

**SWEET SCAPE : VENDOR DESSERT MANAGEMENT
SYSTEM**



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

SWEET SCAPE : VENDOR DESSERT MANAGEMENT SYSTEM

NURAQILAH SAFIAH BT KHAIRUL FAIZAL




This report is submitted in partial fulfillment of the requirements for the Bachelor of [Computer Science (Database Management)] with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled
[SWEET SCAPE : VENDOR DESSERT MANAGEMENT SYSTEM]
is written by me and is my own effort and that no part has been plagiarized
without citations.

STUDENT :  Date : 31 August 2024
(NURAQILAH SAFIAH BT KHAIRUL FAIZAL)

اونيورسيتي تيكنيكل مليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

I hereby declare that I have read this project report and found
this project report is sufficient in term of the scope and quality for the award of
Bachelor of [Computer Science (Software Development)] with Honours.

SUPERVISOR :  Date : 31 Aug 2024
(Ts. Fathin Nabilla Md Leza)

DEDICATION

I would like to convey my special dedication,

To my parents,

Thank you for always support me during my bachelor's degree.

To my friends,

Thank you for being through bittersweetness, and happiness during our journey. We have made this journey memorable and enjoyable and blessed to have you guys as one of my support system.

To my lectures,

Thank you for always trusting me. I appreciated it so much.

To my supervisor,

Thank you for your guidance and support. Thank you for being one of the reason why i still continued.

ACKNOWLEDGEMENTS

I would like to thank Ts. Fathin Nabilla Md Leza for giving assistance to complete this project successfully. I am grateful to her for her guidance, advice, and encouragement.

I'd also like to thank my loving parents for their encouragement and support throughout this effort. I would especially like to thank the following mentors and contributors.

Last but not least, I'd want to thank my institution, the Faculty of Information and Communications Technology at Universiti Teknikal Malaysia Melaka (UTeM), my lectures, and my friends for their assistance with my report. This report would not have been possible without their assistance.



ABSTRACT

Sweet Scape : Vendor Dessert Management System is to aims to address the operational challenges faced by sellers managing food and dessert products sourced from vendors, particularly catering to B40, M40, and T20 groups. Currently, sellers manually record product receipts without comprehensive knowledge of quality parameters such as ingredients and expiry dates. This lack of oversight leads to inconsistent product quality and inventory management issues. Sweet Scape proposes a comprehensive system to streamline vendor management, enhance product quality control, and improve customer service through online purchasing capabilities. The system features modules for vendor inventory management, staff supervision, customer orders, administration and reporting using Power BI analytics. By implementing Sweet Scape, sellers and vendors can achieve operational efficiency, ensure product quality consistency, and meet the dynamic demands of the food sector effectively.

ABSTRAK

Sweet Scape: Sistem Pengurusan Pembekal Kuih Bertujuan untuk menangani cabaran operasi yang dihadapi oleh penjual yang menguruskan produk makanan dan pencuci mulut yang diperoleh daripada pembekal, terutamanya untuk kumpulan B40, M40, dan T20. Pada masa ini, penjual merekodkan penerimaan produk secara manual tanpa pengetahuan menyeluruh mengenai parameter kualiti seperti bahan dan tarikh luput. Ketiadaan pengawasan ini menyebabkan ketidakseragaman kualiti produk dan isu pengurusan inventori. Sweet Scape mencadangkan sistem komprehensif untuk menyusun pengurusan pembekal, meningkatkan kawalan kualiti produk, dan meningkatkan perkhidmatan pelanggan melalui kemampuan pembelian dalam talian. Sistem ini merangkumi modul untuk pengurusan inventori pembekal, pengawasan kakitangan, pesanan pelanggan, dan pelaporan pentadbiran menggunakan analitik Power BI. Dengan melaksanakan Sweet Scape, penjual dan pembekal dapat mencapai kecekapan operasi, memastikan konsistensi kualiti produk, dan menangani keperluan dinamik sektor makanan dengan berkesan.

TABLE OF CONTENTS

DECLARATION.....	II
DEDICATION.....	III
ACKNOWLEDGEMENTS.....	IV
ABSTRACT.....	V
ABSTRAK.....	VI
TABLE OF CONTENTS.....	VII
LIST OF TABLES.....	XI
LIST OF FIGURES.....	XV
LIST OF ATTACHMENTS.....	XVII
CHAPTER 1: INTRODUCTION.....	1
1.1 Project Background.....	1
1.2 Problem statement.....	1
1.3 Objective.....	2
1.4 Scope.....	2
1.4.1 Target Users.....	2
1.4.2 Modules.....	3
1.5 Project Significance.....	4
1.6 Expected Output.....	4
1.7 Conclusions.....	4
CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY .	5
2.1 Introduction.....	5
2.2 Project Methodology.....	5

2.3	Project Schedule and Milestone	8
2.4	Conclusion	9
CHAPTER 3: ANALYSIS.....		10
3.1	Introduction.....	10
3.2	Problem Analysis	10
3.3	Requirement Analysis of The to Be System	10
3.3.1	Functional Requirements	11
3.3.2	Non-Functional Requirements.....	12
3.3.3	Other Requirements	12
3.3.3.1	Software Requirements	13
3.3.3.2	Hardware Requirements.....	14
3.4	Conclusions.....	14
CHAPTER 4: DESIGN.....		15
4.1	Introduction.....	15
4.2	System Architecture	15
4.3	Database Design.....	16
4.3.1	Conceptual Design.....	17
4.3.1.1	Data Flow Diagram	17
4.3.2	Logical Design.....	22
4.3.3	Business Rule.....	24
4.3.4	Data Dictionary for Entity Relational Diagram	24
4.3.5	Normalization	33

4.3.6	Query Design	36
4.3.7	Security Mechanism	37
4.4	Graphical User Interface (GUI) Design	38
4.4.1	Input Design.....	38
4.4.2	Output Design	38
4.5	Conclusion	38
CHAPTER 5: IMPLEMENTATION.....		39
5.1	Introduction.....	39
5.2	System Development Environment Setup	39
5.2.1	Steps Of Installation.....	40
5.3	Physical Design.....	44
5.3.1	Data Definition Language (DDL).....	44
	5.3.1.1 Create Table Commands	44
	5.3.1.2 Create Constraints.....	48
5.3.2	Data Manipulation Language	49
5.3.3	Power BI Implementation.....	50
	5.3.3.1 Connection to Database	50
	5.3.3.2 Synthetic Data Generation.....	51
	5.3.3.3 Power BI Dashboard and Measures.....	51
5.4	Conclusions.....	56
CHAPTER 6: TESTING		57
6.1	Introduction.....	57
6.2	Test Plan.....	57

6.2.1	Test Organization.....	57
6.2.2	Test Environment.....	59
6.3	Test Strategy	60
6.3.1	Classes of Tests.....	62
6.4	Test Design	63
6.4.1	Test Description.....	63
6.4.2	Test Data.....	122
6.5	Test Result and Analysis.....	141
6.6	Conclusion	176
	CHAPTER 7: PROJECT CONCLUSION.....	177
7.1	Introduction.....	177
7.2	Observation Weakness and Strengths.....	177
7.2.1	Strength.....	177
7.2.2	Weaknesses.....	177
7.3	Propositions of Improvement.....	177
7.4	Conclusion	178
	REFERENCES.....	179
	APPENDIX A	180

LIST OF TABLES

	PAGE
Table 3.3.3.1.1: Software Requirements	13
Table 3.3.3.2.1: Hardware Requirement Detail.....	14
Table 4.3.2.2.1: Table Customer	24
Table 4.3.2.3.1: Table Admin	25
Table 4.3.2.3.2: Table Payments	26
Table 4.3.2.3.3: Table Product	27
Table 4.3.2.3.4: Table Product Sizes.....	28
Table 4.3.2.3.5: Table Remark.....	28
Table 4.3.2.3.6: Table Category	29
Table 4.3.2.3.7: Table Vendor	30
Table 4.3.2.3.8: Table Cust Orders.....	31
Table 4.3.2.3.9: Table Product Category.....	32
Table 4.3.2.4.1: Query Design Example	36
Table 6.2.1.1: Testing Roles and Responsibilities.....	58
Table 6.2.2.1: Environment Hardware List.....	59
Table 6.2.2.2: Environment Software List	59
Table 6.3.1: Type of Test Design.....	60
Table 6.4.1.1: Test Description of Registration (Customer).....	63
Table 6.4.1.2: Test Description of Login (Customer).....	65
Table 6.4.1.3: Test Description of Reset Password (Customer).....	67
Table 6.4.1.4: Test Description of Order Product (Customer).....	69
Table 6.4.1.5: Test Description of Choose Delivery Method (Customer)	70
Table 6.4.1.6: Test Description of Choose Payment Method (Customer).....	71
Table 6.4.1.7: Test Description of Track Order Status (Customer).....	73
Table 6.4.1.8: Test Description of View and Download Receipt (Customer)	75
Table 6.4.1.9: Test Description of Update Profile (Customer)	78
Table 6.4.1.10: Test Description of Login (Vendor)	80
Table 6.4.1.11: Test Description of View Sales on Dashboard (Vendor).....	82
Table 6.4.1.12: Test Description of Update Profile (Vendor)	84
Table 6.4.1.13: Test Description of Add Product (Vendor)	86

Table 6.4.1.14: Test Description of Update Product (Vendor).....	89
Table 6.4.1.15: Test Description of Delete Product (Vendor).....	91
Table 6.4.1.16: Test Description of Receive Notification for Expired & Low Stock (Vendor)	93
Table 6.4.1.17: Test Description of Receive Request form by Email (Vendor)...	95
Table 6.4.1.18: Test Description of Login (Admin)	97
Table 6.4.1.19: Test Description of View Dashboard Data (Admin).....	98
Table 6.4.1.20: Test Description of Add Vendor (Admin)	100
Table 6.4.1.21: Test Description of Update Vendor (Admin)	103
Table 6.4.1.22: Test Description of Delete Vendor (Admin)	105
Table 6.4.1.23: Test Description of Update Order Status (Admin).....	107
Table 6.4.1.24: Test Description of View Sales Report (Admin)	109
Table 6.4.1.25: Test Description of View Product Sales (Admin)	112
Table 6.4.1.26: Test Description of View Revenue Reports by Payment Method (Admin).....	115
Table 6.4.1.27: Test Description of Backup and Recover Database (Admin)	117
Table 6.4.1.28: Test Description of Power BI Dashboard Integration (Admin) .	119
Table 6.4.2.1: Test Data of Register (Customer)	121
Table 6.4.2.2: Test Data of Login (Customer)	122
Table 6.4.2.3: Test Data of Reset Password (Customer)	123
Table 6.4.2.4: Test Data of Order Product (Customer).....	123
Table 6.4.2.5: Test Data of Delivery Method (Customer)	124
Table 6.4.2.6: Test Data of Choose Payment Method (Customer)	125
Table 6.4.2.7: Test Data of Track Order Status (Customer)	125
Table 6.4.2.8: Test Data of View and Download Receipt (Customer).....	126
Table 6.4.2.9: Test Data of Update Profile (Customer)	126
Table 6.4.2.10: Test Data of Login (Vendor)	126
Table 6.4.2.11: Test Data of View Sales on Dashboard (Vendor).....	127
Table 6.4.2.12: Test Data of Update Profile (Vendor)	128
Table 6.4.2.13: Test Data of Add Product (Vendor)	129
Table 6.4.2.14: Test Data of Update Product (Vendor)	130
Table 6.4.2.15: Test Data of Delete Product (Vendor).....	131
Table 6.4.2.16: Test Data of Receive Notification for Low Stock & Expired	

Products (Vendor)	132
Table 6.4.2.17: Test Data of Receive Request from Admin (Vendor)	132
Table 6.4.2.18: Test Data of Login (Admin)	133
Table 6.4.2.19: Test Data of View Dashboard Data (Admin).....	134
Table 6.4.2.20: Test Data of Add Vendor (Admin)	134
Table 6.4.2.21: Test Data of Update Vendor (Admin)	135
Table 6.4.2.22: Test Data of Delete Vendor (Admin).....	136
Table 6.4.2.23: Test Data of Update Order Status (Admin).....	136
Table 6.4.2.24: Test Data of View Sales Reports (Admin)	136
Table 6.4.2.25: Test Data of View Product Sales (Admin)	137
Table 6.4.2.26: Test Data of View Revenue Reports by Payment Method (Admin)	138
Table 6.4.2.27: Test Data of Backup and Recover Database (Admin)	138
Table 6.4.2.28: Test Data of Power BI Dashboard Integration (Admin)	139
Table 6.5.1: Test Result of Register (Customer)	140
Table 6.5.2: Test Result of Login (Customer).....	142
Table 6.5.3: Test Result of Reset Password (Customer).....	143
Table 6.5.4: Test Result of Order Product (Customer)	144
Table 6.5.5: Test Result of Delivery Method (Customer).....	147
Table 6.5.6: Test Result of Choose Payment Method (Customer).....	148
Table 6.5.7: Test Result of Track Order Status (Customer).....	148
Table 6.5.8: Test Result of View and Download Receipt (Customer)	149
Table 6.5.9: Test Result of Update Profile (Customer).....	150
Table 6.5.10: Test Result of Login (Vendor).....	151
Table 6.5.11: Test Result of View Sales on Dashboard (Vendor)	152
Table 6.5.12: Test Result of Update Profile (Vendor).....	153
Table 6.5.13: Test Result of Add Product (Vendor)	155
Table 6.5.14: Test Result of Update Product (Vendor)	157
Table 6.5.15: Test Result of Delete Product (Vendor)	159
Table 6.5.16: Test Result of Receive Notification for Low Stock & Expired Products (Vendor)	160
Table 6.5.17: Test Result of Receive Request from Admin (Vendor)	161
Table 6.5.18: Test Result of Login (Admin).....	162
Table 6.5.19: Test Result of View Dashboard Data (Admin).....	163

Table 6.5.20: Test Result of Add Vendor (Admin)	164
Table 6.5.21: Test Result of Update Vendor (Admin)	165
Table 6.5.22: Test Result of Delete Vendor (Admin)	166
Table 6.5.23: Test Result of Update Order Status (Admin).....	167
Table 6.5.24: Test Result of View Sales Reports (Admin).....	168
Table 6.5.25: Test Result of View Product Sales (Admin).....	170
Table 6.5.26: Test Result of View Revenue Reports by Payment Method (Admin)	171
Table 6.5.27: Test Result of Backup and Recover Database (Admin)	172
Table 6.5.28: Test Result of Power BI Dashboard Integration (Admin)	17



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF FIGURES

	PAGE
Figure 2.2.1.1: Agile Methodology	6
Figure 2.3.1.1: Gantt Chart.....	8
Figure 4.2.1.1: 3-Tier Architecture.....	16
Figure 4.3.1.1: Context Diagram	17
Figure 4.3.1.2: Data Flow Diagram (DFD) Level 1	18
Figure 4.3.1.3: Data Flow Diagram (DFD) Level 2 (Registration)	19
Figure 4.3.1.4: Data Flow Diagram (DFD) Level 2 (Login).....	19
Figure 4.3.1.5: Data Flow Diagram (DFD) Level 2 (Make Order)	20
Figure 4.3.1.6: Data Flow Diagram (DFD) Level 2 (Make Payment).....	20
Figure 4.3.1.7: Data Flow Diagram (DFD) Level 2 (Manage Product)	21
Figure 4.3.1.8: Data Flow Diagram (DFD) Level 2 (Manage Order)	21
Figure 4.3.1.9: Data Flow Diagram (DFD) Level 2 (Manage Report).....	22
Figure 4.3.2.1: Entity Relationship Diagram.....	23
Figure 4.3.2.2: 3NF of Vendor Table.....	33
Figure 4.3.2.3: 3NF of Table Product_sizes	33
Figure 4.3.2.4: 3NF of Customer Table.....	34
Figure 4.3.2.5: 3NF of Admin Table.....	34
Figure 4.3.2.6: 3NF of Product Table.....	34
Figure 4.3.2.7: 3NF of Cust_orders Table.....	35
Figure 4.3.2.8: 3NF of Remark Table.....	35
Figure 4.3.2.8: Validate Expiry Card Date When Payment By Debit Card.....	37
Figure 5.2.1.1: Download XAMPP for Windows	40
Figure 5.2.1.2: Run the Installer	40
Figure 5.2.1.3: Click Next to Setup Wizard	41
Figure 5.2.1.4: Choose Installation Folder	41
Figure 5.2.1.5 : Finish Setup for XAMPP	42
Figure 5.2.1.6: Installing XAMPP.....	42
Figure 5.2.1.7: Complete XAMPP Setup Wizard.....	43
Figure 5.2.1.8: XAMPP Control Panel.....	43
Figure 5.2.1.9: Start the Server	44
Figure 5.3.1.1: Create Table Cust_orders.....	45

Figure 5.3.1.2: Create Table Admin	45
Figure 5.3.1.3: Create Table Category	46
Figure 5.3.1.4: Create Table Customer	46
Figure 5.3.1.5: Create Table Payment.....	46
Figure 5.3.1.6: Create Table Product	47
Figure 5.3.1.7: Create Table Product_Category	47
Figure 5.3.1.8: Constraints for Table Cust_orders	48
Figure 5.3.1.9: Constraints for Table Product	48
Figure 5.3.1.10: Constraints for Table Product_Sizes	48
Figure 5.3.1.11: Index for Table Product.....	49
Figure 5.3.2.1 : Insert Statement - Insert Data into Table Admin.....	49
Figure 5.3.2.2: Delete Statement - Delete admin from Admin Table	49
Figure 5.3.2.3: Update Statement - Update Product Information.....	50
Figure 5.3.3.1: Schema Map Power BI.....	51
Figure 5.3.3.3.1: Dashboard Sales Power BI	52
Figure 5.3.3.3.2: Dashboard Vendor Power BI	53
Figure 5.3.3.3.3: Dashboard Product Power BI	54
Figure 5.3.3.3.4: Dashboard Customer Power BI	55

LIST OF ATTACHMENTS

	PAGE
Appendix A	
System Interface Design	180-206



CHAPTER 1: INTRODUCTION

1.1 Project Background

In the current era it is going viral where sellers sell food and dessert from vendors consisting of B40, M40 and T20 groups. When the vendor sends food to the seller's store, the seller only records manually the products sent by the vendor. The seller does not even know the quality of the food sent by the vendor in terms of ingredients, expiry dates and so on. The vendor simply dispatches the food and awaits the seller to generate profit from its sale, without being informed about the quantity of desserts sold or any damaged products.

Sweet Scape is a system that allows solving the issue from vendors that present from the B40 and M40 group who can only access the system from their devices. Through this system, the seller can control the quality of the product, the vendor can also know about the stock of products and the customer can make purchases online.

1.2 Problem statement

There are some problems that have been identified for the current system:

- I. Difficult to maintain consistent quality standards for their products sourced from different vendors, leading to variations in quality that impact customer satisfaction. This also causing vendors maintain accurate information about their products, including quantity, expiry dates, and stock availability.

- II. Lack an efficient and integrated system for vendors, seller and customer caused a challenge in managing multiple vendors supplying different types of food products
- III. Difficult to measure to track key performance indicators (KPIs) and sales relevant to their business goal

1.3 Objective

- I. To monitor and maintain high-quality standards for all desserts products supplied by vendors, thereby enhancing customer satisfaction and loyalty.
- II. To improve vendor management tasks, improve collaboration with multiple vendors, and optimize the supply chain to ensure consistent availability of diverse dessert options for customers .
- III. To provide accurate analysis of vendor reports,sales orders using Power BI analytic dashboard for better decision-making and tracking key performance indicators relevant to their business goals.

1.4 Scope

1.4.1 Target Users

The scope of the system is divided into target user and modules.

I. Vendor

- To update their profile
- To allow vendor manage their inventory by adding, updating, and deleting products
- To retrieve and display their sales data.

II. Admin

- To allow admin access reports in Power BI to stay updated
- To update customer & vendor profile
- To allow monitor the inventory and quality of vendor products
- To allow manage sales records.

III. Customers

- Customer is a person who bought the products sold by Sweet Scape. They can browse their menu to order foods and dessert.

1.5 Modules

I. User Management Module

- It allows users to log in securely, update their profile information, and reset passwords if needed.
- Admins have additional capabilities to manage user roles and permissions.
- Vendor can register and update their details

II. Product Catalog Module

- Vendors can describe the details of their products and ingredients, weight, brands, price, number of stock provided.
- Staff check and confirm the product details provided by vendors.
- It enables vendors to add, update, and delete their products.
- Vendor can check their products stock availability
- Customers can browse and search for products they wish to purchase.

III. Order Management module

- Vendors This module facilitates the entire order lifecycle, from order placement to fulfillment.
- Customers can place orders
- Staff receive and process orders, while staff manage inventory level

IV. Communication module

- It includes features such as alerts to keep users informed about important updates, such as inventory updates like product expired.

V. Report module

- Admins can design customized reports and dashboards using Power BI's intuitive interface, incorporating key metrics and visualizations

relevant to their business goals

- Staff members can access real-time insights and trends through Power BI reports, enabling informed decision-making and performance monitoring.
- Vendors can also benefit from Power BI reports tailored to their products, helping them track sales performance and identify areas for improvement.

1.5 Project Significance

The significance of this project is to propose an improved and efficient vendor management system and sales align with small-medium food industry for Sweet Scape. This project centralized all desserts and food that vendor in Sweet Scape offers. Customers can browse the menu and purchase them easily without the need to go out. Vendor also can sell their products by register with Sweet Scape and add their products. By proposing this project, Sweet Scape can approach and attract wider audience as it is easier to browse.

