Date. This is because it is not logic for the project to end before started. The project also must last at least 1 day.

```
private boolean dateValidation(Date startDate, Date endDate) {
    if (endDate.before(startDate)) {
        lblErrorEnd.setVisibility(View.VISIBLE);
        lblErrorEnd.setText("Invalid END date (smaller than START date)");
        lblErrorEnd.requestFocus();
        return false;
    } else if (startDate.equals(endDate)) {
        lblErrorEnd.setVisibility(View.VISIBLE);
        lblErrorEnd.setText("Project must last at least 1 day.");
        lblErrorEnd.requestFocus();
        return false;
    } else {
        lblErrorEnd.setVisibility(View.GONE);
        return true;
    }
}
```

Figure 4.93 Date Validation

## 4.6 Testing of implementation

### 4.6.1 Proof of Testing

We have conducted the testing with the Farmer named Encik Suhaimi Bin Puteh, PSM supervisor Sir Muhammad Zulfahmi Toh bin Abdullah @ Toh Chin Lai and the Project Super (Senior lecturer of Faculty of Manufacturing and Mechatronic Engineering Technology, FTKPM) Dr.Mohd Azraai bin Mohd Razman for the flow and the usability of the Mobile\_AFS.

Some inputs fields have been changed to match the requirements from the farmer. Some technical term like HST, NEG also been implemented based on the requirements from farmer. More module raised for the future like the Notification and Supplements.



Figure 4.94 Proof of Testing with Farmer



## 4.6.2 User Acceptance Test 1

USER ACCEPTANCE TEST				
AUTOMATED CLOUD- BASED FERTIGATION MOBILE APPLICATION (MOBILE-AFS)				
No	Acceptance Requirement	Test Result		Comment
	<b>Login</b>			
1.	Login	(Yes)	No	
2.	Forgot Password	(Yes)	No	
3.	Logout	(Yes)	No	
	<b>Main</b>			
1.	Navigate using bottom navigation	(Yes)	No	
2.	Navigate using button under "Function"	(Yes)	No	
	<b>Profile</b>			
1.	Update profile	(Yes)	No	
	<b>Manage Crops</b>			
1.	Add Crops	(Yes)	No	
2.	View Crops	(Yes)	No	
3.	Update Crops	(Yes)	No	
4.	Delete Crops	(Yes)	No	
5.	Edit Crops Status	(Yes)	No	
6.	View Crops Status	(Yes)	No	
	<b>Manage Sensor</b>			
1.	View Sensor Data	(Yes)	No	
2.	View Sensor Graph	(Yes)	No	
	<b>Manage Timer</b>			
1.	Add Timer	(Yes)	No	
2.	View Timer	(Yes)	No	
3.	Update Timer	(Yes)	No	
4.	Delete Timer	(Yes)	No	
	<b>Manage Pesticide</b>			
1.	Add Pesticide	(Yes)	No	
2.	View Pesticide	(Yes)	No	
3.	Update Pesticide	(Yes)	No	
4.	Delete Pesticide	(Yes)	No	
	<b>Manage Sales</b>			
1.	Add Sales	(Yes)	No	
2.	View Sales	(Yes)	No	
3.	View Edit History	(Yes)	No	
4.	View Sales Graph	(Yes)	No	
5.	Export Sales data to PDF	(Yes)	No	
	<b>Manage Schedule</b>			
1.	View Water Schedule	(Yes)	No	
2.	View Fertilizer Schedule	(Yes)	No	
3.	View Pesticide Schedule	(Yes)	No	
	<b>Manage Project</b>			

Figure 4.95 User Acceptance Test 1 (Part 1)

1.	Add Project	(Yes)	No	
2.	View Project	(Yes)	No	
3.	Update Project	(Yes)	No	
4.	Delete Project	(Yes)	No	
	<b>Manage Purchase</b>			
1.	Add Purchase	(Yes)	No	
2.	View Purchase	(Yes)	No	
3.	Update Purchase	(Yes)	No	
4.	Delete Purchase	(Yes)	No	
	<b>Manage Inventory</b>			
1.	Add Inventory	(Yes)	No	
2.	View individual Inventory	(Yes)	No	
3.	Update Inventory	(Yes)	No	
4.	Delete Inventory	(Yes)	No	
5.	View Total Stock	(Yes)	No	

Comment (Improve/ Design/ Bugs):

Can improve design for User Graph and faster loading time.

Name: DR. MOHD AZRAAI BIN MOHD RAZMAN

Date: 09 / 06 / 2023

I, DR. MOHD AZRAAI BIN MOHD RAZMAN had conducted the User Acceptance Test as requested. I admit that the information that filled is my true personal opinion.