1.6 Expected Output

The expected outcome of Sweet Scape is a system that is easy to use but still systematic where sellers can manage the products stocks availability from their vendors. Customer also can make order anywhere and anytime they want from home.

1.7 Conclusions

In nutshell, this chapter introduces the system that will be developed. It also provides background information about Sweet Scape, a vendor shop that helps small businesses sell their products from home. Before moving forward with the project, this chapter discusses several analyses, including the problem statements, objectives, scope, significance, and expected outcomes. The next chapter will explain the project methodology and planning in detail.

CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY

2.1 Introduction

In this chapter, the existing system has been analyzed. However, there are lack of studies and similar existing systems to the Sweet Scape model. Therefore, the methodology is based on current requirements. Project methodology refers to a set of methods and processes used to develop a project. It typically includes a series of steps and activities for each phase of the project's life cycle. To build the system effectively, the Agile methodology is used. Agile focuses on iterative development, collaboration, and flexibility, allowing the project to adapt to changing requirements and feedback from stakeholders. This section will further explain what Agile is and the phases involved in Agile methodology.

2.2 Project Methodology

Agile methodology elucidates the stages involved in implementing a project from initial planning to continuous improvement. The Agile process is divided into several iterative cycles known as sprints, which typically last 2-4 weeks. Each sprint involves specific phases which is Planning, Design, Development, Testing, Review, and Retrospective.

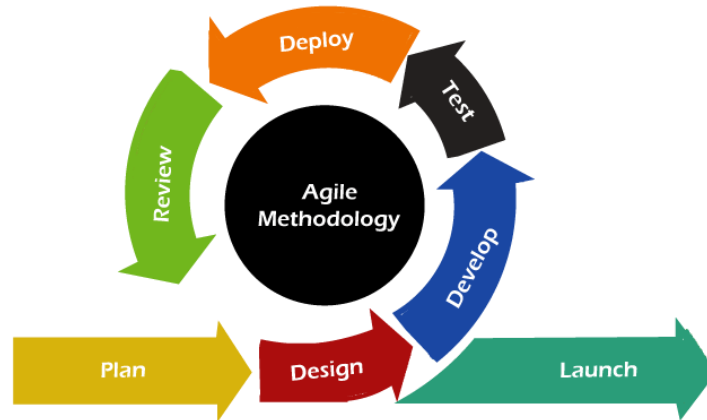


Figure 2.2.1.1: Agile Methodology

I. Planning

The planning phase involves defining the project's objectives, identifying stakeholders, and gathering requirements. The team creates user stories and prioritizes them in a product backlog. Sprint planning sessions are held to determine which user stories will be developed in the upcoming sprint.

II. Design

During the design phase, the team works on the architectural and detailed design of the system. This includes designing the database schema, user interfaces, and system components. Agile emphasizes just-in-time design, which means only the necessary components for the current sprint are designed in detail.

III. Development

In the development phase, the team implements the features planned for the current sprint. Developers write code, create database structures, and integrate various system components. Agile practices such as pair programming, test-driven development (TDD), and continuous integration are often employed to enhance quality and collaboration.

IV. Testing and Evaluation

Testing is a continuous activity in Agile, integrated into each sprint. The team conducts various tests, including unit tests, integration tests, and user acceptance tests (UAT),

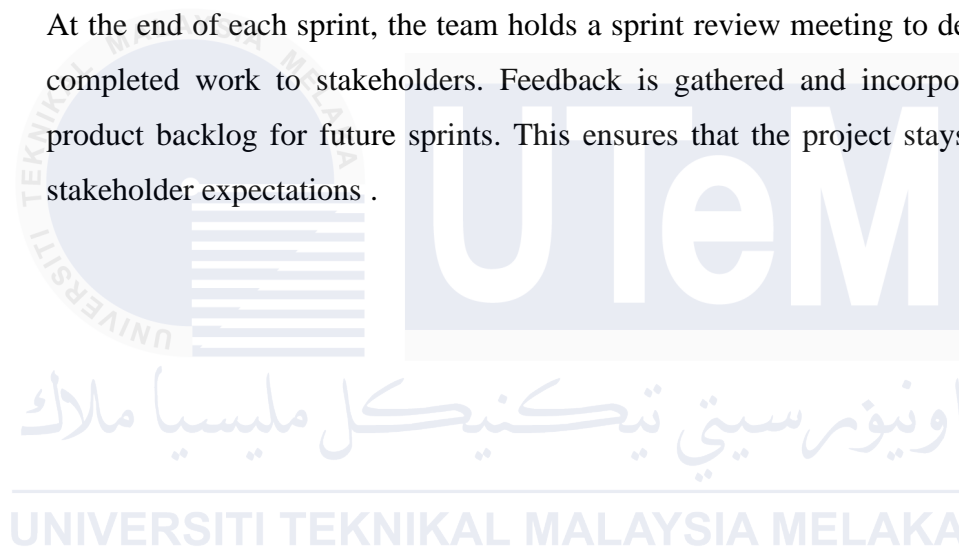
to ensure that the implemented features meet the required standards and function correctly.

V. Deploy

Deployment is performed incrementally at the end of each sprint, ensuring that new features and improvements are continuously delivered to users. Automated deployment processes are used to push updates to staging or production environments, facilitating frequent and reliable releases.

VI. Review

At the end of each sprint, the team holds a sprint review meeting to demonstrate the completed work to stakeholders. Feedback is gathered and incorporated into the product backlog for future sprints. This ensures that the project stays aligned with stakeholder expectations .



2.3 Project Schedule and Milestone

Figure 2.3.1.1 below shows the Gantt chart of Sweet Scape.

Week /	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Task															
Project Initiation	█	█													
Requirement Gathering			█	█	█	█	█	█	█	█	█	█	█	█	█
Database Design						█	█	█	█	█	█	█	█	█	█
Development And Implementation															
Testing And Evaluation															
Demonstration															

Figure 2.3.1.1: Gantt Chart

2.4 Conclusion

The chosen methodology ensures efficiency and flexibility during the development process of Sweet Scape. Agile methodology ensures that the system will deliver a high-quality product that meets user needs in an ever-changing business environment.



CHAPTER 3: ANALYSIS

3.1 Introduction

In the analysis chapter, a detailed explanation of the database development planning will build on the previous chapter. This chapter includes gathering requirements, analyzing current processes, and developing the system architecture. Most developers consider database development while taking into account external factors such as cost, timeframe, development platform, programming language, and expected output. This chapter will also analyze the current system and the future system.

3.2 Problem Analysis

In this current era, there is no system and website for seller who sells products dessert from vendor. So basically customer needs to come to the shop and because of the ‘viral’, they need to queue for buy the products. This situation causes the customer to wait for a period time to get their food.

Sellers face significant challenges in managing food and dessert products from vendors due to manual recording processes and also lack of quality control for the product. The manual recording of product deliveries leads to inefficiencies and potential errors, as vendors also are unaware of their product sales performance or inventory status.

3.3 Requirement Analysis of The to Be System

The requirement analysis will outline the system requirements for each user and the functional requirements.

3.3.1 Functional Requirements

I. Admin

- Manage quality products.
- Manage process orders placed by customers.
- View sales report

II. Vendor

- Manage products.
- Manage inventory products.
- Notify product expired and stock levels are low.
- View sales report by each product.

III. Customer

- Allow customers to browse available products.
- Allow to place orders online
- Review their order history and track the status of their current orders.

3.3.2 Non-Functional Requirements

No	Non-functional	Description
1	Scalability	Appropriate error message to notify user if any input data is invalid to process.
2	Compatibility	The system shall support integration with third-party services, such as payment gateways
3	Reliability	The calculation should be correct .

اونيورسيتي تيكنيكل مليسيا ملاك

3.3.3 Other Requirements


UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The requirements for developing the database system are divided into two categories: software requirements and hardware requirements. Software requirements specify the tools and platforms used during development, while hardware requirements outline the physical equipment and resources needed to support the system.

3.3.3.1 Software Requirements

There have been listed the requirements and specifications of the software components which have been used in this system.

Table 3.3.3.1.1: Software Requirements

Software	Description
Draw.io	Draw.io, also known as diagrams.net, is a free, web-based tool for creating various diagrams, including flowcharts, organizational charts, network diagrams, and UML diagrams.
Microsoft Word 365	Microsoft Word is a word processor developed by Microsoft for writing reports.
MySQL (My structured Query Language)	MySQL is an open-source relational database management system that handles data queries and updates, schema creation and modification, and data access control.
PHP	PHP is a general-purpose programming language initially designed for web development.
Visual Studio Code	Visual Studio Code is a source-code editor used for designing interfaces and implementing code.
XAMPP  XAMPP	XAMPP is a free, open-source, cross-platform web server solution stack package used to run the MySQL database.
Microsoft Power BI Desktop	Power BI is a business analytics tool developed by Microsoft. It enables users to visualize and share insights from their data through interactive reports and dashboards. Power BI helps in data analysis, visualization, and reporting.

3.3.3.2 Hardware Requirements

The list of hardware component that will be used in the system is as shown below in the Table 3.3.3.2.1

Table 3.3.3.2.1: Hardware Requirement Detail

Component	Specification
Model	Acer Swift SF314-54G
Processor	Intel(R) Core (TM) i5-8250 CPU @ 1.60 GHz 1.80 GHz
Hard Drive	HDD 932 GB SSD 477 GB
Memory (RAM)	12 GB
Screen Resolution	1920x 1080, 60 HZ

3.4 Conclusions

The purpose of this chapter includes extensive analysis and research for each criterion that is needed to design the system that is effective, and usable for the end user. When the analysis process is carried out thoroughly by creating each diagram, the system can be constructed systematically. The diagram gives insight into how the system will function and what needs to be done to meet user expectations.

CHAPTER 4: DESIGN

4.1 Introduction

During the design phase, the architecture of the system is defined. The main goal of this chapter is to create a design that fulfills the specified requirements. The requirements identified in the analysis phase are refined and expanded upon to encompass all functionalities of the application as envisioned.

4.2 System Architecture

A three-tier architecture for the Sweet Scape system divides functionality into three distinct layers: the presentation tier, application logic tier, and data tier. The presentation tier handles user interaction through interfaces tailored for vendors, admin, and customers, facilitating tasks such as product management, order processing, and reporting. The application logic tier manages business rules and processes, ensuring that user requests are processed efficiently and consistently. It includes modules for user management, product cataloging, order fulfillment, and real-time reporting using tools like Power BI. The data tier centrally stores and manages all system data in a relational database, supporting operations like storing product details, user profiles, order records, and inventory level.

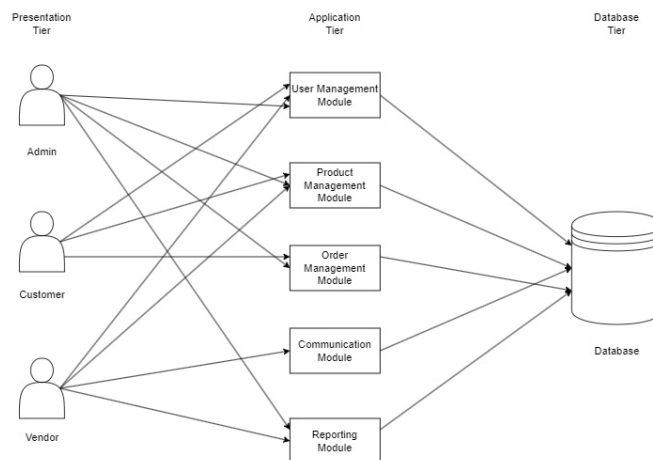


Figure 4.2.1.1: 3-Tier Architecture

4.3 Database Design

Database design to the systematic process of structuring and organizing data within a database system to effectively store, manage, and retrieve information according to specific requirements. This includes defining the database schema, which encompasses tables, fields, relationships, and constraints, tailored to the chosen data model. The objective is to create a well-structured database that supports the application's functionality, ensures efficient data handling, and maintains data accuracy and consistency. Efficient database design is critical for achieving long-term data reliability, scalability, and ease of maintenance throughout the system's lifecycle.

4.3.1 Conceptual Design

4.3.1.1 Data flow diagram

a) Context Diagram

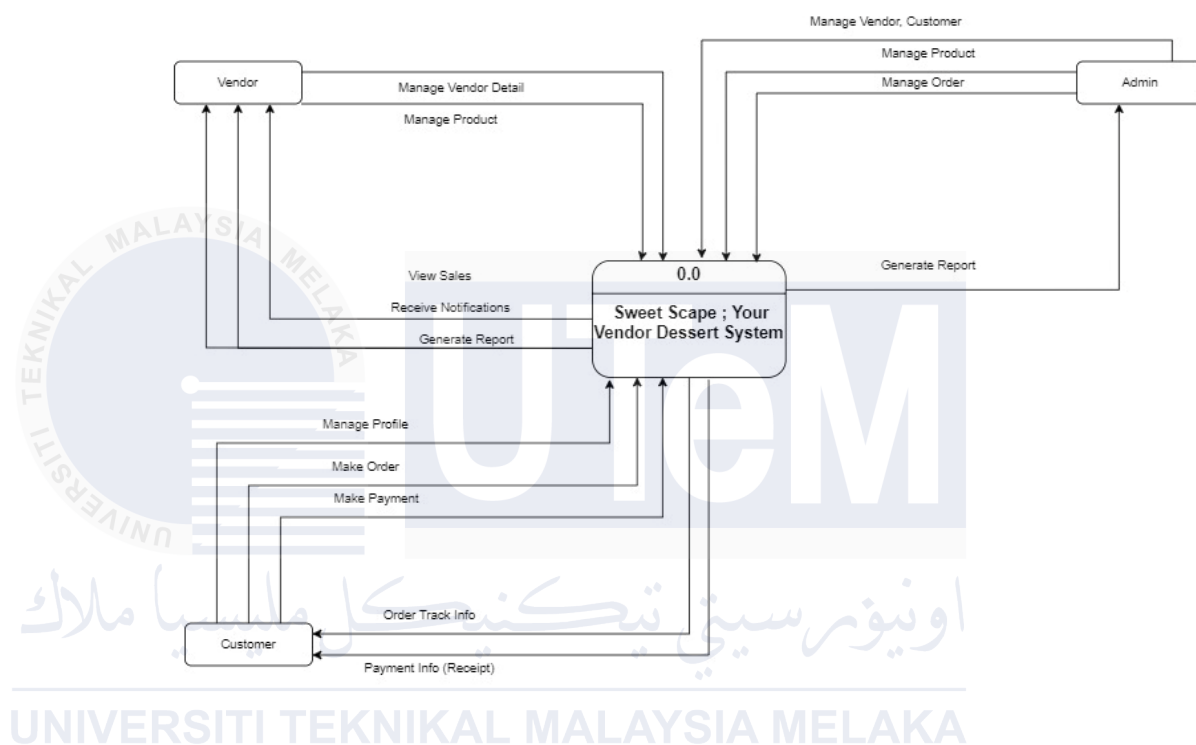


Figure 4.3.1.1: Context Diagram

Figure 4.3.1.1 above shows the context diagram which shows the overall look of the system and its user function to the system.

b) Data Flow Diagram Level 1

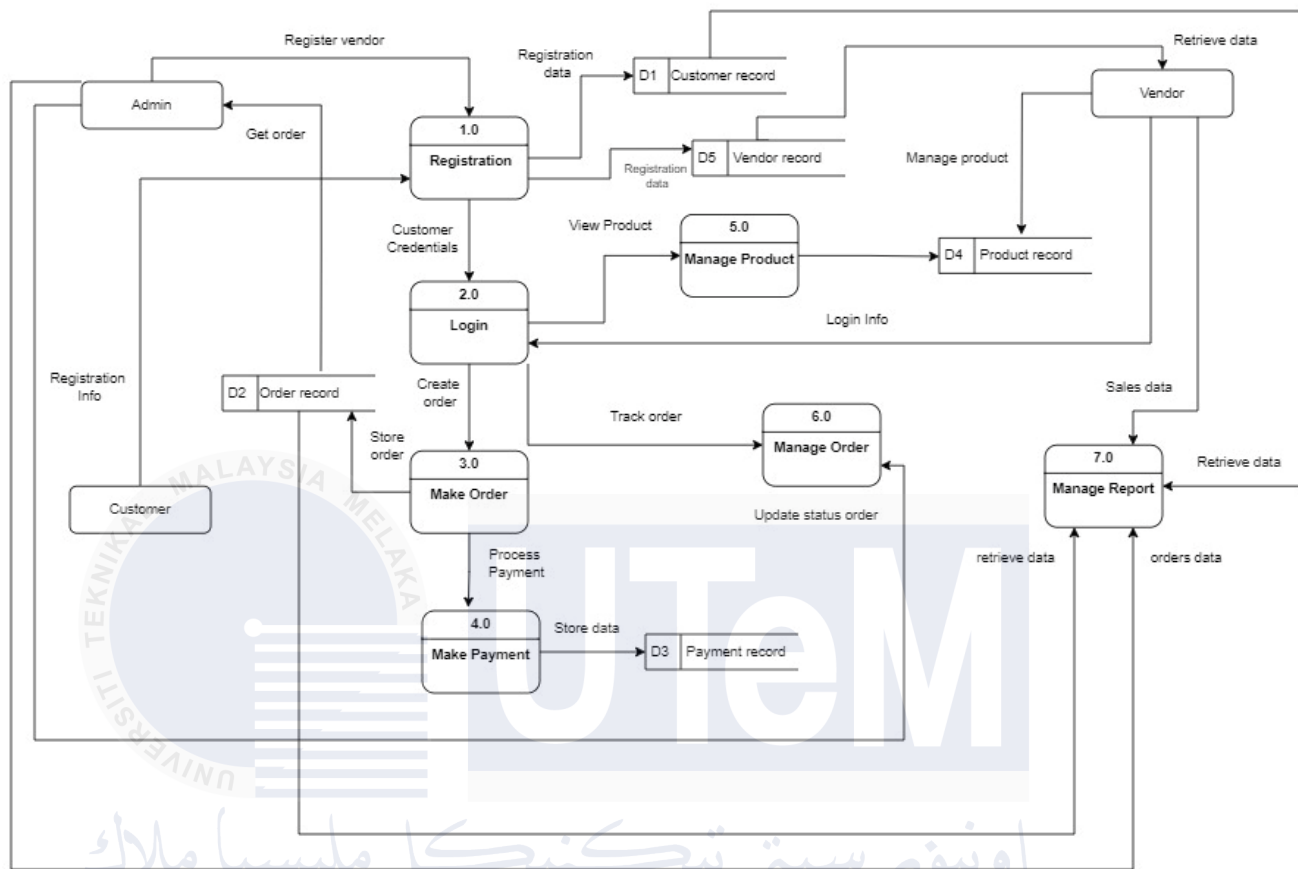


Figure 4.3.1.2: Data Flow Diagram (DFD) Level 1

The figure above provides a detailed breakdown of the Context Diagram, highlighting the main functions of the system. It shows how the high-level process of the Context Diagram is divided into its sub-processes.

c) *Data Flow Diagram (DFD) Level 2*

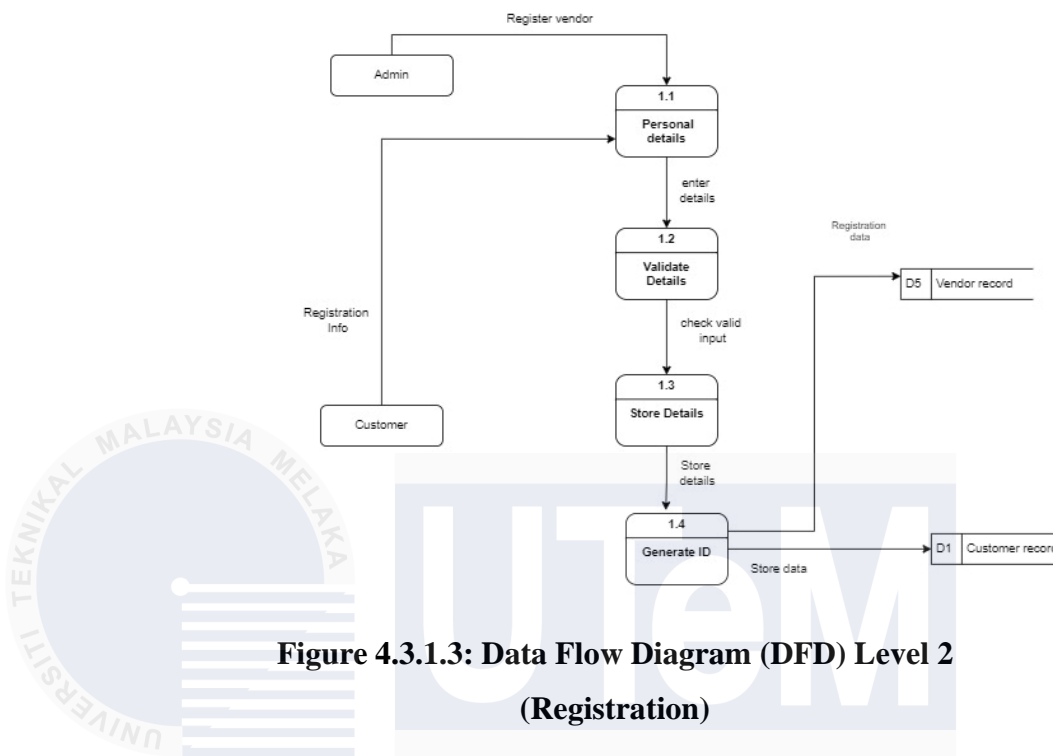


Figure 4.3.1.3: Data Flow Diagram (DFD) Level 2 (Registration)

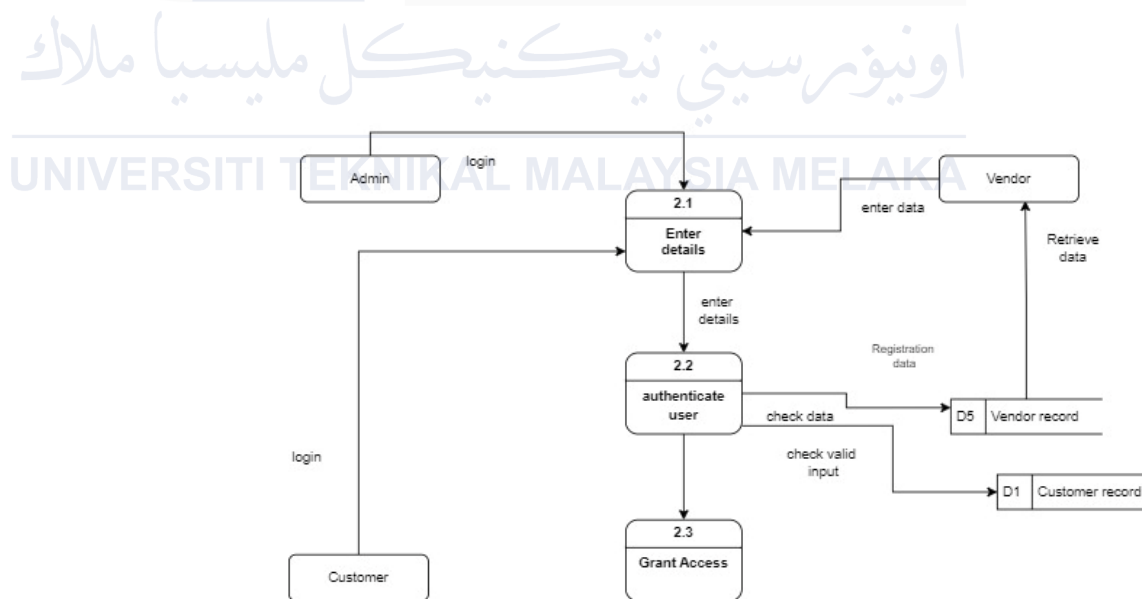


Figure 4.3.1.4: Data Flow Diagram (DFD) Level 2 (Login)

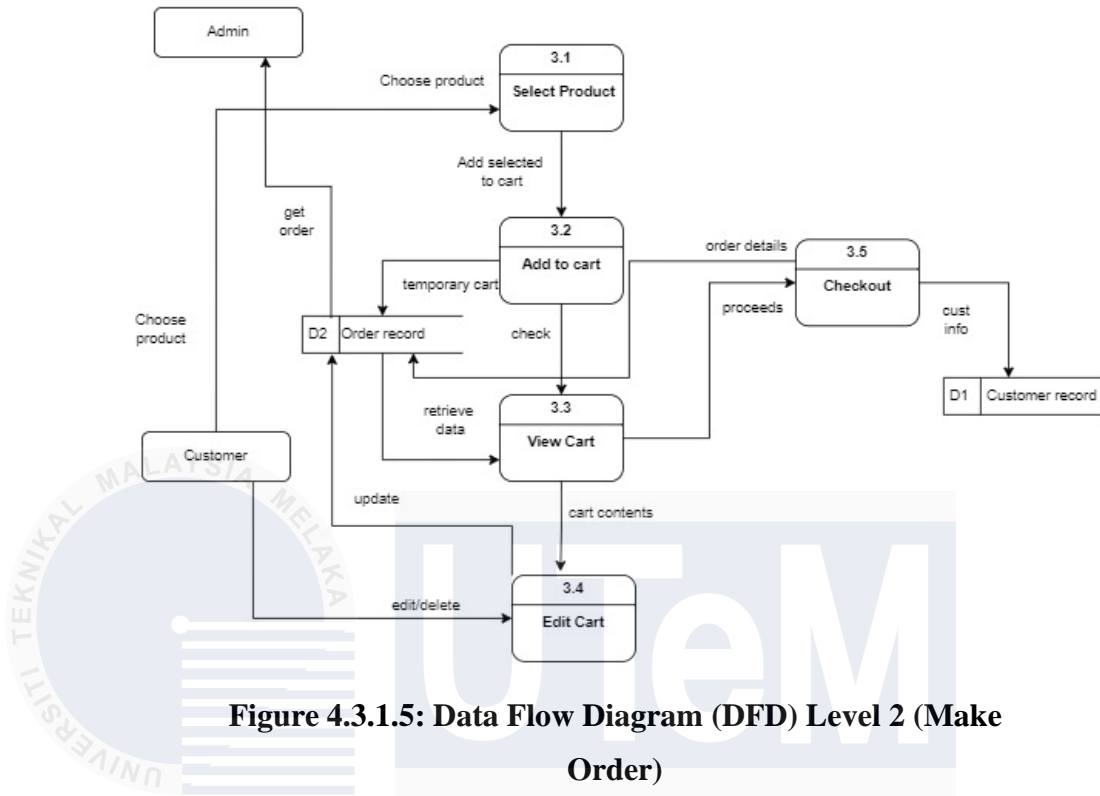


Figure 4.3.1.5: Data Flow Diagram (DFD) Level 2 (Make Order)

اونيورسيتي تيكنيكل مليسيا ملاك

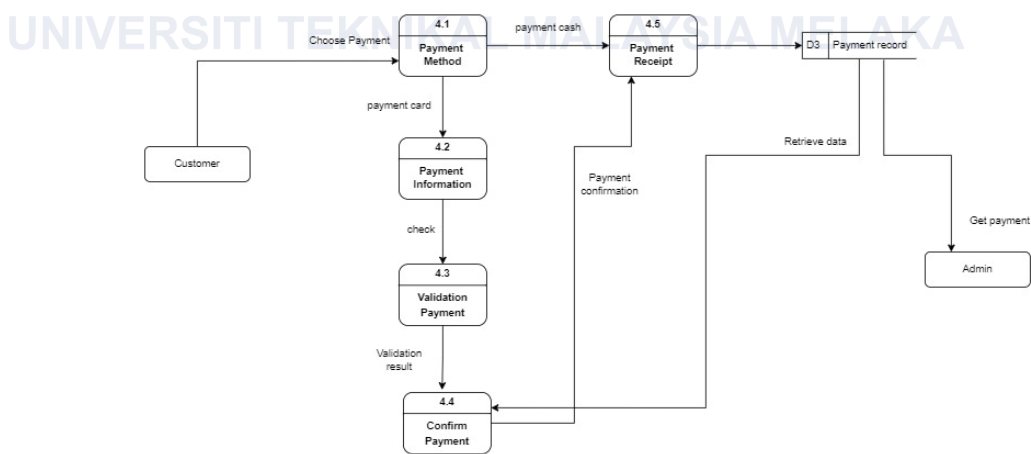
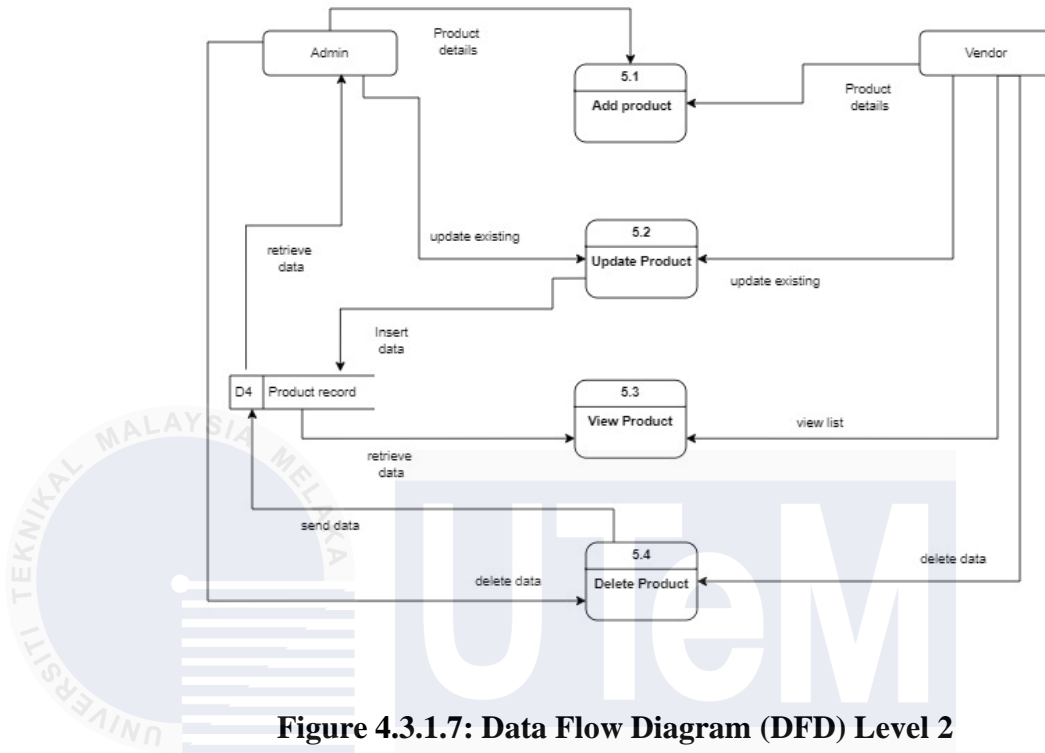
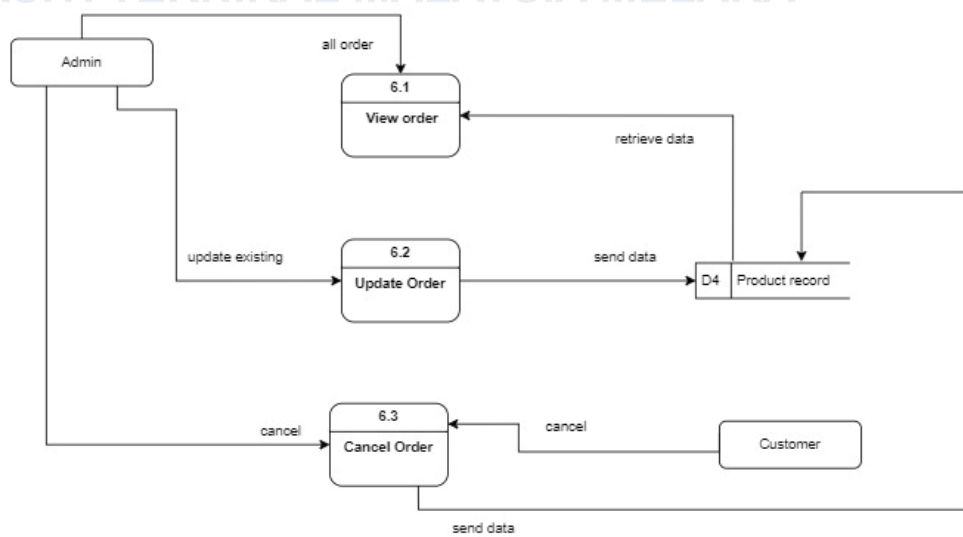


Figure 4.3.1.6 Data Flow Diagram (DFD) Level 2 (Make Payment)

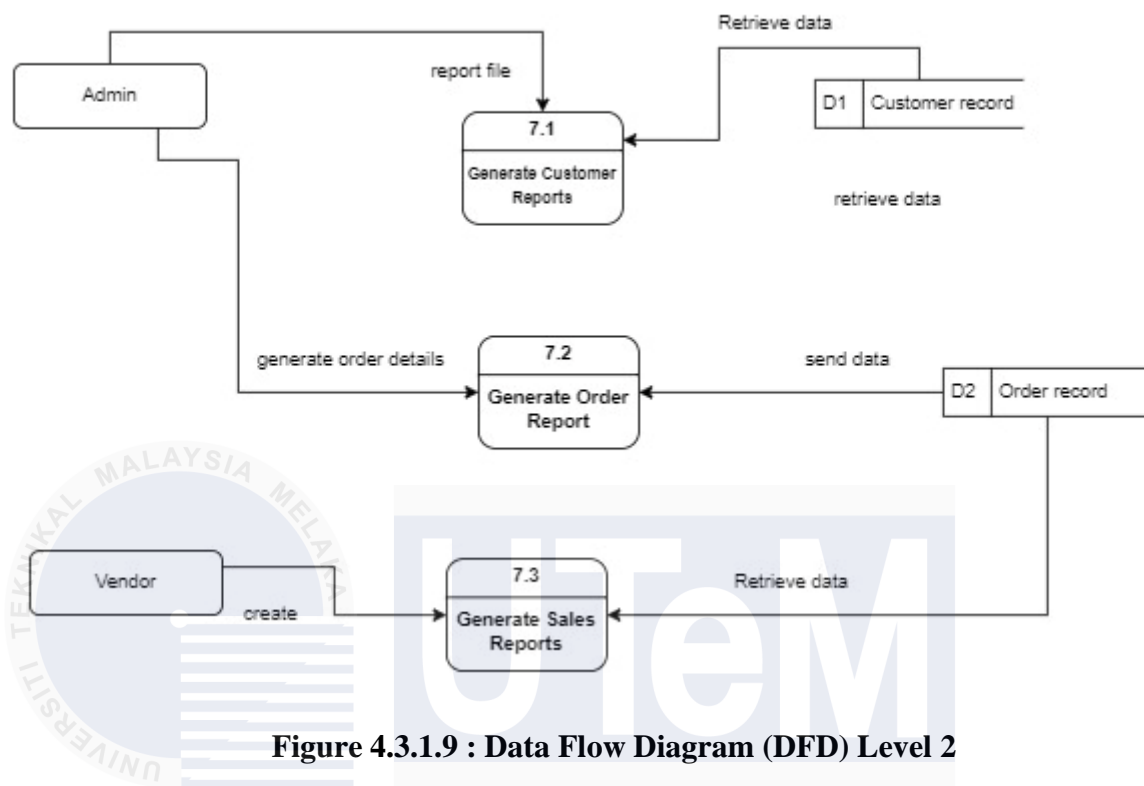


**Figure 4.3.1.7: Data Flow Diagram (DFD) Level 2
(Manage Product)**

اونيورسيٲي بيٲيكيكل مليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA



**Figure 4.3.1.8: Data Flow Diagram (DFD) Level 2
(Manage Order)**



**Figure 4.3.1.9 : Data Flow Diagram (DFD) Level 2
(Manage Report)**

4.3.2 Logical Design

Logical design lays the foundation for the physical implementation of the database schema in a specific DBMS. It ensures that the database structure is well-organized, normalized, and capable of efficiently supporting the application's requirements while adhering to best practices in database design.

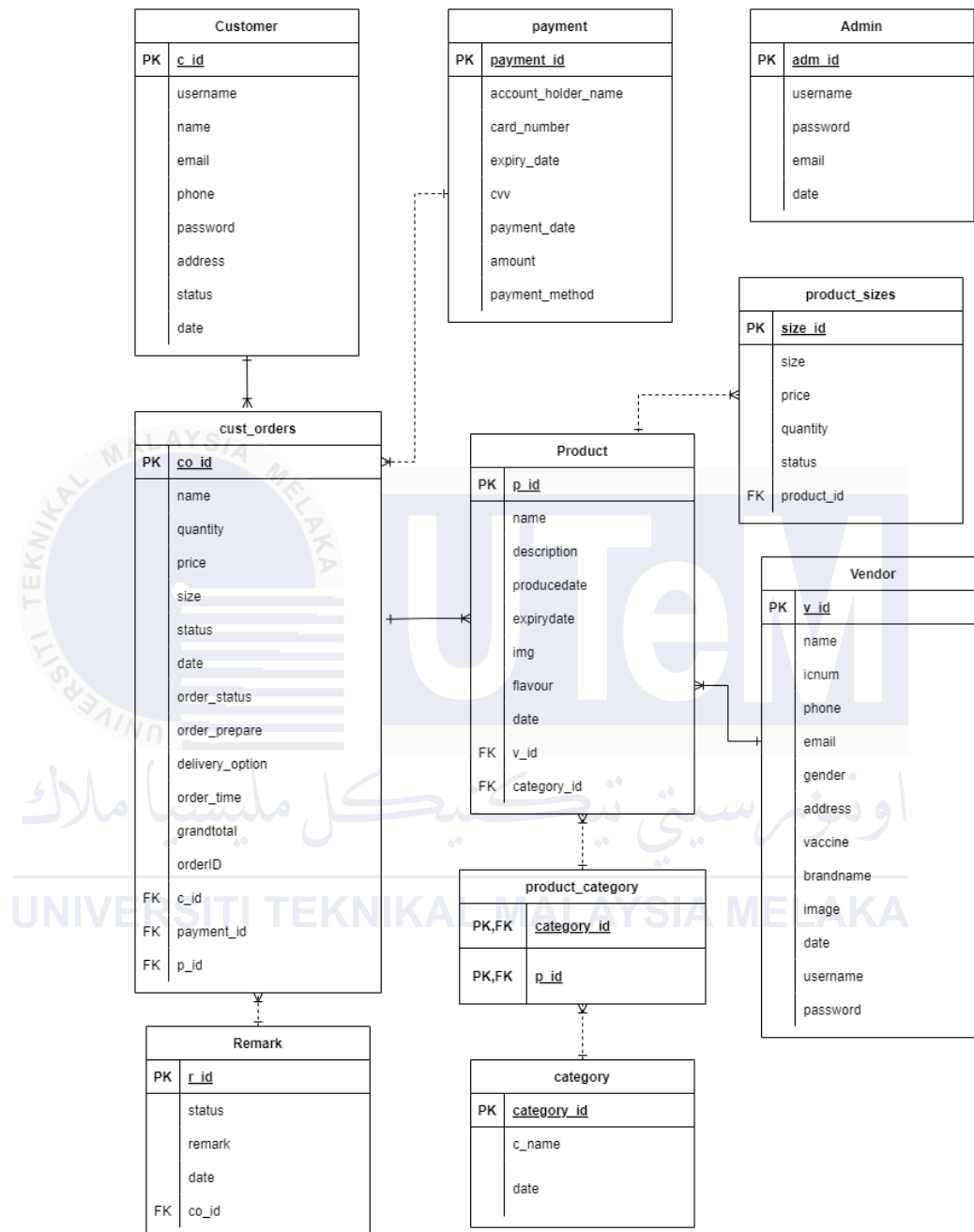


Figure 4.3.2.1 : Entity Relationship Diagram

Figure 4.3.2.1 shows the Entity Relationship Diagram (ERD) for Sweet Scape.

4.3.2.1 Business Rule

1. Vendor can have one or many products.
2. Product is owned by one vendor.
3. Customer can have one or many custs order and each order belong to one customer.
4. One product can have multiple category.
5. Admin can manage one or many order status tracking.
6. Vendor can manage one or many products.
7. Each customer order have a valid payment method.

4.3.2.2 Data Dictionary for Entity Relational Diagram

I. Customer Table

This table keep each customer detail information.

Table 4.3.2.2.1: Table Customer

Column	Type	Constraints	Default
c_id	Int (11)	Not null	Auto increment
Username	Varchar (222)	Not null	-
Name	Varchar (222)	Not null	
Email	Varchar (222)	Not null	
Phone	Varchar (222)	Not null	
Password	Varchar (222)	Not null	

Address	Text	Not null	
Status	Int (11)	Not null	
Date	Date	Not null	Current_timestamp()

II. Admin Table

This table keeps admin detailed information.

Table 4.3.2.2.2: Table Admin

Column	Type	Constraints	Default
Adm_id	Int (11)	Not null	Auto increment
Username	Varchar (222)	Not null	
Password	Varchar (222)	Not null	
Email	Varchar (222)	Not null	
Code	Varchar (222)	Not null	
Date	Timestamp	Not null	Current_timestamp

III. Payments Table

This table keeps card_payments information.

Table 4.3.2.2.3: Table Payments

Column	Type	Constraints	Default
Payment_id	Int (11)	Not null	Auto increment
Amount	Decimal(10,2)	Not null	
Payment_date	date	Not null	
Payment_method	Varchar (50)	Not null	
Card_number	Varchar (20)	Null	
Expiry_date	date	Null	
cvv	Varchar(5)	Null	
Co_id	Int(11)	Null	

IV. Product Table

This table keeps product detailed information.

Table 4.3.2.2.4: Table Product

Column	Type	Constraints	Default
p_id	Int (11)	Not null	Auto increment
Name	Varchar (222)	Not null	
Description	Text	Not null	
Producedate	Date	Not null	
Expirydate	Date	Not null	
Img	Varchar (222)	Not null	
Flavour	Text	Not null	
Date	Date	Not null	Current_timestamp()
V_id	Int (11)	Not null	Foreign key
Category_id	Int (11)	Not null	Foreign key

V. Product_sizes Table

Table keeps product_sizes detailed information.

Table 4.3.2.2.5: Table Product_sizes

Column	Type	Constraints	Default
size_id	Int (11)	Not null	Auto increment
Size	Varchar (50)	Not null	
Price	Decimal (10,2)	Not null	
Quantity	Int (11)	Not null	
Status	Varchar (100)	Not null	
Product_id	Int (11)	Not null	Foreign key

VI. Remark Table

This table keeps remark detailed information.