Name: DR. MOHD AZRAAI BIN MOHD RAZMAN

DR. MOHD AZRAAI BIN MOHD RAZMAN  
 PENYAHKAPAN  
 FAKULTI TEKNOLOGI KEAHLIFTERAMAN  
 PEMULUTAN & USAHATRONIK,  
 UNIVERSITI MALAYSIA PAHANG  
 Tel: 09-424 9100 Faksim: 09-424 8888

Figure 4.96 User Acceptance Test 1 (Part 2)

### 4.6.3 User Acceptance Test 2

**USER ACCEPTANCE TEST**  
AUTOMATED CLOUD- BASED FERTIGATION MOBILE APPLICATION (MOBILE-AFS)

No	Acceptance Requirement	Test Result		Comment
	<b>Login</b>			
1.	Login	(Yes)	No	
2.	Forgot Password	(Yes)	No	
3.	Logout	(Yes)	No	
	<b>Main</b>			
1.	Navigate using bottom navigation	(Yes)	No	
2.	Navigate using button under "Function"	(Yes)	No	
	<b>Profile</b>			
1.	Update profile	(Yes)	No	
	<b>Manage Crops</b>			
1.	Add Crops	(Yes)	No	
2.	View Crops	(Yes)	No	
3.	Update Crops	(Yes)	No	
4.	Delete Crops	(Yes)	No	
5.	Edit Crops Status	(Yes)	No	
6.	View Crops Status	(Yes)	No	
	<b>Manage Sensor</b>			
1.	View Sensor Data	(Yes)	No	
2.	View Sensor Graph	(Yes)	No	
	<b>Manage Timer</b>			
1.	Add Timer	(Yes)	No	
2.	View Timer	(Yes)	No	
3.	Update Timer	(Yes)	No	
4.	Delete Timer	(Yes)	No	
	<b>Manage Pesticide</b>			
1.	Add Pesticide	(Yes)	No	
2.	View Pesticide	(Yes)	No	
3.	Update Pesticide	(Yes)	No	
4.	Delete Pesticide	(Yes)	No	
	<b>Manage Sales</b>			
1.	Add Sales	(Yes)	No	
2.	View Sales	(Yes)	No	
3.	View Edit History	(Yes)	No	
4.	View Sales Graph	(Yes)	No	
5.	Export Sales data to PDF	(Yes)	No	
	<b>Manage Schedule</b>			
1.	View Water Schedule	(Yes)	No	
2.	View Fertilizer Schedule	(Yes)	No	
3.	View Pesticide Schedule	(Yes)	No	
	<b>Manage Project</b>			

Figure 4.97 User Acceptance Test 2 (Part 1)

1.	Add Project	(Yes)	No	
2.	View Project	(Yes)	No	
3.	Update Project	(Yes)	No	
4.	Delete Project	(Yes)	No	
<b>Manage Purchase</b>				
1.	Add Purchase	(Yes)	No	
2.	View Purchase	(Yes)	No	
3.	Update Purchase	(Yes)	No	
4.	Delete Purchase	(Yes)	No	
<b>Manage Inventory</b>				
1.	Add Inventory	(Yes)	No	
2.	View individual Inventory	(Yes)	No	
3.	Update Inventory	(Yes)	No	
4.	Delete Inventory	(Yes)	No	
5.	View Total Stock	(Yes)	No	

Comment (Improve/ Design/ Bugs):

All functioning well. Can improve for fast loading time.

Name: Nurul Syafiqah Binti Zaidi

Date: 09 / 06 / 2023

I, Nurul Syafiqah Binti Zaidi had conducted the User Acceptance Test as requested. I admit that the information that filled is my true personal opinion.



Name: Nurul Syafiqah Binti Zaidi

Figure 4.98 User Acceptance Test 2 (Part 2)

## **CHAPTER 5**

### **CONCLUSION**

#### **5.1 Introduction**

As a conclusion, the system meets my requirements in fulfilment of PSM. As can be seen from the problem statement, farmer lack of inventory management practices leads to farmer always frustrated when they are looking for items in the storage. Mobile\_AFS have solve the problem by providing them the complete inventory management system to records the items they bought, the estimated delivery date, price of the item and quantity of the items in the stock.

Another problem statement indicates that farmers are currently facing challenges in effectively managing budgets. With the assist of the Manage Project and Purchase. The farmer now can easily get know to their budget, expense and money left for their project. Farmer can know what money they spend by tracking at the Purchase and look for the image of the receipt they uploaded also.

The last problem statement is the waste of water and fertilizer. Now by having the Manage Crops the farmer now can monitor many types of crops. By knowing the Hari Selepas Tanam (HST) or Day After Planting (DAP), the different water and fertilizer value can be control. The Manage Timer module also enable user to record the exact watering time that needed by the farmer.



## **5.2 Recommendation**

From the User Acceptance Test, we can know the Mobile\_AFS need to have some improvement at the interface design. Therefore, better UI for the graph to let the graph look better and more informative. The slow loading time is due to because of the mobile application is using the free hosting server. For the improvement, we can pay for the premium version or change the paid hosting server that have better loading time like Hostinger.

For the future improvements, we already having some discussion with the farmer and the senior lecturer from the Faculty FTKPM, Dr. Azraai which implements the Notification feature to alert the farmer when the irrigation for pesticide schedule is nearby. This can remind the farmer to spray the pesticide, because the farmer often forgets to do it.