Table 4.3.2.2.6: Table Remark

Column	Type	Constraints	Default
r_id	Int (11)	Not null	Auto increment
Frm_id	Int (11)	Not null	

Status	Varchar (255)	Not null	
Remark	Mediumtext	Not null	
Date	Date	Not null	Current_timestamp()

VII. Category Table

This table keeps pro_category detailed information.

Table 4.3.2.2.7: Table Category

Column	Type	Constraints	Default
Category_id	Int (11)	Not null	Auto increment
C_name	Varchar(222)	Not null	
Date	Date	Not null	Current_timestamp()

VIII. Vendor Table

This table keeps vendor detailed information.

Table 4.3.2.2.8: Table Vendor

Column	Type	Constraints	Default
V_id	Int (11)	Not null	Auto increment
Name	Varchar (222)	Not null	
Icnum	Varchar (222)	Not null	
Phone	Varchar (222)	Not null	
Email	Varchar (222)	Not null	
Gender	Varchar(222)	Not null	
Address	Text	Not null	
Vaccine	Varchar (100)	Not null	
Brandname	Varchar (222)	Not null	
Image	Text	Not null	
Date	Datetime	Not null	Current_timestamp()
Username	Varchar (255)	Not null	
password	Varchar (255)	Not null	

IX. Cust_orders Table

This table keeps cust_orders detailed information.

Table 4.3.2.2.9: Table cust_orders

Column	Type	Constraints	Default
co_id	Int (11)	Not null	Auto increment
Name	Varchar (222)	Not null	
Quantity	Int (11)	Not null	
Price	Decimal (10,2)	Not null	
Size	Varchar (30)	Not null	
Status	Varchar (222)	Not null	
Date	Datetime	Not null	Current_timestamp()
Order_status	enum('pending', 'completed')	Null	Pending
Order_prepare	Varchar (222)	Not null	
Delivery_option	Varchar (10)	Not null	
Payment_method	Varchar (30)	Not null	
Order_time	Datetime	Not null	Current_timestamp()
Grandtotal	Decimal (10,2)	Null	

orderID	Varchar (100)	Not null	
C_id	Int (11)	Not null	Foreign key

X. Product_category Table

This table keeps product_category detailed information

Table 4.3.2.2.10: Table Product_category

Column	Type	Constraints	Default
V_id	Int (11)	Null	Foreign Key
Category_id	Int (11)	Null	Foreign Key

4.3.2.3 Normalization

The conceptual design uses normalization, as displayed. Each table includes attributes, primary keys, and foreign keys. Figures 4.3.2.3.1 to 4.3.2.3.8 illustrate the third normal form (3NF) of the Sweet Scape system.

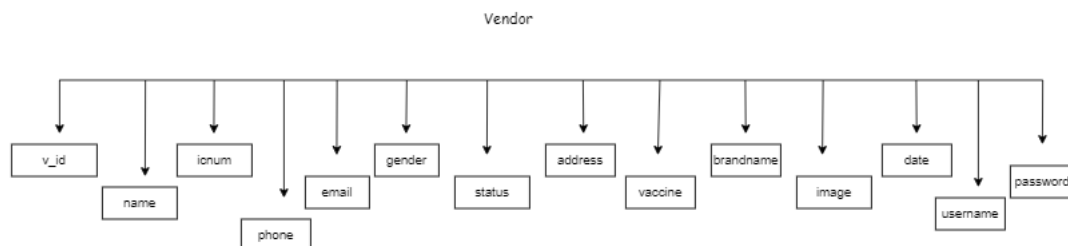


Figure 4.3.2.3.1: 3NF of Vendor Table

Figure 4.3.2.2 illustrates that table vendor is in 3NF.

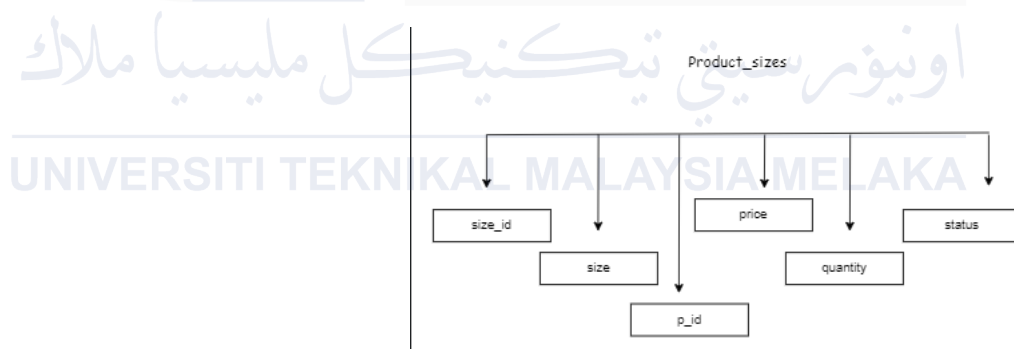


Figure 4.3.2.3.2: 3NF of Table Product_sizes

Figure 4.3.2.3 illustrates that table product_sizes is in 3NF.

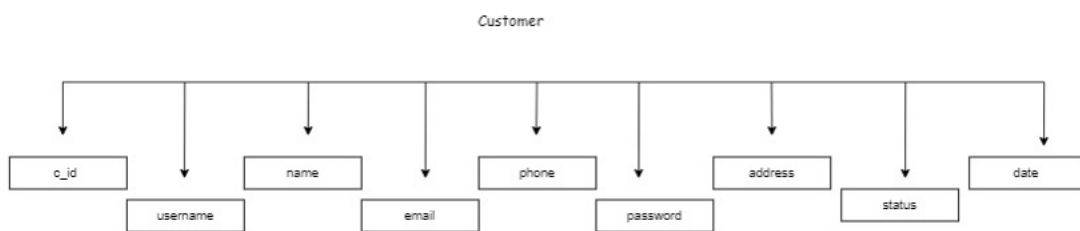


Figure 4.3.2.3.3: 3NF of Customer Table

Figure 4.3.2.4 illustrates that the customer table is in 3NF.

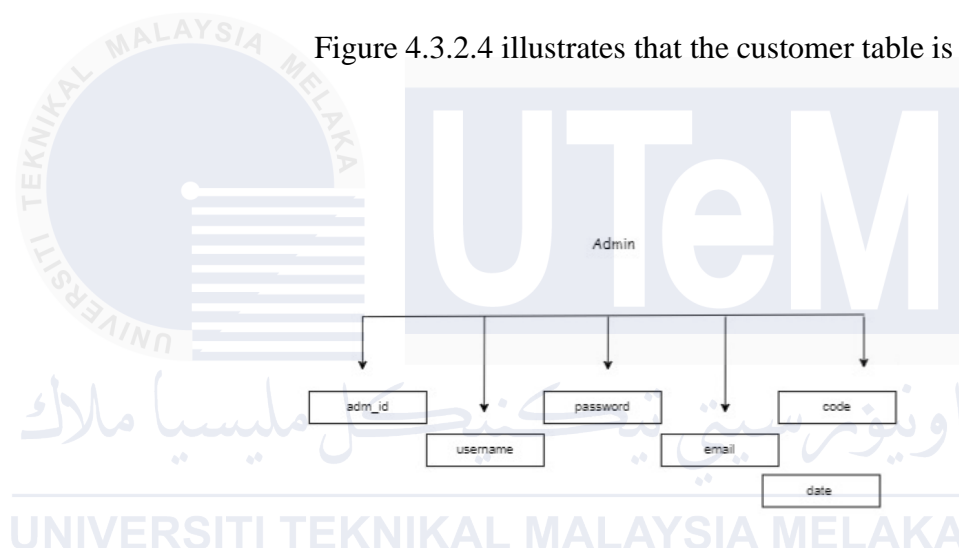


Figure 4.3.2.3.5: 3NF of Admin Table

Figure 4.3.2.5 illustrates that the Admin table is in 3NF.

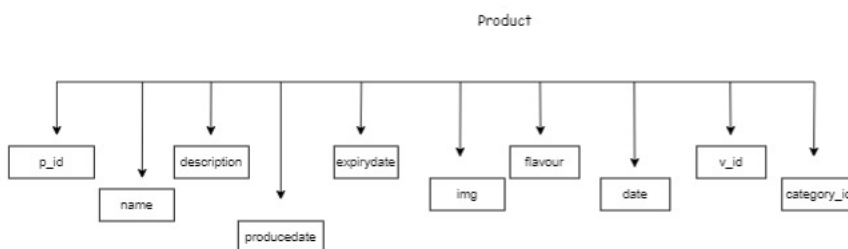


Figure 4.3.2.3.6 : 3NF of Product Table

Figure 4.3.2.6 illustrates that the product table is in 3NF

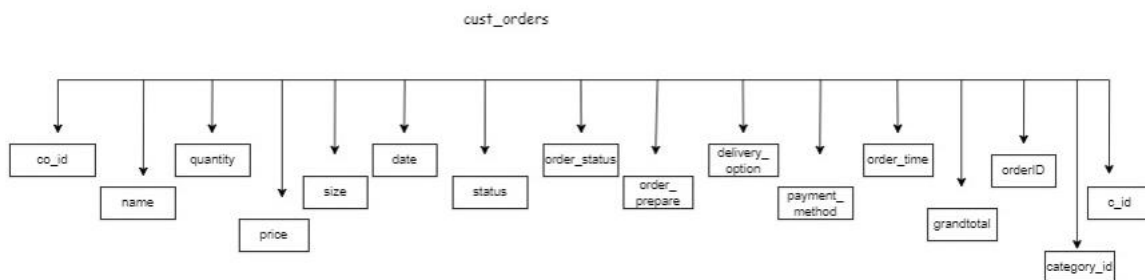


Figure 4.3.2.3.7: 3NF of Cust_orders Table

Figure 4.3.2.7 illustrates that the cust_orders table is in 3NF.

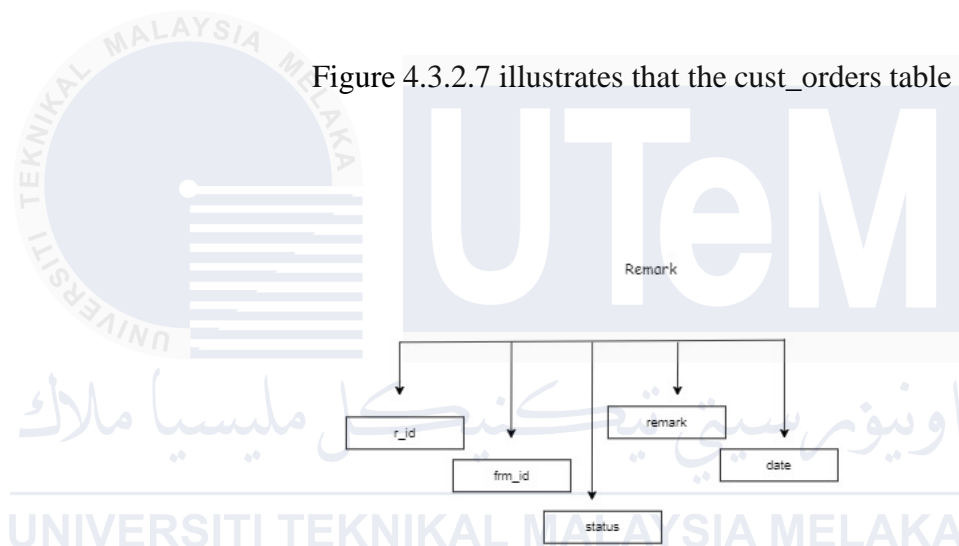


Figure 4.3.2.3.8: 3NF of Remark Table

Figure 4.3.2.8 illustrates that the remark table is in 3NF.

4.3.2.4 Query Design

This various query designs can generate diverse types of outputs, each tailored to specific requirements, rationale, and objectives. Table 4.3.2.4.1 presents several examples of query design.

Table 4.3.2.4.1: Query Design Example

Type of Query	Query	Explanation
Simple Query	UPDATE product SET flavour = 'Chocolate'	Update table product and set flavour = chocolate.
Join Table Query	SELECT customer.*, cust_orders.* FROM customer INNER JOIN cust_orders ON customer.c_id=cust_orders.c_id	To retrieve the customer and customer's order to confirm their order.
Aggregate Query and Join Query	SELECT orderID, GROUP_CONCAT(name SEPARATOR ' , ') AS names, SUM(quantity) AS total_quantity, GROUP_CONCAT(size SEPARATOR ' , ') AS sizes, SUM(price) AS total_price, delivery_option, payment_method, status, MIN(date) AS min_date FROM cust_orders WHERE c_id = '{\$_SESSION['c_id']}'	To retrieve customer's order data to show their orders.

	GROUP BY orderID ORDER BY min_date ASC";	
--	--	--

4.3.2.5 Security Mechanism

Security mechanism is used for the system to validate the expiry card. For example, the customer that choose to payment by card need to fill up the card details.

```

// Function to validate expiry date
function validateExpiryDate($expiry_date) {
    // Check if the format is MM/YYYY
    if (preg_match("/^(0[1-9]|1[0-2])\.[0-9]{4}$/", $expiry_date)) {
        // Extract month and year
        $parts = explode('.', $expiry_date);
        $expiry_month = $parts[0];
        $expiry_year = $parts[1];

        // Get the current month and year
        $current_month = date('m');
        $current_year = date('Y');

        // Check if the card is expired
        if ($expiry_year > $current_year || ($expiry_year == $current_year && $expiry_month >= $current_month)) {
            return true; // Card is not expired
        }
    }
    return false; // Card is expired or format is invalid
}
?>

```

Figure 4.3.2.8: Validate Expiry Card Date When Payment By Card

4.4 Graphical User Interface (GUI) Design

A graphical user interface (GUI) design outlines how users will interact with the system and the types of inputs the system accepts and produces. It encompasses screen displays that guide navigation through the system and capture data. GUI design consists of three components: navigation design, input design, and output design.

4.4.1 Input Design

The input design emphasizes how users input data into the system, whether structured or unstructured. Screens and forms are specifically created to capture and store information related to actions performed within the system. Figure illustrates the input design, can be referred to in Appendix A.

4.4.2 Output Design

The output design focuses on displaying retrieved information from the system on the screen or form. Details of the output design are shown in Appendix A.

4.5 Conclusion

This chapter concludes with insights into designing the Sweet Scape in a systematic manner, ensuring compliance with both functional and non-functional requirements identified in the analysis phase. Additionally, the design phase aims to address the problem defined in the requirement document, transitioning from the problem domain to the solution domain.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

This chapter covers the database implementation process, focusing on installing and configuring MySQL on the Windows 11 platform. It details the execution of Data Definition Language (DDL) and Data Manipulation Language (DML) SQL statements within the database implementation phase. The status of implementation is documented for each module.

5.2 System Development Environment Setup

In Sweet Scape, setting up the software development environment is crucial before starting web application development. This setup generally involves four main components: a Web Server (such as Apache), a Database Server (like MySQL), a Web Programming Language (such as PHP), and a Database Management Tool (like phpMyAdmin). These components can be conveniently installed together using XAMPP.

5.3.3 Steps Of Installation

Step 1: Open your web browser and go to <https://www.apachefriends.org/download.html> to download XAMPP for Windows.

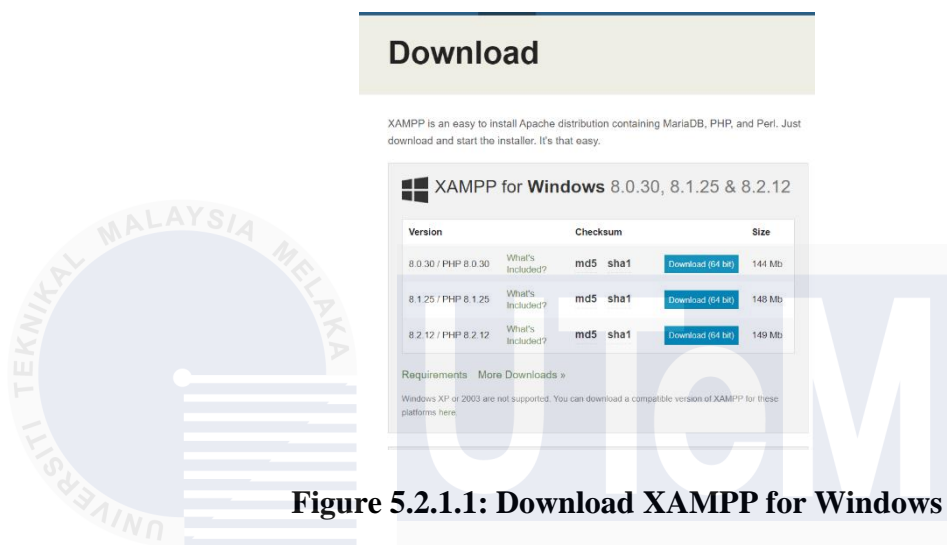


Figure 5.2.1.1: Download XAMPP for Windows

Step 2: Click the installer exe file and click run.

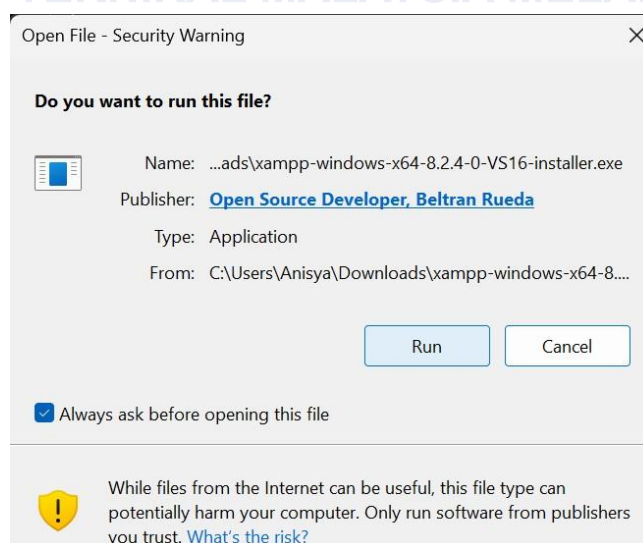


Figure 5.2.1.2: Run the Installer

Step 3: After the installer finishes running, click Next to configure the XAMPP component.

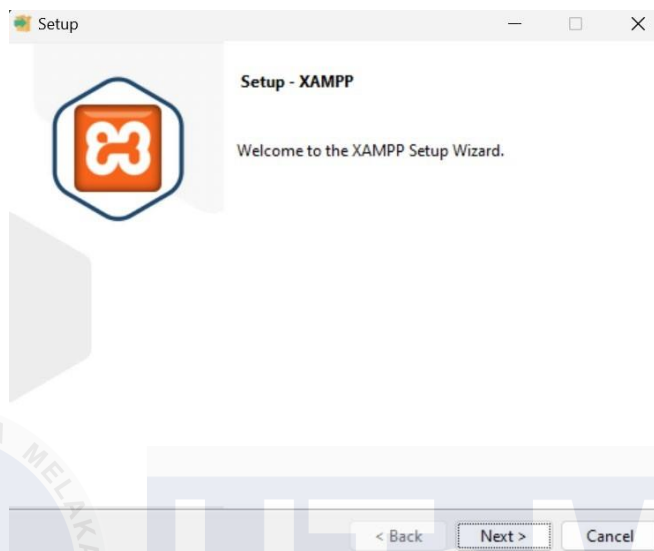


Figure 5.2.1.3: Click Next to Setup Wizard

Step 4: Choose a folder to install XAMPP, then click Next.

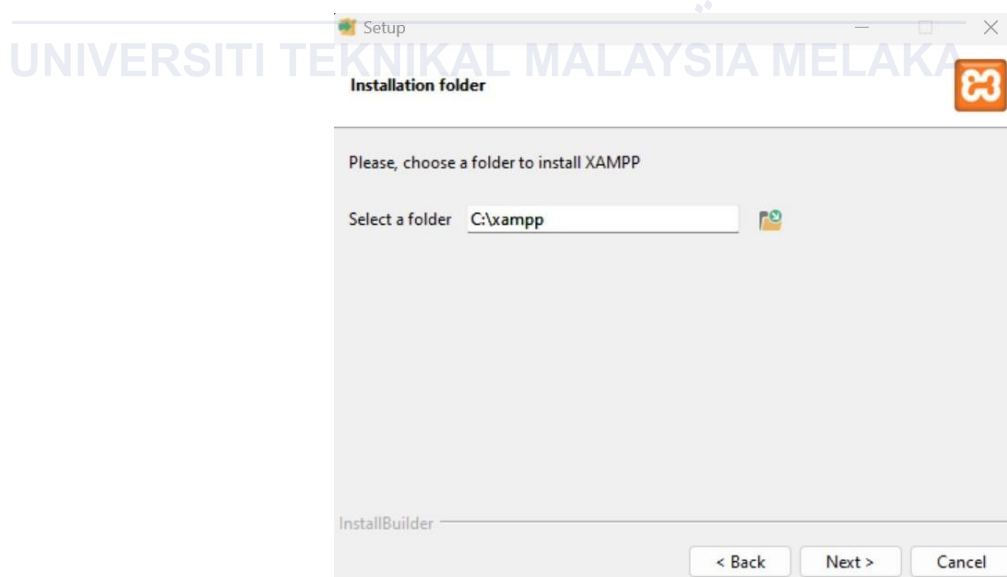


Figure 5.2.1.4 : Choose Installation Folder

Step 5: Then after the Setup is finish, the software will be ready to install. Click Next to perform installation.

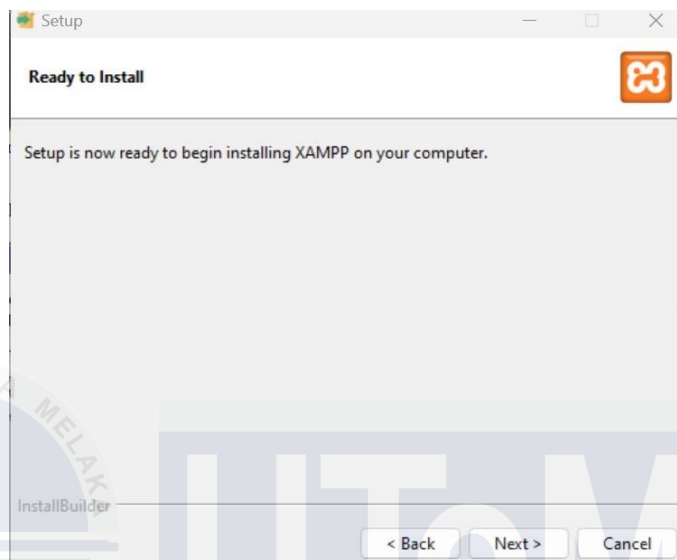


Figure 5.2.1.5 : Finish Setup for XAMPP

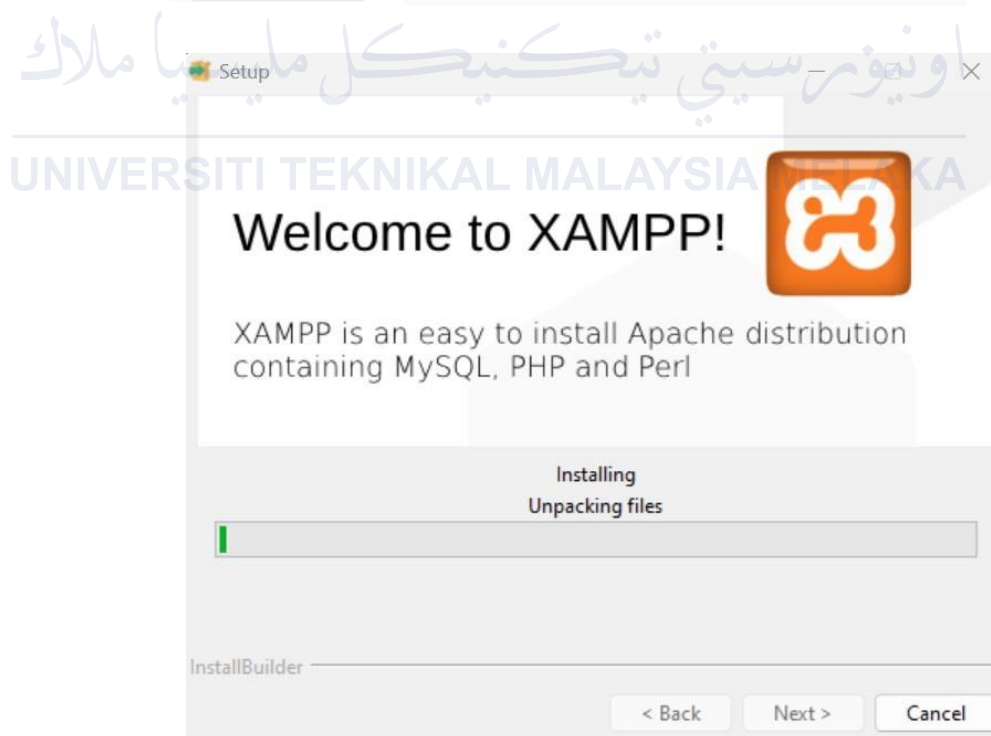


Figure 5.2.1.6 : Installing XAMPP

Step 6: After installation is finished, Click Finish.

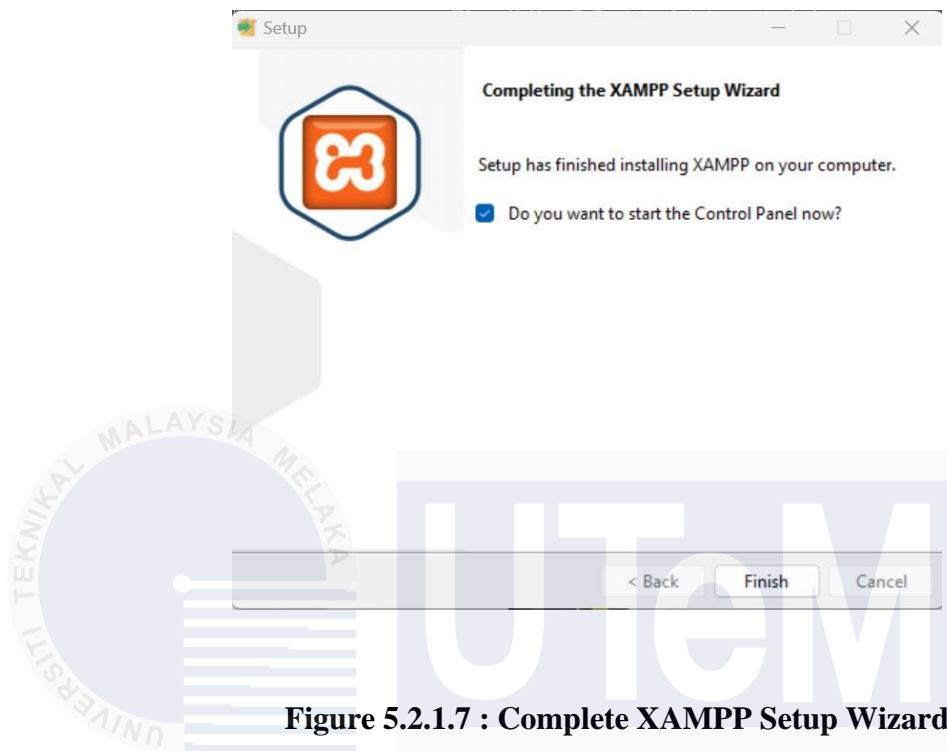


Figure 5.2.1.7 : Complete XAMPP Setup Wizard

Step 7: After closing the setup page, it will appear XAMPP Control Panel.

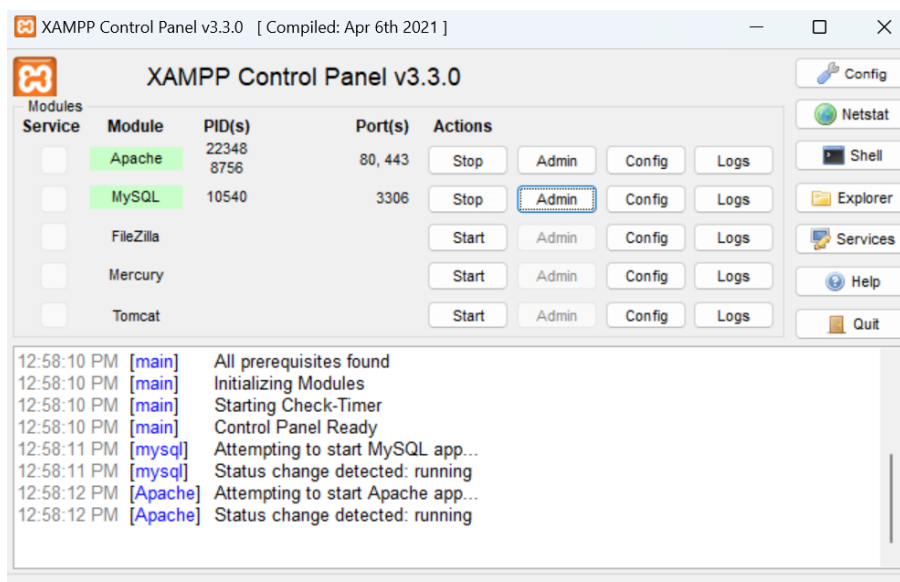


Figure 5.2.1.8 : XAMPP Control Panel

Step 9: Click Start on Apache and MySQL to start the server.

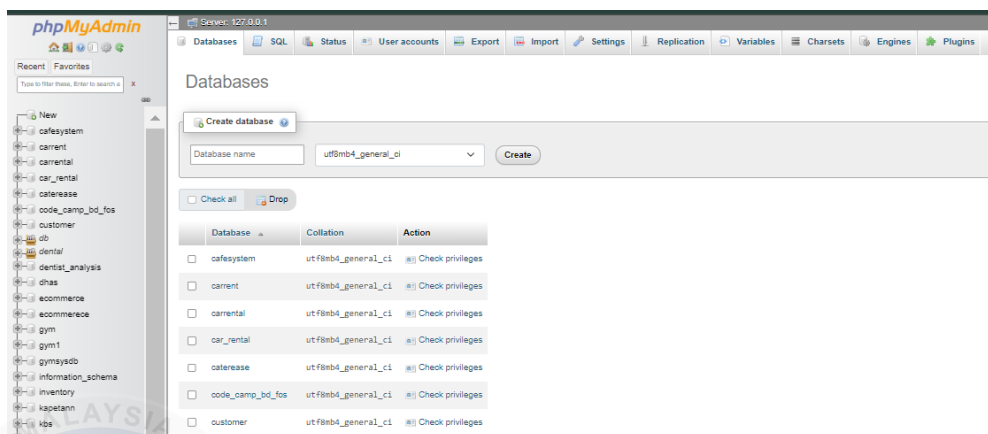


Figure 5.2.1.9 : Start the Server

5.3 Physical Design

In Sweet Scape database implementation, the database is used to test queries, including simple queries, stored functions, and triggers.

5.3.1 Data Definition Language (DDL)

Data Definition Language (DDL) encompasses SQL commands utilized to create and manage tables within a relational database. These statements facilitate the creation, alteration, and removal of database objects such as table, index, and trigger for Sweet Scape.

5.3.1.1 Create Table Commands

Figure 5.3.1.1 to Figure 5.3.1.7 show the query of each table commands.

```

CREATE TABLE `cust_orders` (
  `co_id` int(11) NOT NULL,
  `c_id` int(11) NOT NULL,
  `name` varchar(222) NOT NULL,
  `quantity` int(11) NOT NULL,
  `price` decimal(10,2) NOT NULL,
  `size` varchar(30) NOT NULL,
  `status` varchar(222) DEFAULT NULL,
  `date` datetime NOT NULL DEFAULT current_timestamp(),
  `order_status` enum('pending','completed') DEFAULT 'pending',
  `order_prepare` varchar(222) NOT NULL,
  `dish_id` int(11) DEFAULT NULL,
  `delivery_option` varchar(10) NOT NULL,
  `order_time` datetime NOT NULL DEFAULT current_timestamp(),
  `grandtotal` decimal(10,2) DEFAULT NULL,
  `orderID` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

Figure 5.3.1.1 : Create Table Cust_orders

```

CREATE TABLE `admin` (
  `adm_id` int(11) NOT NULL,
  `username` varchar(222) NOT NULL,
  `password` varchar(222) NOT NULL,
  `email` varchar(222) NOT NULL,
  `date` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

Figure 5.3.1.2: Create Table Admin

```
CREATE TABLE `category` (
  `category_id` int(11) NOT NULL,
  `c_name` varchar(222) NOT NULL,
  `date` date NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 5.3.1.3: Create Table Category

```
CREATE TABLE `customer` (
  `c_id` int(11) NOT NULL,
  `username` varchar(222) NOT NULL,
  `name` varchar(222) NOT NULL,
  `email` varchar(222) NOT NULL,
  `phone` varchar(222) NOT NULL,
  `password` varchar(222) NOT NULL,
  `address` text NOT NULL,
  `status` int(11) NOT NULL DEFAULT 1,
  `date` date NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 5.3.1.4: Create Table Customer

```
CREATE TABLE `payment` (
  `payment_id` int(11) NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `payment_date` date NOT NULL,
  `payment_method` varchar(50) NOT NULL,
  `card_number` varchar(20) DEFAULT NULL,
  `expiry_date` date DEFAULT NULL,
  `cvv` varchar(5) DEFAULT NULL,
  `co_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 5.3.1.5: Create Table Payment


```

CREATE TABLE `product` (
  `p_id` int(11) NOT NULL,
  `v_id` int(11) NOT NULL,
  `name` varchar(222) NOT NULL,
  `description` text NOT NULL,
  `producedate` date NOT NULL,
  `expirydate` date NOT NULL,
  `img` varchar(222) NOT NULL,
  `flavour` text NOT NULL,
  `date` date NOT NULL DEFAULT current_timestamp(),
  `category_id` int(11) DEFAULT NULL,
  `status` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

Figure 5.3.1.6: Create Table Product

```

CREATE TABLE `product_category` (
  `v_id` int(11) DEFAULT NULL,
  `category_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

Figure 5.3.1.7: Create Table Product_Category

5.3.1.2 Create Constraints

Figure 5.3.1.8 until Figure 5.3.1.11 show constraints for this system.

```
ALTER TABLE `cust_orders`
  ADD CONSTRAINT `c_id` FOREIGN KEY (`c_id`) REFERENCES
`customer` (`c_id`),
  ADD CONSTRAINT `fk_c_id` FOREIGN KEY (`c_id`) REFERENCES
`customer` (`c_id`) ON DELETE CASCADE,
  ADD CONSTRAINT `fk_cid` FOREIGN KEY (`c_id`) REFERENCES
`customer` (`c_id`) ON DELETE CASCADE;
```

Figure 5.3.1.8 : Constraints for Table Cust_orders

```
ALTER TABLE `product`
  ADD CONSTRAINT `fk_category` FOREIGN KEY (`category_id`)
REFERENCES `category` (`category_id`),
  ADD CONSTRAINT `fk_vendor` FOREIGN KEY (`v_id`) REFERENCES
`vendor` (`v_id`) ON DELETE CASCADE;
```

Figure 5.3.1.9 : Constraints for Table Product

```
ALTER TABLE `product_sizes`
  ADD CONSTRAINT `fk_product` FOREIGN KEY (`product_id`)
REFERENCES `product` (`p_id`) ON DELETE CASCADE;
COMMIT;
```

Figure 5.3.1.10: Constraints for Table Product_Sizes

```
ALTER TABLE `product`
ADD PRIMARY KEY (`p_id`),
ADD KEY `fk_category` (`category_id`),
ADD KEY `fk_vendor` (`v_id`);
```

Figure 5.3.1.11: Index for Table Product

5.3.2 Data Manipulation Language

Figure 5.3.2.1 until Figure 5.3.2.3 shows some examples of data manipulation language for this system.

```
INSERT INTO `admin` (`adm_id`, `username`, `password`, `email`, `date`)
VALUES
(1, 'ccbd', '0d89ec971a7bcfe26d68c177a9d53334', 'admin@gmail.com',
'2023-02-22 07:18:13'),
(2, 'staff', '0d89ec971a7bcfe26d68c177a9d53334', 'staff@gmail.com', '2024-
06-09 07:29:30');
```

Figure 5.3.2.1: Insert Statement - Insert Data into Table Admin

```
DELETE FROM admin
WHERE adm_id = '1';
```

Figure 5.3.2.2: Delete Statement - Delete admin from Admin Table