Besides, I also have some thoughts about the sharing Crops and Timers templates in the social media. When the farmer creates their owns Crops and Timer that helps them grow the crops well and healthy, they can share their details on the social media platform like Facebook, WhatsApp, or Telegram. So, when others farmer clicks the link, they will be direct to the app can see the Crops and Timer details. The farmer has the ability to clone the details that share by the sharer.

## REFERENCES

- Agriculture. Department of Statistics Malaysia Official Portal. (2021, November 29). Retrieved October 22, 2022, from [https://www.dosm.gov.my/v1/index.php?r=column%2FctwoByCat&parent\\_id=45&menu\\_id=Z0VTZGU1UHBUT1VJMF1paXRRR0xpdz09#:~:text=The%20contribution%20of%20the%20agriculture,cent%20\(2019%3A%201.5%25\).](https://www.dosm.gov.my/v1/index.php?r=column%2FctwoByCat&parent_id=45&menu_id=Z0VTZGU1UHBUT1VJMF1paXRRR0xpdz09#:~:text=The%20contribution%20of%20the%20agriculture,cent%20(2019%3A%201.5%25).)
- Boman, B., & Obreza, T. (2008). Fertigation Nutrient Sources and Application Considerations for Citrus. Retrieved November 2, 2022, from <https://ucanr.edu/sites/nm/files/76684.pdf>
- Ranjan, Shivani & Sow, Sumit. (2021). Fertigation: An efficient means for fertilizer application to enhance nutrient use efficiency.
- Hagin, J., & Lowengart, A. (1996). Fertigation for minimizing environmental pollution by fertilizers. *Fertilizers and Environment*, 23–25. doi:10.1007/978-94-009-1586-2\_5 [https://doi.org/10.1007/978-94-009-1586-2\\_5](https://doi.org/10.1007/978-94-009-1586-2_5)
- Mahajan, G., & Singh, K. G. (2006). Response of Greenhouse tomato to irrigation and fertigation. *Agricultural Water Management*, 84(1-2), 202–206. doi:10.1016/j.agwat.2006.03.003 <https://doi.org/10.1016/j.agwat.2006.03.003>
- Tranquil. (2023, January 24). 12 common inventory management problems and solutions. TRANQUIL. <https://www.tranquilbs.com/inventory-management-problems/>
- Chhachhar, A. R., & Md Salleh Hassan, M. S. (2012). The use of mobile phone among farmers for Agriculture Development. *International Journal of Scientific Research*, 2(6), 95–98. <https://doi.org/10.15373/22778179/june2013/31>
- Baccara. (2018). Ri Smart Control - Home. ii. Retrieved November 21, 2022, from <https://ii-ri.com/>

Saillog Ltd (2022). Protect your crops. harvest more! Agrio. Retrieved November 21, 2022, from <https://agrio.app/>

Smart Watering. (2021). Irrigation assistant that monitors drip irrigation and fertilization 24/7/365. Smart Watering. Retrieved November 21, 2022, from <https://smart-watering.com/>

Roberts, T. L. (2007). Right product, right rate, right time and right place... the foundation of best management practices for fertilizer. Fertilizer best management practices, 29, 1-8.

Zhu, Z. L., & Chen, D. L. (2002). Nutrient Cycling in Agroecosystems, 63(2/3), 117–127. <https://doi.org/10.1023/a:1021107026067>

Hasnain, M., Chen, J., Ahmed, N., Memon, S., Wang, L., Wang, Y., & Wang, P. (2020). The effects of fertilizer type and application time on soil properties, plant traits, yield and quality of tomato. Sustainability, 12(21), 9065. <https://doi.org/10.3390/su12219065>

DIYIOT. (2021, May 7). Arduino vs ESP8266 vs ESP32 microcontroller comparison. DIYIOT. Retrieved December 1, 2022, from <https://diyi0t.com/technical-datasheet-microcontroller-comparison/>

How To Electronics. (2022, August 20). Sim7600 4G GSM with Arduino: At commands, call, SMS. Retrieved December 1, 2022, from <https://how2electronics.com/using-sim7600-4g-gsm-with-arduino-at-commands-call-sms/>

Mouser Electronics, Inc. (2018, April 27). DFRobot SIM7000C Arduino Expansion Shield. Mouser Electronics, Inc. Retrieved December 1, 2022, from <https://my.mouser.com/new/dfrobot/dfrobot-sim7000c-expansion-shield>



RandomNerdTutorials. (2019, July 26). Dht11 vs dht22 vs LM35 vs DS18B20 vs BME280 vs BMP180. Random Nerd Tutorials. Retrieved December 1, 2022, from <https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/>

Shawn. (2019). Grove - Soil Moisture Sensor. Seeed Studio. Retrieved December 1, 2022, from [https://www.seeedstudio.com/Grove-Moisture-Sensor.html?utm\\_source=blog&utm\\_medium=blog](https://www.seeedstudio.com/Grove-Moisture-Sensor.html?utm_source=blog&utm_medium=blog)

Gurung, G., Shah, R., & Jaiswal, D. P. (2020). Software development life cycle models-A comparative study. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 30–37. <https://doi.org/10.32628/cseit206410>