```
UPDATE product SET  
name = 'Kek Batik' WHERE  
p_id = '17';
```

Figure 5.3.2.3: Update Statement - Update Product Information

5.3.3 Power BI Implementation

5.3.3.1 Connection to Database

To integrate the sweetscape MySQL database with Power BI, the following steps were performed :

- I. Power BI Desktop was opened to establish the connection
- II. In Power BI Desktop, the Get Data was used to connect to the MySQL database folder
- III. Put your server name and database name.
- IV. The Data Connectivity mode was set to either import or DirectQuery
- V. The relevant tables were selected and loaded into Power BI
- VI. The relationships between the tables were visualized using schema map in the Model view Power BI.

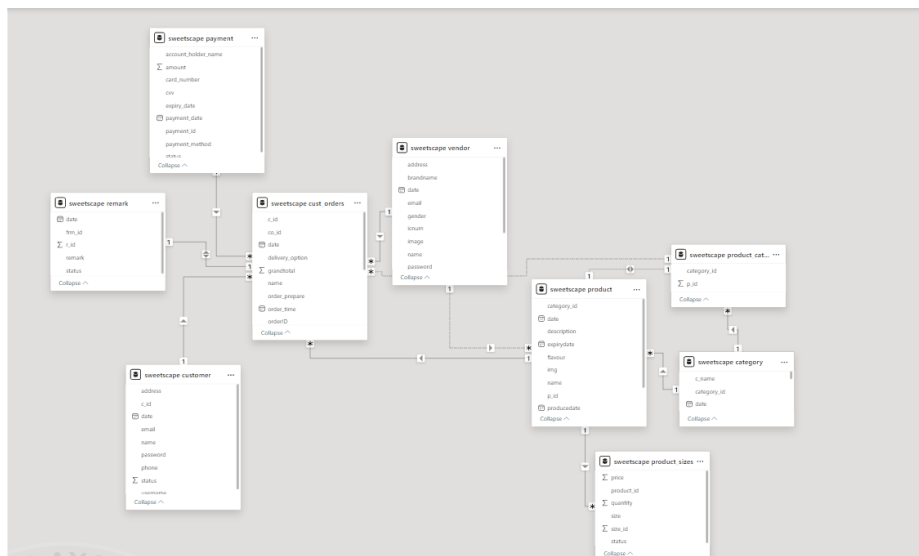


Figure 5.3.3.1: Schema Map Power BI

5.3.3.2 Synthetic Data Generation

To handle the large data requirements of this project, I used GPT-4 to generate synthetic data. Creating over 10,000 data entries manually was too challenging, so GPT-4 helped by producing a large volume of realistic and diverse data. This synthetic data closely mimics real-world scenarios, making the analysis and testing processes more effective. By using GPT-4, I was able to simulate various conditions and test the system's performance with a substantial dataset. This was crucial during the implementation phase for thorough testing of the Power BI dashboards and measures, ensuring they provide accurate and valuable insights.

5.3.3.3 Power BI Dashboard and Measures

The Power BI dashboard features four key sections: Sales, Vendor, Product, and Customer. Each dashboard is tailored to address specific business needs by offering detailed insights. The Sales dashboard monitors overall sales performance with essential metrics and visualizations. The Vendor dashboard evaluates vendor performance and relationships. The Product dashboard tracks product metrics, including sales and inventory levels. The Customer dashboard provides insights into customer behavior and preferences. Measures used include basic calculations like total sales, and more complex metrics such as the Sales Rate, which helps analyze the efficiency of sales performance over time.

I. Dashboard in Power BI

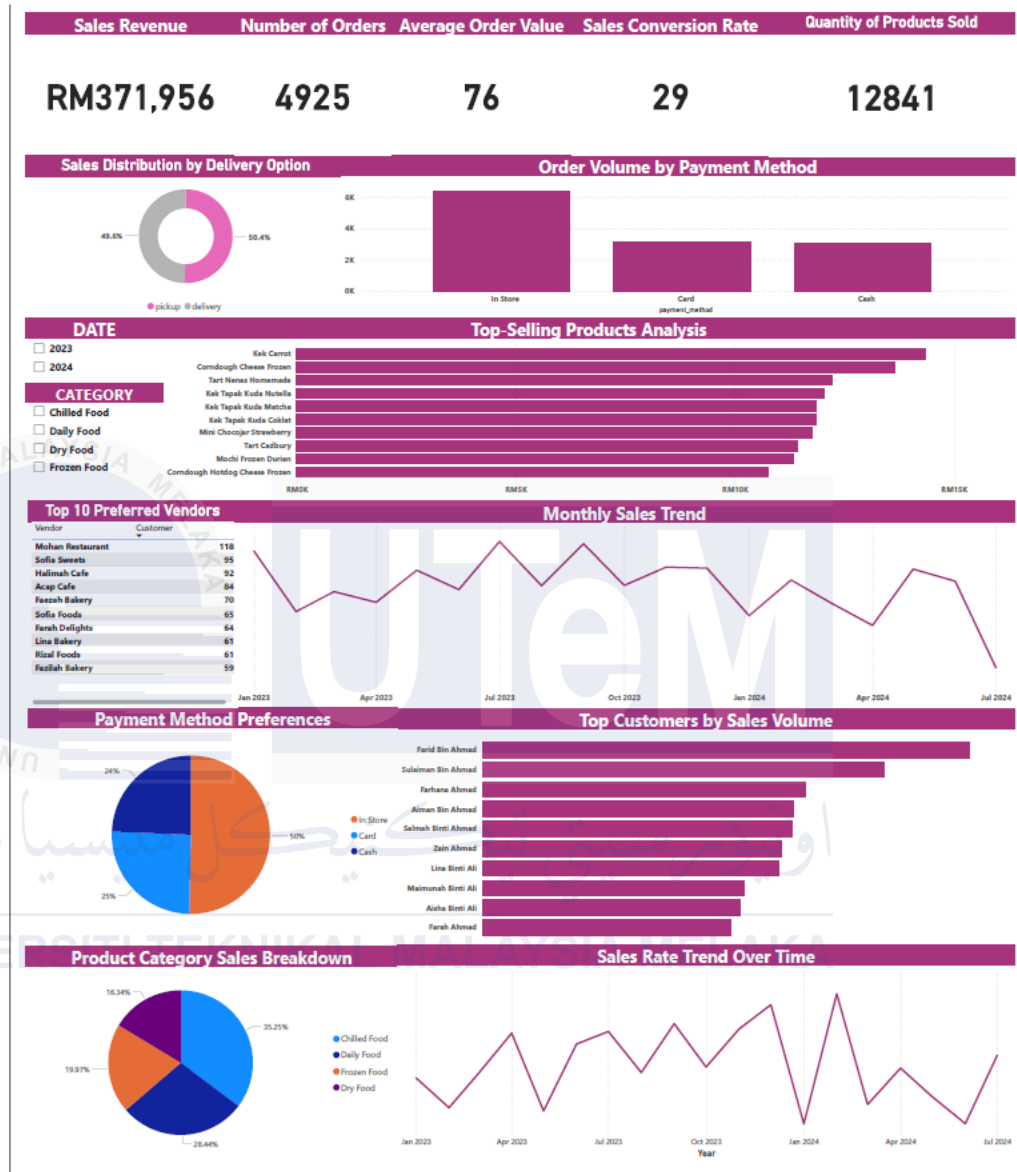


Figure 5.3.3.1: Dashboard Sales Power BI

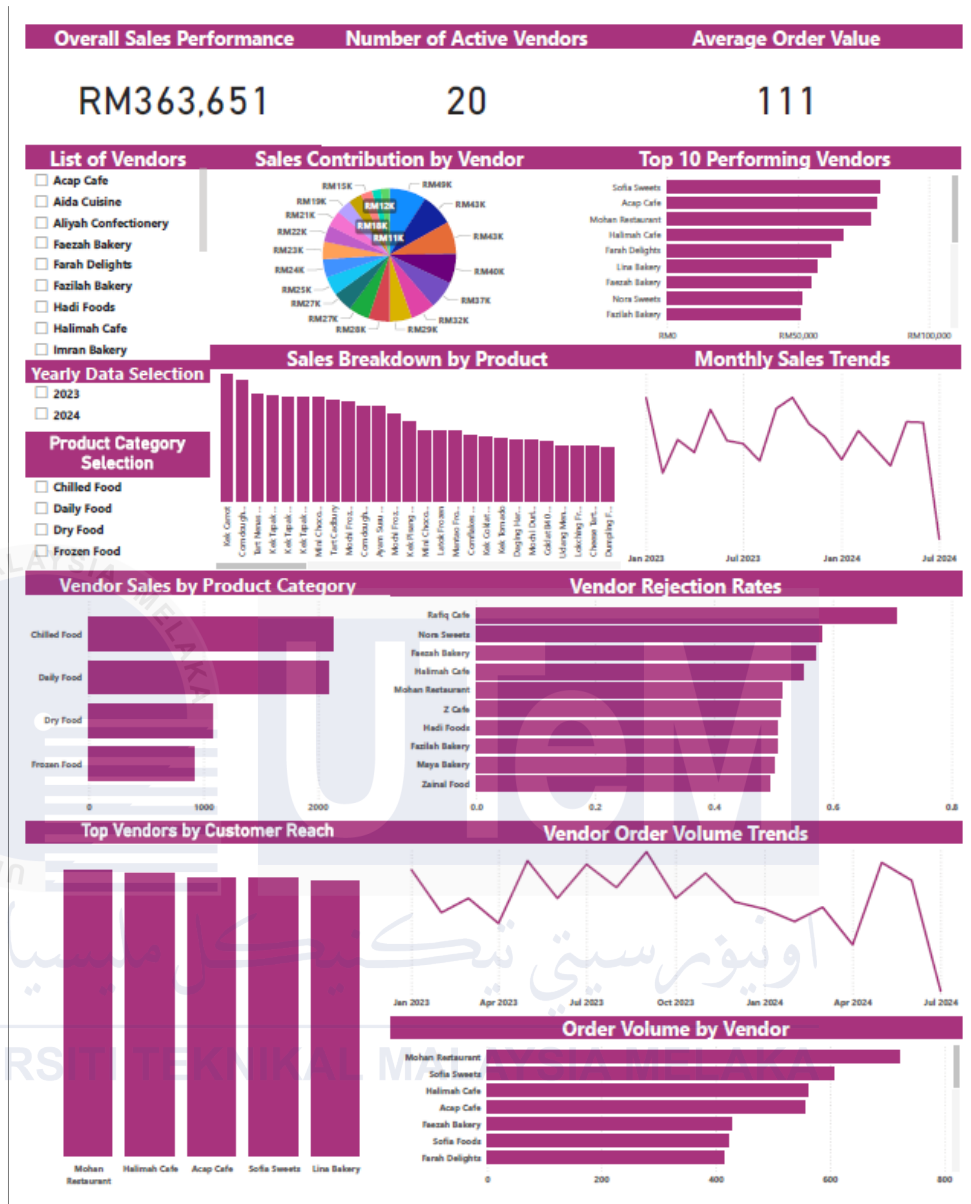


Figure 5.3.3.2: Dashboard Vendor Power BI

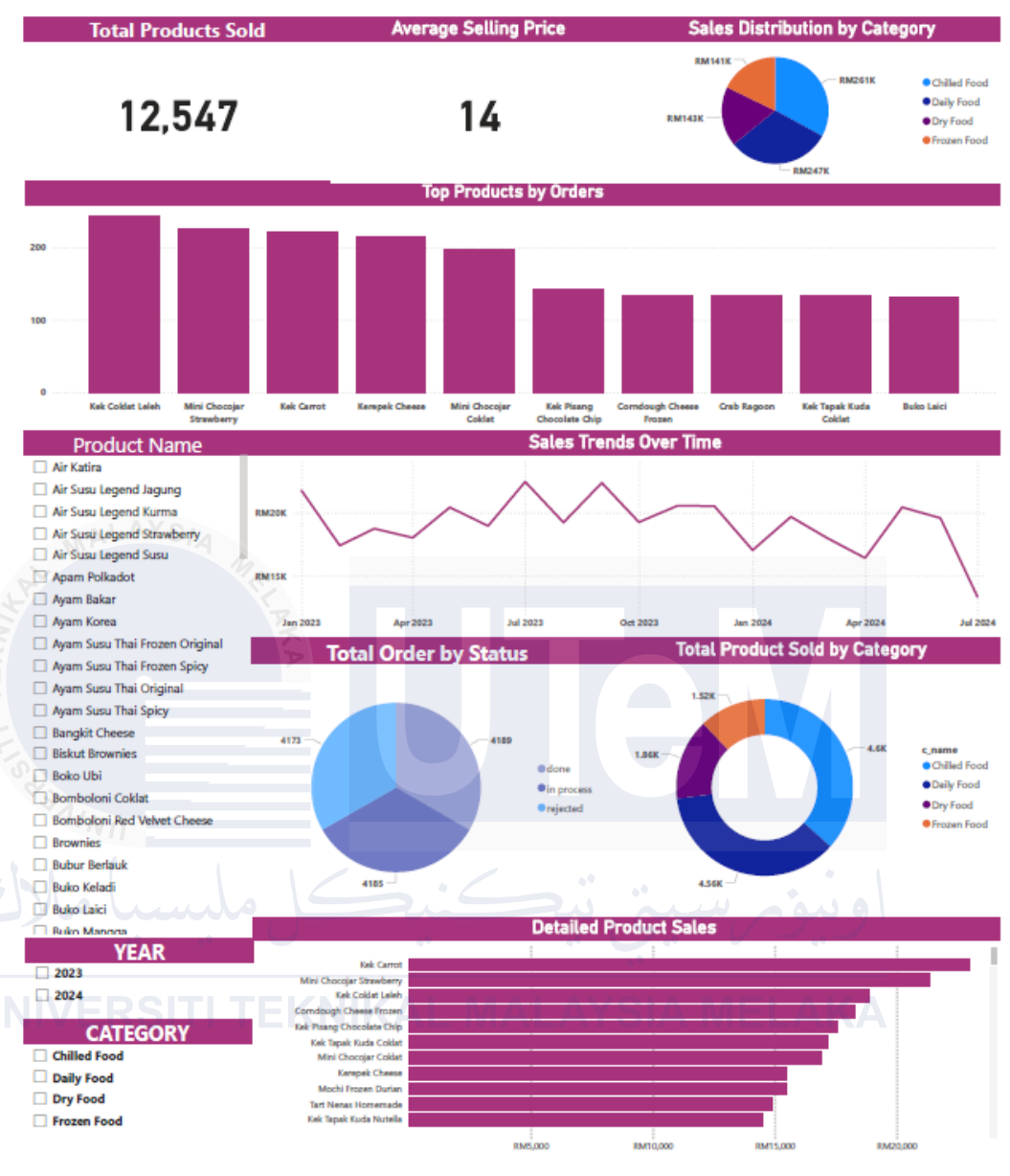


Figure 5.3.3.3: Dashboard Product Power BI

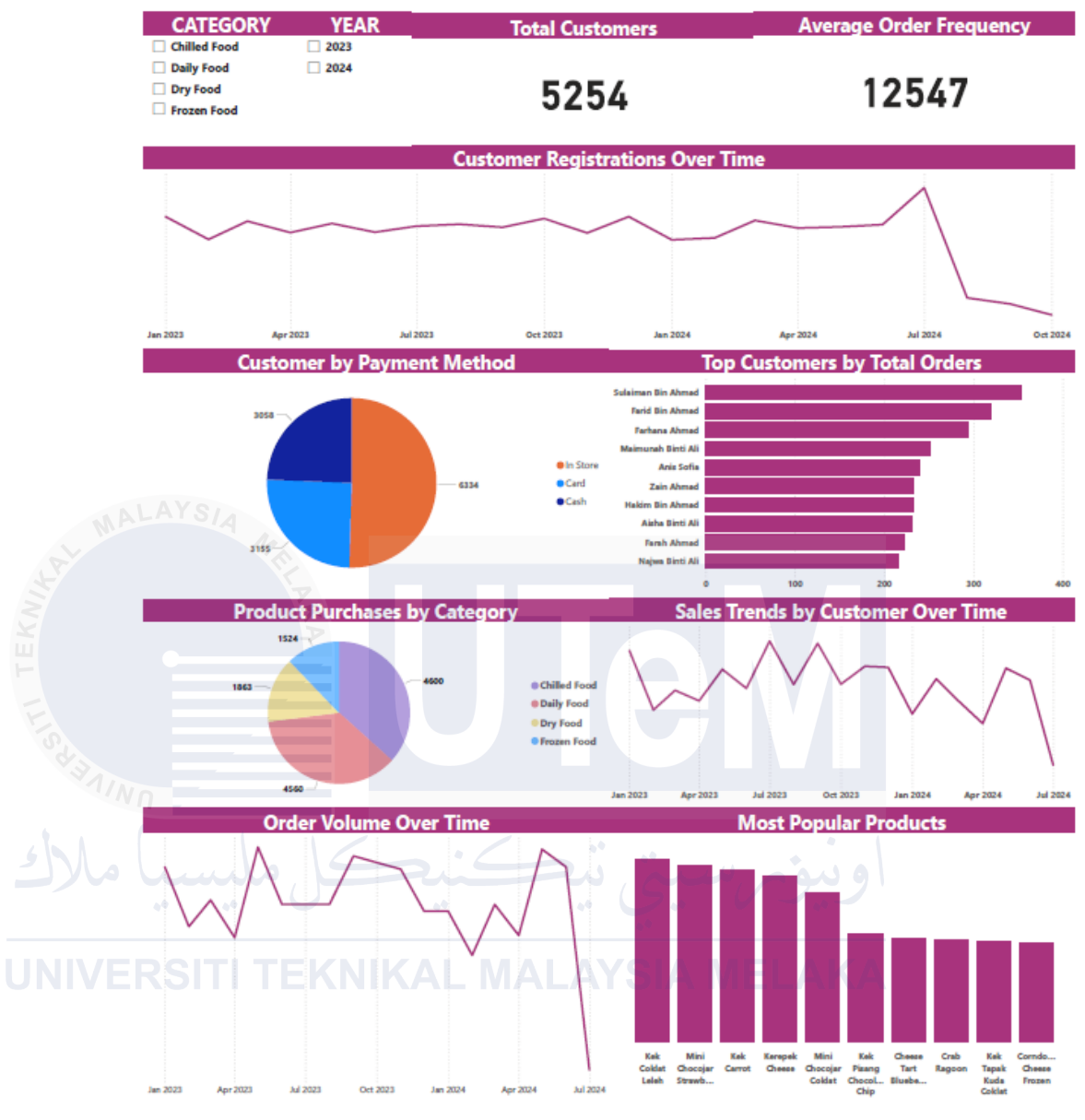


Figure 5.3.3.4: Dashboard Customer Power BI

II. Measures

1. Simple measures

AverageSellingPrice =

AVERAGEX('sweetscape cust_orders', 'sweetscape cust_orders'[price])

2. Complex measures

SalesRate=

DIVIDE([DistinctGrandTotalPerPaymentID_ExcludingRejected],

SUMX(SUMMARIZE(FILTER('sweetscape cust_orders',

'sweetscape cust_orders'[status] <> "rejected"

),

'sweetscape cust_orders'[orderID],

"TotalQuantity", MAX('sweetscape cust_orders'[quantity])), [TotalQuantity]))

5.4 Conclusions

Essentially, this chapter describes how to set up the software development environment, including the installation of XAMPP, PHP, and MySQL on a Windows platform. Additionally, database implementation was carried out to control the system operations. The system's foundation includes business logic with instructions for creating database tables using Data Definition Language (DDL) and managing data with Data Manipulation Language (DML). Furthermore, the chapter covers the setup and integration of Power BI to visualize and analyze the data. It details the process of connecting Power BI to the MySQL database, creating interactive dashboards, and implementing measures to derive meaningful insights from the data.

CHAPTER 6: TESTING

6.1 Introduction

Testing involves evaluating a software application's functionality to ensure it meets specified requirements and identifying any defects, aiming to ensure the system is error-free. This chapter concentrates on verifying the system through testing using the selected testing method. The primary objectives for testing the system are:

- To demonstrate that system satisfies its user requirements.
- To uncover any bugs or faults within the system using various testing strategies.

6.2 Test Plan

A test plan is a detailed technical document that outlines the testing strategy, scope, and the required resources, such as personnel, software, and hardware. It also includes the testing schedule and deliverables. The document provides an in-depth understanding of the system's workflow and functionality, describing how each component will be tested to ensure the system functions as intended and to identify any potential limitations.

6.2.1 Test Organization

In the Sweet Scape Vendor Dessert Management System , the test organization includes three user roles: Customer, Vendor, and Staff. Each role will be involved in testing both functional and non-functional requirements based on their responsibilities. Table 6.2.1.1 outlines the testing responsibilities assigned to each user role.

Table 6.2.1.1 Testing Roles and Responsibilities

Test ID	User	Responsibilities
T1	Vendor	<ul style="list-style-type: none"> • Testing the system, follow the test script. • Testing the product module.
T2	Staff	<ul style="list-style-type: none"> • Testing the system, follow the test script. • Testing the customer module. • Testing the product module. • Testing the vendor module. • Testing the order module. • Testing the report module. • Testing the backup & recovery
T3	Customer	<ul style="list-style-type: none"> • Testing the system, follow the test script. • Testing order module.

6.2.2 Test Environment

The test environment includes all the software, hardware, operating systems, tools, and network settings needed for testing. Tables 6.2.2.1 and 6.2.2.2 list the hardware and software required for the system test environment.

Table 6.2.2.1 Environment Hardware List

Environment specification	Description
Laptop	Lenovo IdeaPad 5 14ALC05
Processor	AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
Random Access Memory (RAM)	8.00 GB

Table 6.2.2.2 Environment Software List

Environment	Description
Database	MYSQL Manages data in database tables on a server.
Web Server	Xampp Provides a local web server for deploying and testing applications.
Operating System	Windows 11 Manages computer hardware and software resources and serves as the platform for running applications.
Web Browser	Google Chrome Executes PHP source code and tests the system's interface functionality.
Visual Studio Code	Utilized for writing PHP code.

Microsoft Word / Canva	Used for preparing final reports and creating presentation slides.
------------------------	--

6.3 Test Strategy

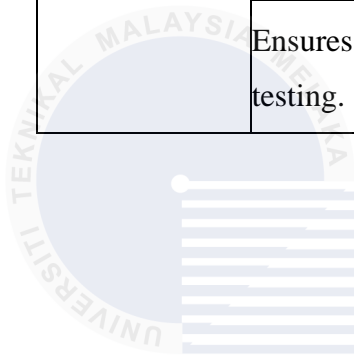
The test strategy outlines how testing will be carried out, including the methods, techniques, and specific modules to be tested. It provides a plan for ensuring that the software meets its requirements and functions correctly.

White box testing is a technique that involves examining the internal structure and logic of software. It includes several levels of testing to ensure the software operates reliably and meets all necessary criteria. Unit testing focuses on verifying that individual components function correctly. Integration testing assesses how well different components work together. System testing evaluates the entire software system to ensure it meets specified requirements. Finally, user acceptance testing (UAT) involves end users to confirm that the software meets their needs and is ready for deployment.

Table 6.3.1 Type of Test Design

White Box	
Type of Testing	<p><u>Unit testing</u> Test individual components or unites the software to ensure each one functions correctly. This ensures that each part of the code is functioning correctly on its own.</p> <p><u>Integration testing</u> Verifies how different components interact and function together.</p> <p><u>System Testing</u> Assesses the complete end-to-end scenarios of the software.</p> <p><u>User Acceptance Testing (UAT)</u> Performed by end users to ensure the system meets their needs and is ready for deployment.</p>

Test Design	<u>Code Coverage</u>
Techniques	Measures the extent to which the code has been executed during testing, ensuring that all parts of the code are tested
	<u>Path Testing</u>
	Involves testing all possible paths through the code to ensure that each path works correctly.
	<u>Branch Testing</u>
	Focuses on testing all the branches in the code to verify that each decision point functions as expected.
	<u>Statement Testing</u>
	Ensures that each line of code is executed at least once during testing.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

6.3.1 Classes of Tests

Below have several types of test class description that are being implemented on Sweet Scape Vendor Dessert Management System:

I. Input Validation Testing

Input validation testing ensures that the system correctly checks and handles user input. This type of testing verifies that all required fields are filled out and that the input meets the expected format and constraints. For example, it checks whether the username and name fields are not empty and whether the email address follows the correct format. If any input is missing or incorrect, the system should display appropriate error messages and prevent the registration process from proceeding. This testing helps ensure that users provide valid data and prevents errors or security issues from invalid input.

II. Control Flow Testing

Control flow testing ensures that the system processes data and follows its internal logic correctly. This type of testing focuses on verifying that the system correctly handles different paths in the code based on the input provided. For instance, it tests whether the system proceeds with registration when all inputs are valid and whether it displays error messages and halts the registration if any inputs are invalid. Control flow testing helps ensure that the program performs the correct actions and handles different scenarios as expected, including error handling and successful outcomes.

6.4 Test Design

Test design involves creating test cases based on the internal logic and code of the software. This process includes developing test cases that focus on different parts of the code, using test data to check various inputs and scenarios, and analyzing test results to see if the code behaves as expected. The goal is to ensure that all parts of the code are tested and that the software works correctly according to its design.

6.4.1 Test Description

The test case outlines the identification details, type of testing, testing strategy, and test class, along with preconditions and test requirements. It includes a step-by-step procedure for each test case and specifies the expected output results. Each module's test case is designed and documented to ensure comprehensive validation of the functionality, focusing on validating required fields, input formats, error handling, and successful output result.

**Table 6.4.1.1: Test Description of Registration
(Customer)**

Test ID	T001 – Register – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation		
Test Class	User Registration		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC1_1	Validate that all required fields are	<ul style="list-style-type: none"> Begin with the <code>if(isset(\$_POST['submit']))</code> check to ensure the form is 	- The system should not proceed

	<p>filled before allowing registration.</p>	<p>submitted.</p> <ul style="list-style-type: none"> • Ensure that form fields (username, email, password) are captured correctly via \$_POST. • Check that each required field (username, name, email) is not empty. • Validate the format of the email using a regular expression. • Ensure the address meets security requirements • If any field is empty or invalid, ensure an appropriate error message is displayed and the registration does not proceed. • Query the database to ensure the username and email are not already registered. • Validate that a new user record is created if all validations pass. 	<p>with registration if any required fields are empty, displaying appropriate error messages.- If all fields are valid and unique, the user should be successfully registered, and a confirmation message or email should be sent.</p>
--	---	--	--

Table 6.4.1.2 Test Description of Login (Customer)

Test ID	T002 – Login – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing and Session Management		
Test Class	User Authentication		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC2_1	Validate that the system properly handles user login, including session creation.	<ol style="list-style-type: none"> 1. Ensure the form submission is detected by checking <code>isset(\$_POST['submit'])</code>. 2. Capture form fields (username, password) via <code>\$_POST</code>. 3. Verify that the username and password fields are not empty. 4. Ensure input data is sanitized to prevent SQL injection. 5. Query the database to verify the existence of the username and ensure the password matches the stored hash. 6. Use functions like <code>password_verify()</code> to compare the entered 	<p>- The system should allow login with valid credentials, create a session, and redirect the user to the dashboard.</p> <p>- If invalid credentials are provided, the system should display an appropriate error message and</p>

		<p>password with the stored hashed password.</p> <p>7. If authentication is successful, initiate a session using <code>session_start()</code>.</p> <p>8. Set session variables like <code>\$_SESSION['user_id']</code> to track the logged-in user.</p> <p>9. Redirect the user to the dashboard or homepage after successful login.</p> <p>10. Display an appropriate error message if the username or password is incorrect.</p>	<p>prevent login.</p>
--	--	--	-----------------------

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.3: Test Description of Reset Password
(Customer)**

Test ID	T003 – Reset Password – Customer		
Testing Type	White box testing		
Test Strategy	Input Validation and Email Functionality		
Test Class	User Authentication and Security		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC3_1	Validate that the system handles the password reset process correctly, including email validation and sending the reset link.	<ol style="list-style-type: none"> 1. Ensure the form submission is detected by checking <code>isset(\$_POST['submit'])</code>. 2. Capture the email field via <code>\$_POST</code>. 3. Verify that the email field is not empty. 4. Validate the email format using a regular expression. 5. Query the database to check if the provided email exists in the system. 6. If the email exists, generate a unique token for the password reset link. 7. Store the token in the database associated with the user's email. 8. Send an email containing 	<p>- The system should send a password reset link to the provided email if it exists in the database.</p> <p>- If the email is invalid or not registered, the system should display an appropriate error message.</p>

		<p>the password reset link with the token to the user's email address.</p> <p>9. Ensure the system provides appropriate feedback, such as a success message indicating that a reset link has been sent or an error message if the email is not registered.</p>	
--	--	--	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.4: Test Description of Order Product
(Customer)**

Test ID	T004 – Order Product – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction		
Test Class	Order Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC4_1	Validate that the system correctly handles the ordering process, including item selection, price calculation, and cart management.	<ol style="list-style-type: none"> 1. Check that the user is logged in by verifying the existence of \$_SESSION['c_id']. 2. Retrieve category_id, product_id from the URL and validate its existence in the category table. 3. Prepare and execute a SQL query to fetch products based on category_id, product_id ensuring they are not expired and have a valid production date. 	The system should correctly add items to the cart, calculate the total, and display an accurate order summary.

**Table 6.4.1.5: Test Description of Choose Delivery
Method (Customer)**

Test ID	T005 – Delivery Method – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction		
Test Class	Order Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC5_1	Validate that the system allows customers to choose between pickup and delivery options.	<ol style="list-style-type: none"> 1. Ensure the form submission is detected and the delivery option is captured via <code>\$_POST['delivery_option']</code>. 2. Calculate the total price by iterating over the cart items from <code>\$_POST['cart_items']</code>. 3. Insert payment details into the payment table and retrieve the generated <code>payment_id</code>. 4. For each item in the cart, insert the order details into the <code>cust_orders</code> table, ensuring the correct <code>v_id</code> is fetched based on <code>p_id</code>. 5. Update the <code>product_sizes</code> table to reduce the quantity based on the ordered items. 	Store the delivery option and cart items in the session and redirect to the <code>checkout.php</code> page.

**Table 6.4.1.6: Test Description of Choose Payment
Method (Customer)**

Test ID	T006 – Choose Payment Method – Customer		
TestingType	White box testing		
Test Strategy	Control Flow Testing, Database Interaction		
Test Class	Payment Processing		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC6_1	Validate that the system correctly processes a cash payment and updates the order status.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['c_id'] and \$_SESSION['orderID'] are set; redirect to login.php or your_orders.php if not. 2. Retrieve orderID from the session and fetch all items associated with this order from cust_orders. 3. Calculate the total price, including the delivery charge, and update the payment table with the grand total using payment_id. 4. Display the items, their prices, quantities, and the grand total to the user. 5. Upon user confirmation (POST['confirm_order']), redirect to your_orders.php. 	The system should correctly calculate the grand total, update the payment amount in the database, and redirect the user after confirming the cash payment.
TC6_2	Validate that the system correctly processes a card payment, including	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['c_id'] and \$_SESSION['orderID'] are set; redirect to login.php or your_orders.php if not. 2. Retrieve orderID from the 	The system should validate card details, update the payment amount,

	<p>validation of card details and updating the payment status.</p>	<p>session and fetch all items associated with this order from cust_orders.</p> <ol style="list-style-type: none"> 3. Calculate the total price, including the delivery charge, and prepare to update the payment table with the grand total. 4. Validate card details on form submission (POST['cardSubmit']), including card number, expiry date, and CVV. 5. If the card is valid and not expired, update the payment details in the payment table and redirect to your_orders.php. 6. If the card details are invalid or expired, display an error message and prevent the update. 	<p>and redirect the user after successful payment processing.</p>
--	--	--	---

**Table 6.4.1.7: Test Description of Track Order Status
(Customer)**

Test ID	T007 – Track Order Status – Customer		
TestingType	White box testing		
Test Strategy	Control Flow Testing, Database Interaction		
Test Class	Order Management		
Test Case ID	Test requirements	Test / step procedure	Expectedoutput
TC7_1	Validate that the system correctly handles the tracking of customer orders and supports order cancellation.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['c_id'] is set; redirect to login.php if not. 2. If an order cancellation request (\$_GET['order_del']) is made, fetch payment_id from cust_orders using orderID. 3. If the payment_id is found, delete the order from cust_orders and the corresponding payment record from payment. 4. Handle date filtering with filter_date by setting it to the selected date or the current date by default. 5. Fetch consolidated 	The system should correctly fetch and display orders based on the selected date, allow order cancellation when appropriate, and provide a receipt link for completed orders.

		<p>orders for the customer from cust_orders, grouped by orderID and filtered by filter_date.</p> <p>6. Display the order details, including item names, quantities, sizes, total price, delivery option, payment method, status, and date.</p> <p>7. For each order, display a cancel button if the status is not closed, and a receipt link if the status is closed.</p>	
--	--	---	--

Table 6.4.1.8: Test Description of View and Download Receipt (Customer)

Test ID	T008 – View and Download Receipt – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, File Handling, Database Interaction		
Test Class	Order Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC8_1	Validate that the system correctly retrieves and displays the receipt details for viewing.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['c_id'] is set; redirect to login.php if not. 2. Capture orderID from the URL (\$_GET['orderID']) 3. Query the cust_orders and payment tables to fetch order details, payment method, and amount for the given orderID and c_id. 4. Fetch customer details from the customer table based on c_id. 5. Display the customer details, 	The system should accurately retrieve and display the order details and customer information, and allow the user to download the receipt as a PDF.

		<p>including name, phone number, address, payment method, and delivery option.</p> <p>6. Display the order details in a table format, including item name, quantity, size, and price.</p> <p>7. Calculate and display the total quantity and total price in the order summary.</p>	
TC8_2	<p>Validate that the system correctly generates a PDF receipt with all relevant order details.</p>	<ol style="list-style-type: none"> 1. Ensure <code>\$_SESSION['c_id']</code> is set; redirect to <code>login.php</code> if not. 2. Capture <code>orderID</code> from the URL (<code>\$_GET['orderID']</code>) 3. Query the <code>cust_orders</code> and <code>payment</code> tables to fetch order details, payment method, and amount for the given <code>orderID</code> and <code>c_id</code>. 4. Fetch customer details from the customer table 	<p>The system should generate a correct PDF receipt for download.</p>

		<p>based on c_id.</p> <ol style="list-style-type: none">5. Use the FPDF library to create a new PDF document.6. Add customer details to the PDF, including name, phone number, and address.7. Add order details to the PDF in a tabular format, including item name, quantity, size, price, delivery option, and payment method.8. Calculate and add the total quantity and total price to the PDF.9. Output the PDF for download as receipt.pdf.	
--	--	---	--

**Table 6.4.1.9: Test Description of Update Profile
(Customer)**

Test ID	T009 – Update Profile – Customer		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction		
Test Class	User Profile Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC9_1	Validate that the system correctly updates the user's profile details, ensuring all fields are properly validated and stored.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['c_id'] is set; redirect to login.php if not. 2. Retrieve the user's existing profile details from the customer table using c_id. 3. If the form is submitted, capture and sanitize the input fields (username, name, email, phone, address, and password) using mysqli_real_escape_string. 4. Validate the address to ensure it contains "Ayer Keroh"; if not, display an error message. 5. If the password field is not empty, hash the new password using md5. 6. Update the user's details in 	<p>The system should correctly validate and update the user's profile information, including handling address validation and optional password updates. It should provide appropriate feedback messages for success</p>

		<p>the customer table.</p> <p>7. If the update is successful, display a success message and refresh the user details.</p> <p>8. If an error occurs during the update, display an error message.</p>	<p>or failure.</p>
--	--	---	--------------------



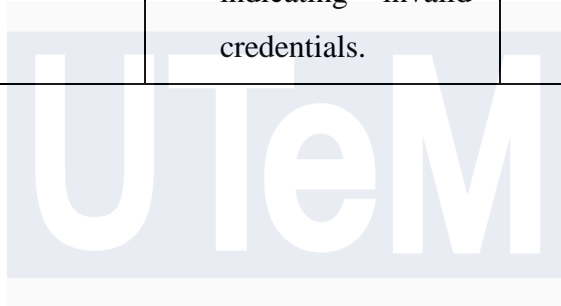
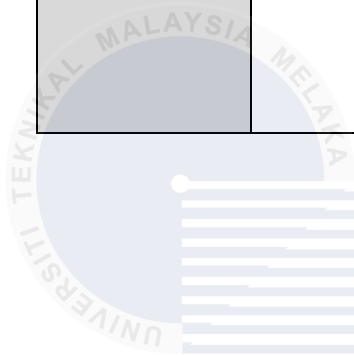
اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 6.4.1.10: Test Description of Login (Vendor)

Test ID	T010 – Login – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Session Management		
Test Class	Vendor Authentication		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC10_1	Validate that the system correctly authenticates the vendor's login credentials and manages session data.	<ol style="list-style-type: none"> 1. Ensure the login form is submitted (POST['submit']). 2. Capture the input fields username and password from the form submission. 3. Construct a SQL query to check the vendor table for a matching username and password (hashed using md5). 4. Execute the query and check if a matching row is found. 5. If a match is found, set the session variable 	<p>The system should authenticate valid credentials, initiate a session, and redirect the vendor to the dashboard.</p> <p>Invalid credentials should trigger an error message without initiating a session.</p>

		<p><code>\$_SESSION['v_id']</code> to the vendor's <code>v_id</code>.</p> <p>6. Redirect the user to <code>dashboard.php</code> and display a success message.</p> <p>7. If no match is found, display an error message indicating invalid credentials.</p>	
--	--	---	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 6.4.1.11: Test Description of View Sales on Dashboard (Vendor)

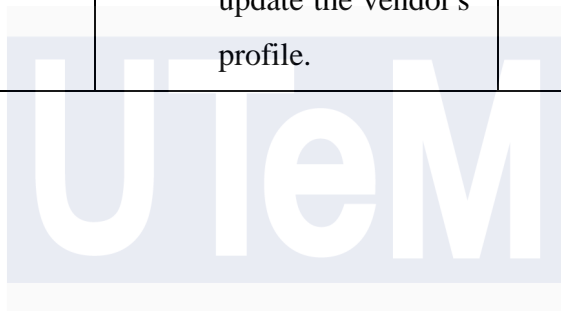
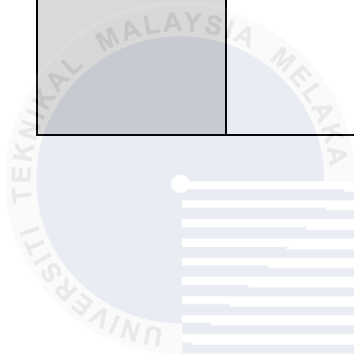
Test ID	T011 – View Sales on Dashboard – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Data Validation, SQL Query Execution		
Test Class	Data Retrieval and Display		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC11_1	Validate that the system correctly retrieves and displays all necessary data on the vendor's dashboard, including total products, inventory, sales, top-selling products, and alerts.	<ol style="list-style-type: none"> 1. Ensure the user is logged in (\$_SESSION['v_id']), redirect to login.php if not. 2. Retrieve and validate the vendor ID from the session (\$vendor_id = \$_SESSION['v_id']). 3. Execute SQL queries to fetch: <ul style="list-style-type: none"> • Total products • Total inventory • Total sales • Daily, weekly, and monthly sales data • Top-selling products • Low stock and expired products 	The system should accurately retrieve and display all required data on the vendor's dashboard, including proper alerts for low stock and expired products. Charts should be correctly

		<ul style="list-style-type: none"> • Sales by product category <ol style="list-style-type: none"> 4. Verify that the queries are executed without errors and return valid data. 5. Merge low stock and expired products into the \$alerts array. 6. Ensure that all retrieved data is correctly displayed in the corresponding sections of the dashboard. 7. Validate that the charts for sales data and top-selling products are generated correctly using Chart.js. 	<p>rendered with the fetched data.</p>
--	--	--	--

**Table 6.4.1.12: Test Description of Update Profile
(Vendor)**

Test ID	T012 – Update Profile – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Data Validation, SQL Query Execution		
Test Class	Form Submission and Data Update		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC12_1	<p>Validate that the system correctly updates the vendor's profile, including handling form submission, validating inputs, processing file uploads, and updating the database.</p>	<p>1. Ensure the vendor is logged in by checking the \$_SESSION['v_id'] session variable. If the vendor is not logged in, redirect to the login page.</p> <p>2. Upon form submission, validate the input fields:</p> <ul style="list-style-type: none"> • All fields except the password should be filled in. • Validate the email format. • Check for a file upload, ensuring the file type and size are correct. 	<p>The system should successfully update the vendor's profile with validated inputs. If any errors occur, appropriate error messages should be displayed.</p>

		<ul style="list-style-type: none">• If a password is provided, ensure it is at least 6 characters long. <ol style="list-style-type: none">3. Process the file upload if no errors are found.4. Prepare and execute the SQL update query to update the vendor's profile.	
--	--	--	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.13: Test Description of Add Product
(Vendor)**

Test ID	T013 – Add Product – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, File Handling		
Test Class	Product Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC13_1	Validate that the system correctly handles the addition of a new product, ensuring all inputs are properly validated, and the product details are stored in the database.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION['v_id'] is set; redirect to login.php if not. 2. Retrieve the vendor's brandname from the vendor table using v_id. 3. When the form is submitted, check if all required fields (p_name, about, producedate, expirydate, flavour, category, sizes, prices, quantities) are filled. 4. Validate the uploaded image file: 	<p>The system should correctly validate all inputs, handle image uploads, and store the product details in the database.</p> <p>The user should receive appropriate feedback for success or failure.</p>

		<ol style="list-style-type: none"> 1. Ensure the file is an image (jpg, jpeg, png, gif). 2. Ensure the file size is within the allowed limit (max 1024kb). 5. If validation passes: <ul style="list-style-type: none"> • Upload the image and store it in the products directory. • Insert the product details into the product table. • Insert the product sizes, prices, and quantities into the product_sizes table. • Calculate and store the status of each size based on quantity and expirydate. 6. If the image upload and database operations are successful, display a success message. 7. If any errors occur during validation or database 	
--	--	--	--

		operations, display an appropriate error message.	
--	--	---	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.14: Test Description of Update Product
(Vendor)**

Test ID	T014 – Update Product – Vendor		
TestingType	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, File Handling		
Test Class	Product Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC14_1	Validate that the system correctly handles the updating of an existing product, ensuring all inputs are properly sanitized, validated, and updated in the database.	<ol style="list-style-type: none"> 1. Ensure <code>\$_SESSION['v_id']</code> is set; redirect to <code>login.php</code> if not. 2. Retrieve the vendor's brandname from the vendor table using <code>v_id</code>. 3. On form submission, sanitize and validate all inputs (<code>p_name</code>, <code>about</code>, <code>producedate</code>, <code>expirydate</code>, <code>flavour</code>, <code>category</code>, <code>sizes</code>, <code>prices</code>, <code>quantities</code>). 4. Validate and handle image upload: <ul style="list-style-type: none"> • Ensure the file is an allowed image type (<code>jpg</code>, <code>jpeg</code>, <code>png</code>, <code>gif</code>). • Ensure the file size is within the allowed limit (max 1024kb). 	The system should correctly sanitize inputs, handle file uploads, update product details in the database, and provide appropriate feedback for success or failure.

		<ul style="list-style-type: none"> • Move the file to the products directory and update the image path in the database. <ol style="list-style-type: none"> 5. If no image is uploaded, update the product details without changing the image path. 6. Update or insert the product sizes, prices, and quantities in the product_sizes table, ensuring that sizes not included in the update are removed from the database. 7. Update the status of each size based on the quantity and expiry date. 8. If all operations succeed, display a success message; otherwise, display appropriate error messages. 	
--	--	--	--

**Table 6.3.1.15: Test Description of Delete Product
(Vendor)**

Test ID	T015 – Delete Product – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction		
Test Class	Product Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC15_1	Validate that the system correctly handles the deletion of a product, ensuring the input is properly sanitized, validated, and the database interaction is successful.	<ol style="list-style-type: none"> 1. Ensure that \$_SESSION['v_id'] is set. If not, the user should be redirected to index.php. 2. Check if the menu_del parameter is set in the GET request and is not empty. 3. Sanitize the menu_del parameter using mysqli_real_escape_string() to prevent SQL injection. 4. Execute the delete query DELETE FROM product WHERE p_id = '\$menu_del_id' to remove the product with the specified p_id from the database. 5. If the deletion is successful and rows are affected, redirect the user to 	The system should correctly handle the deletion process, including sanitization, execution, and proper feedback to the user, covering all possible paths and edge cases.

		<p>all_product.php with a delete_success=1 parameter.</p> <ol style="list-style-type: none">6. If no rows are affected, display a message indicating that no product was found with the specified ID.7. If the deletion fails, display an error message with the details of the failure.8. Test the scenario where menu_del is not set or is empty, ensuring the system displays an appropriate message.	
--	--	--	--

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.16: Test Description of Receive Notification
for Expired & Low Stock (Vendor)**

Test ID	T016 – Receive Notification for Low Stock – Vendor		
TestingType	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, and Session Management		
Test Class	Vendor Dashboard Notifications		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC16_1	Validate that the system correctly fetches notifications for low stock and expired products, properly handles session management, and displays notifications on the vendor dashboard.	<ol style="list-style-type: none"> 1. Verify that the user is logged in by checking if <code>\$_SESSION['v_id']</code> is set. If not, the user should be redirected to <code>login.php</code>. 2. Execute SQL queries to retrieve low stock products (quantity < 5) and expired products (expirydate < current date) for the logged-in vendor. 3. Combine these results into a single array alert for further processing. 4. Ensure the alerts are correctly displayed on the vendor 	The system should correctly handle session checks, fetch and combine alerts for low stock and expired products, and display the notifications on the vendor dashboard accurately, covering all paths and edge cases.

		<p>dashboard, including proper badge labels for low stock and expired products.</p> <p>5. Confirm the correct notification count is shown in the bell icon.</p>	
--	--	---	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.17: Test Description of Receive Request form
by Email (Vendor)**

Test ID	T017 – Receive Request from Admin – Vendor		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, and Email Functionality		
Test Class	Vendor Request Processing		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC17_1	Validate that the system correctly receives requests from the admin, processes the request data, fetches the correct product and vendor details, and sends an email to the vendor.	<ol style="list-style-type: none"> 1. Ensure that the connection to the database is established using valid credentials (\$servername, \$username, \$password, \$dbname). 2. Using the provided product_name, retrieve the associated p_id and v_id from the product table. 3. Fetch the vendor_name and brandname from the vendor table using the 	The system should successfully process the request, fetch the correct product and vendor details, validate the inputs, and send an email to the vendor.

		<p>retrieved v_id.</p> <ol style="list-style-type: none">4. Populate the form fields (product_name, vendor_name, brandname) with the fetched details.5. Allow the admin to fill in a description for the vendor before submitting the request.6. Verify that the correct product and vendor details are displayed.7. Test form submission to ensure the request is processed correctly and the email is sent to the vendor.	
--	--	--	--

Table 6.4.1.18: Test Description of Login (Admin)

Test ID	T018 – Login – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Session Management, and Security Testing		
Test Class	Admin Authentication		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC18_1	Validate that the system correctly handles admin login by verifying username and password, managing sessions securely, and providing appropriate feedback.	<ol style="list-style-type: none"> 1. Ensure that both username and password fields are provided. 2. Validate that a session is correctly initiated upon successful login. 3. Check if <code>\$_SESSION['adm_id']</code> is set correctly. 4. The session variable <code>\$_SESSION['adm_id']</code> should be set upon successful login, and the user should be redirected to <code>dashboard.php</code>. 	The system should correctly handle admin login, manage sessions securely, provide appropriate feedback for success or failure, and protect against security vulnerabilities such as SQL injection.

**Table 6.4.1.19: Test Description of View Dashboard Data
(Admin)**

Test ID	T019 – View Dashboard Data – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Session Management		
Test Class	Admin Panel Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC19_1	Ensure the admin dashboard loads correctly with all components, and the data displayed (new vendors, products, customers) is accurate based on the current date.	<ol style="list-style-type: none"> 1. Ensure \$_SESSION["adm_id"] is set; redirect to index.php if not. 2. Retrieve data for new vendors, products, customers, orders, product categories, new orders, delivered orders, canceled orders, and today's total earnings. 3. Validate SQL queries for retrieving today's records from vendor, product, customer, cust_orders, and category tables. 4. Display the correct 	The system should load and display all sections on the admin dashboard, showing accurate data based on today's records. If any data retrieval fails, an appropriate error message

		<p>number of records in the respective sections of the dashboard.</p> <ol style="list-style-type: none"> 5. Verify that the correct total earnings are displayed by summing up the price from cust_orders where status is 'closed'. 6. Check that the PowerBI link is accessible and correctly integrated. 7. If any error occurs during data retrieval, ensure an error message is displayed. 	<p>should be shown.</p>
--	--	---	-------------------------

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 6.4.1.20: Test Description of Add Vendor (Admin)

Test ID	T020 – Add Vendor – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, File Upload Handling		
Test Class	Vendor Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC20_1	Ensure that the system correctly adds a new vendor, including validating inputs, handling file uploads, and storing data in the database.	<ol style="list-style-type: none"> 1. Check that all required fields (res_name, icnum, phone, email, gender, vaccine, address, brandname, username, password) are filled. If any are empty, display an error message. 2. Ensure the phone number contains 10 to 11 digits. 3. Validate the IC number follows the format XXXXXX-XX-XXXX. 4. Verify that the 	The system should correctly validate all input fields, handle file uploads according to the specified criteria, insert the vendor's information into the database, and provide appropriate feedback to the user, whether the operation succeeds or fails.

		<p>email address is correctly formatted.</p> <ol style="list-style-type: none">5. Ensure that the uploaded file's extension is jpg, png, gif, or jpeg.6. Check that the file size does not exceed 1MB.7. If the file meets all criteria, store the image in the <code>../Res_img/</code> directory.8. Hash the password using md5.9. Insert the vendor's details into the vendor table.10. On successful insertion, move the uploaded file to the specified directory.11. If an error occurs during the insertion, display an error message detailing the	
--	--	---	--

		<p>problem.</p> <p>12. If the vendor is successfully added, display a success message confirming the addition.</p>	
--	--	---	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.21: Test Description of Update Vendor
(Admin)**

Test ID	T021 – Update Vendor – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, File Upload Handling		
Test Class	Vendor Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC21_1	Ensure that the system correctly updates the vendor's profile details, including handling optional password updates and image uploads.	<ol style="list-style-type: none"> 1. Ensure all required fields (res_name, icnum, phone, email, gender, vaccine, address, brandname, username) are filled; if any are empty, display an error message. 2. Validate the uploaded file's extension and size; store the file in ../Res_img/ if valid. 3. If the password field is filled, hash the new 	The system should correctly validate the input fields, handle optional password and image updates, update the vendor's information in the database, and provide appropriate feedback to the user, whether the operation succeeds or fails.

		<p>password using md5 and include it in the update.</p> <p>4. Construct and execute the SQL query to update the vendor's details in the vendor table.</p> <p>5. Display appropriate error or success messages based on the outcome.</p>	
TC21_2	<p>Validate that the changes are saved correctly in the database.</p>	<p>1. After updating the vendor information, query the database to retrieve the updated vendor details.</p> <p>2. Verify that the vendor details in the database match the changes made.</p>	<p>The database should accurately reflect the updated vendor details, confirming that the changes were saved correctly.</p>

**Table 6.4.1.22: Test Description of Delete Vendor
(Admin)**

Test ID	T022 – Delete Vendor – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Session Management.		
Test Class	Vendor Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC22_1	Validate that the admin can delete an existing vendor from the system.	<ol style="list-style-type: none"> 1. Check if the res_del parameter is set in the URL. If not, redirect to the vendor list page. 2. Sanitize the res_del parameter to prevent SQL injection attacks. 3. Execute the SQL query to delete the vendor record based on the sanitized vendor_id. 4. If the deletion is successful, redirect the user 	<p>-The system should display a confirmation message indicating that the vendor has been successfully deleted.</p> <p>-The deleted vendor should no longer appear in the vendor list.</p>

		<p>to the vendor list page with a success message.</p> <p>5. If the deletion fails, display an error message showing the reason for the failure.</p>	
--	--	--	--



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.23: Test Description of Update Order Status
(Admin)**

Test ID	T023 – Update Order Status – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Session Management		
Test Class	Order Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC23_1	Validate that the system correctly updates the order status, inserts remarks, and manages payment and order preparation status based on the current status.	<ol style="list-style-type: none"> 1. Ensure the OrderID is retrieved from the URL and sanitized using <code>mysqli_real_escape_string</code>. 2. When the form is submitted, capture and sanitize the input fields (status and remark) using <code>mysqli_real_escape_string</code>. 3. Verify that the remark is correctly inserted into the remark table with the associated OrderID and status. 4. Retrieve the <code>payment_id</code> from the <code>cust_orders</code> table and store it. 5. Update the order status in the <code>cust_orders</code> table: <ul style="list-style-type: none"> • If the status is "closed," update the status to 	The system should correctly update the order status, manage payment and order preparation status based on the input, and provide appropriate feedback messages for success or failure.

		<p>"closed" and order_prepare to "complete."</p> <ul style="list-style-type: none"> • For other statuses, update only the status. <p>6. Always update the payment status in the payment table based on the status:</p> <ul style="list-style-type: none"> • If status is "rejected," update the payment and order_prepare status to "rejected." • For other statuses, update payment status to "success." <p>7. If the update is successful, display a success message. If not, display an error message.</p>	
--	--	--	--

**Table 6.4.1.24: Test Description of View Sales Report
(Admin)**

Test ID	T024 – View Sales Reports – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Visualization Validation		
Test Class	Admin Dashboard Data Visualization.		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC24_1	Validate that the system correctly retrieves and displays sales data, including the total number of vendors, products, customers, and orders for each month of the selected year, ensuring accurate calculation and representation of earnings and order statuses.	<ol style="list-style-type: none"> 1. Start a session and include the connection file. 2. Check if the admin is logged in by verifying the adm_id session. If not, redirect to index.php. 3. Retrieve distinct years from cust_orders for year selection. 4. Set the selected year to the current year or the year provided by the user. 5. Retrieve the total number of vendors, 	The system should correctly retrieve and display sales-related data, including the total number of vendors, products, and customers.

		<p>products, and customers from their respective tables.</p> <p>6. Query and calculate the number of orders per month for the selected year.</p> <p>7. Retrieve the total earnings per month for successful payments in the selected year.</p> <p>8. Retrieve the distribution of order statuses for the selected year.</p> <p>9. Query the count of customers using each payment method in the selected year.</p> <p>10. Render the data using Chart.js for visual representation on the dashboard.</p>	
--	--	--	--

		<p>11. Provide year selection and overview cards for total vendors, products, and customers.</p> <p>12. Generate bar charts for orders per month, line charts for earnings, and pie charts for order statuses and payment methods.</p> <p>13. Include error handling for database queries.</p> <p>14. Conclude by including Bootstrap and jQuery scripts.</p>	
--	--	---	--

**Table 6.4.1.25: Test Description of View Product Sales
(Admin)**

Test ID	T025 – View Product Sales – Admin		
TestingType	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Pagination Validation, Data Visualization		
Test Class	Admin Report Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC25_1	<p>Validate that the system retrieves and displays the correct product sales data based on payment methods for the selected month. Ensure correct pagination, sorting, and chart visualization.</p>	<ol style="list-style-type: none"> 1. Start a session and ensure the user is logged in as an admin. If not, redirect to the login page. 2. Retrieve and validate the selected month from \$_POST or \$_GET or default to the current month. 3. Connect to the database and check the connection status. 4. Calculate pagination variables and determine the starting index 	<p>The system should display product sales data accurately for the selected month, categorized by payment methods, with proper pagination and sorting. Charts should be correctly populated with the top 10 most purchased items,</p>

		<p>based on the current page.</p> <ol style="list-style-type: none"> 5. Retrieve the total number of distinct products sold within the selected month, ensuring the correct query is constructed. 6. Calculate the total number of pages required for the results. 7. Fetch the sorted product sales data based on the selected month, ensuring correct JOINS between cust_orders, product, and product_category tables. 8. Display the fetched data in a table format, ensuring all data points (product names, quantities) are correctly aligned. 9. Validate that the 	<p>grouped by food type, and should reflect the accurate quantities sold.</p>
--	--	---	---

		<p>pagination controls work correctly, ensuring navigation between pages does not lose the selected month's context.</p> <p>10. Fetch and display the top 10 most purchased items, categorized by food type (e.g., daily, chilled, frozen, dry) using Google Charts.</p> <p>11. Ensure the charts display accurate data corresponding to the selected month and are rendered without errors.</p>	
--	--	--	--

Table 6.4.1.26: Test Description of View Revenue Reports by Payment Method (Admin)

Test ID	T026 – View Revenue Reports by Payment Method – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, Database Interaction, Data Visualization		
Test Class	Revenue Report Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC26_1	Validate that the system accurately retrieves and displays the total revenue and revenue by payment method (Card, Cash, In Store) for the selected month, using the correct date range.	<ol style="list-style-type: none"> 1. Ensure <code>\$_SESSION['adm_id']</code> is set; redirect to <code>index.php</code> if not. 2. Retrieve the selected month from <code>\$_POST['selected_month']</code>, defaulting to the current month. 3. Set the date range for the selected month using the first and last day. 4. Execute the SQL query to calculate the total revenue for the selected month. 5. Execute the SQL queries to calculate the revenue by payment method (Card, Cash, In Store) for the selected month. 	The system should correctly calculate and display the total revenue and revenue per payment method for the selected month.

		<ol style="list-style-type: none">6. Store the resulting revenue data in variables for display in the HTML.7. Display the revenue data in a table format, showing the total revenue and revenue per payment method.8. Display bar and donut charts to visualize the revenue by payment method using Google Charts.9. Handle any errors during SQL execution by displaying an error message.	
--	--	--	--

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**Table 6.4.1.27: Test Description of Backup and Recover
Database (Admin)**

Test ID	T027 – Backup and Recover Database – Admin		
Testing Type	White box testing		
Test Strategy	Control Flow Testing, Input Validation, File Handling, Database Interaction		
Test Class	Database Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC27_1	Validate that the system correctly backs up the database to an SQL file and recovers the database from the latest backup file.	<ol style="list-style-type: none"> 1. Establish a database connection using the provided host, username, password, and database name. If the connection fails, terminate the script with an error message. 2. Retrieve all table names from the database using the SHOW TABLES command. Handle any errors in retrieving the tables. 3. Generate an SQL script for creating table structures and inserting data for each table. Ensure proper formatting and data escape handling. 4. Write the generated SQL script to a backup file in the specified directory. 	The system should successfully create a backup of the database, save it to a specified directory, and recover the database from the latest backup file with appropriate feedback for success or failure.

		<p>Ensure the file is created with a proper timestamped name.</p> <p>5. Verify that the backup file is successfully saved and accessible in the specified directory.</p> <p>6. Identify the latest SQL backup file by sorting files based on modification time in the backup directory.</p> <p>7. Restore the database by reading and executing SQL commands from the latest backup file. Handle SQL execution errors and log any issues.</p> <p>8. Provide user feedback: show a success message if the backup and restore operations are completed successfully or display an error message if any step fails.</p> <p>9. Redirect the user to the dashboard after a successful operation or handle errors by showing an appropriate alert and redirecting to the backup page.</p>	
--	--	---	--

**Table 6.4.1.28: Test Description of Power BI Dashboard
Integration (Admin)**

Test ID	T028 – Power BI Dashboard Integration – Admin		
TestingType	White box testing		
Test Strategy	Control Flow Testing, UI Interaction, External Link Validation		
Test Class	Admin Dashboard Management		
Test Case ID	Test requirements	Test / step procedure	Expected output
TC28_1	<p>Validate that the system correctly integrates the Power BI dashboard link into the admin panel and ensures the link redirects to the appropriate Power BI report.</p>	<ol style="list-style-type: none"> 1. Ensure the admin is logged in by verifying the session variable <code>\$_SESSION["admin_id"]</code>. Redirect to the login page if not set. 2. Verify that the Power BI dashboard link is correctly rendered on the admin panel using the <code><a></code> tag. 3. Click on the Power BI dashboard link and ensure it redirects to the expected URL: <code>https://app.powerbi.com/groups/me/reports/a52160cf-b73d-4e74-be60-78ed15ed28a0/6c2caea66dfe38578170?</code> 	<p>The Power BI dashboard link should be correctly rendered on the admin panel and should redirect to the specified Power BI report URL when clicked. The report should load without errors, and unauthorized users should not have access to the link.</p>

		<p>experience=power-bi.</p> <p>4. Confirm that the external Power BI dashboard loads correctly without any errors and verify that the report displayed is consistent with the data being managed in the admin panel.</p> <p>5. Check that the Power BI dashboard link is only accessible to logged-in admin users and is not visible to unauthorized users or users without admin privileges.</p>	
--	--	---	--

6.4.2 Test Data

Table 6.4.2.1: Test Data of Register (Customer)

Test Scenario	Input Data	Expected Result
Missing Username	{'username': '', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	Error message indicating that the username field is required.
Missing Name	{'username': 'Roslan', 'name': '', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	Error message indicating that the name field is required.
Invalid Email Format	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	Error message indicating that the email format is invalid.
Valid Registration Data	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	User should be successfully registered, and a confirmation message should be sent

Table 6.4.2.2: Test Data of Login (Customer)

Test Scenario	Input Data	Expected Result
Missing Username	{'username': '', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username field is required.
Missing Password	{'username': 'Roslan', 'password': ''}	Error message indicating that the password field is required.
Invalid Username	{'username': 'ruslan', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username does not exist.
Valid Credentials	{'username': 'Roslan', 'password': 'cc079201c0405996aca524961b6e17de'}	User should be successfully logged in, and a session should be created.

Table 6.4.2.3: Test Data of Reset Password (Customer)

Test Scenario	Input Data	Expected Result
Missing Email	{'email': ""}	Error message indicating that the email field is required.
Invalid Email Format	{'email': 'Roslan.com'}	Error message indicating that the email format is invalid.
Email Not Registered	{'email': 'roslan@hotmail.com'}	Error message indicating that the email is not registered.
Valid Email	{'email': 'roslan@gmail.com'}	A reset link should be sent to the provided email.

Table 6.4.2.4: Test Data of Order Product (Customer)

Test Scenario	Input Data	Expected Result
Valid Product Selection	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '2', 'price': '14', 'size': 'medium', 'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option': 'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '28', 'grandtotal': '28', 'orderID': 'ORDER-663474', 'payment_id': '9', 'v_id': '21', 'p_id': '101'}	The system should correctly add items to the cart and display an accurate order summary.

Invalid Product ID	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '1', 'price': '14', 'size': 'medium', 'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option': 'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '14', 'grandtotal': '14', 'orderID': 'ORDER-663474', 'payment_id': '9', 'v_id': '21', 'p_id': '999'}	Error message indicating that the product does not exist.
Expired Product	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '1', 'price': '14', 'size': 'medium', 'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option': 'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '14', 'grandtotal': '14', 'orderID': 'ORDER-663474', 'payment_id': '9', 'v_id': '21', 'p_id': '102'}	Error message indicating that the product is expired.

Table 6.4.2.5: Test Data of Delivery Method (Customer)

Test Scenario	Input Data	Expected Result
Valid Delivery Option	{'delivery_option': 'pickup'}	The system should correctly store the delivery option and

		redirect to the checkout page.
Invalid Delivery Option	{'delivery_option': 'invalid_option'}	Error message indicating that the delivery option is invalid.

**Table 6.4.2.6: Test Data of Choose Payment Method
(Customer)**

Test Scenario	Input Data	Expected Result
Valid Cash Payment	{'payment_method': 'cash'}	The system should correctly process the payment and update the order status.
Invalid Card Details	{'payment_method': 'card', 'card_number': '1234 5678 9012 3456', 'expiry_date': '01/20', 'cvv': '123'}	Error message indicating that the card details are invalid or expired.

**Table 6.4.2.7: Test Data of Track Order Status
(Customer)**

Test Scenario	Input Data	Expected Result
Valid Order ID	{'orderId': 'ORDER-478058'}	The system should correctly display the order details and allow cancellation if applicable.
Invalid Order ID	{'orderId': 'ORDER-333333'}	Error message indicating that the order ID does not exist.

**Table 6.4.2.8: Test Data of View and Download Receipt
(Customer)**

Test Scenario	Input Data	Expected Result
Valid Order ID	{'orderID': 'ORDER-478058'}	The system should correctly generate and display the receipt details.
Invalid Order ID	{'orderID': 'ORDER-333333'}	Error message indicating that the order ID does not exist.

Table 6.4.2.9: Test Data of Update Profile (Customer)

Test Scenario	Input Data	Expected Result
Valid Address	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	User should be successfully registered with the valid address.
Invalid Address	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': '123 Invalid Address'}	Error message indicating that the address is invalid.

Table 6.4.2.10: Test Data of Login (Vendor)

Test Scenario	Input Data	Expected Result
Missing Username	{'username': '', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username field is required.

Missing Password	{'username': 'faezah', 'password': ''}	Error message indicating that the password field is required.
Invalid Username	{'username': 'fazah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username does not exist.
Valid Credentials	{'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Vendor should be successfully logged in, session should be created, and the user should be redirected to the dashboard.

**Table 6.4.2.11: Test Data of View Sales on Dashboard
(Vendor)**

Test Scenario	Input Data	Expected Result
Valid Vendor ID	{'v_id': '10'}	The system should correctly retrieve and display all required data on the vendor's dashboard.
Invalid Vendor ID	{'v_id': '00'}	Error message indicating that the vendor ID does not exist.

Table 6.4.2.12: Test Data of Update Profile (Vendor)

Test Scenario	Input Data	Expected Result
Valid Profile Update	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly update the vendor's profile and display a success message.
Invalid Email Format	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': 'fiyahx12@gmail', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the email format is invalid.
Missing Required Fields	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': '', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that required fields are missing.

Table 6.4.2.13: Test Data of Add Product (Vendor)

Test Scenario	Input Data	Expected Result
Valid Product Data	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024', 'expirydate': '30/8/2024', 'img': '6680c26209b64.jpeg', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]}'}	The system should correctly validate all inputs, handle image uploads, and store the product details in the database.
Invalid Image File Type	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024',	Error message indicating that the image file type is invalid.

	<pre>'expirydate': '30/8/2024', 'img': 'invalid_file.txt', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]'</pre>	
--	---	--

Table 6.4.2.14: Test Data of Update Product (Vendor)

Test Scenario	Input Data	Expected Result
Valid Product Update	<pre>{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '31/8/2024', 'expirydate': '10/9/2024', 'img': '6680c26209b64.jpeg', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]'}</pre>	The system should correctly update the product details in the database and provide a success message.
Missing Required Fields	<pre>{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna</pre>	Error message indicating that required fields are missing.

	<p>Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '31/8/2024', 'expirydate': '10/9/2024', 'img': '6680c26209b64.jpeg', 'flavour': '', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]'</p>	
--	--	--

Table 6.4.2.15: Test Data of Delete Product (Vendor)

Test Scenario	Input Data	Expected Result
Valid Product ID	{'p_id': '1'}	The system should correctly delete the product and provide a success message.
Invalid Product ID	{'p_id': '0'}	Error message indicating that the product ID does not exist.

Table 6.4.2.16: Test Data of Receive Notification for Low Stock & Expired Products (Vendor)

Test Scenario	Input Data	Expected Result
Valid Vendor ID	{'v_id': '1'}	The system should correctly fetch and display notifications for low stock and expired products.
Invalid Vendor ID	{'v_id': '0'}	Error message indicating that the vendor ID does not exist.

Table 6.4.2.17: Test Data of Receive Request from Admin (Vendor)

Test Scenario	Input Data	Expected Result
Valid Product and Vendor Details	{'name': 'Kek Pandan Cheese', 'vendor_name': 'Nur Faezah', 'brandname': 'Faezah Bakery'}	The system should correctly process the request, fetch the correct product and vendor details, and send an email to the vendor.
Invalid Product Name	{'name': 'Kek Pandan', 'vendor_name': 'Nur Faezah', 'brandname': 'Faezah Bakery'}	Error message indicating that the product does not exist.

Table 6.4.2.18: Test Data of Login (Admin)

Test Scenario	Input Data	Expected Result
Missing Username	{'username': '', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username field is required.
Missing Password	{'username': 'ccbd', 'password': ''}	Error message indicating that the password field is required.
Invalid Username	{'username': 'invalidadmin', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username does not exist.
Valid Credentials	{'username': 'ccbd', 'password': 'cc079201c0405996aca524961b6e17de'}	Admin should be successfully logged in, session should be created, and the user should be redirected to the dashboard.

**Table 6.4.2.19: Test Data of View Dashboard Data
(Admin)**

Test Scenario	Input Data	Expected Result
Valid Admin ID	{'adm_id': '1'}	The system should correctly retrieve and display all required data on the admin's dashboard.
Invalid Admin ID	{'adm_id': '0'}	Error message indicating that the admin ID does not exist.

Table 6.4.2.20: Test Data of Add Vendor (Admin)

Test Scenario	Input Data	Expected Result
Valid Vendor Data	{'v_id': 12, 'name': 'Nur Fazilah', 'icnum': '761218-01-1837', 'phone': '0128472293', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery', 'image': '667993075cef7.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly validate all inputs, handle file uploads, and store the vendor details in the database.
Invalid IC Format	{'v_id': 12, 'name': 'Nur Fazilah', 'icnum': '1234', 'phone': '0128472293', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery', 'image':	Error message indicating that the IC number is in

	'667993075cef7.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah', 'password': 'cc079201c0405996aca524961b6e17de'}	
--	--	--

Table 6.4.2.21: Test Data of Update Vendor (Admin)

Test Scenario	Input Data	Expected Result
Valid Vendor Update	{'v_id': 12, 'name': 'Nur Fazilah Updated', 'icnum': '761218011837', 'phone': '0128472293', 'email': 'fiyahx12_updated@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery Updated', 'image': '667993075cef7_updated.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah_updated', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly update the vendor details in the database and provide a success message.
Missing Required Fields	{'v_id': 12, 'name': '', 'icnum': '', 'phone': '0128472293', 'email': 'vendorC@example.com', 'gender': '', 'vaccine': 'Yes', 'address': '789 Road', 'brandname': '', 'image': 'vendorC.jpg', 'date': '24/6/2024 23:38', 'username': '', 'password': 'VendorPassword123'}	Error message indicating that required fields are missing.

Table 6.4.2.22: Test Data of Delete Vendor (Admin)

Test Scenario	Input Data	Expected Result
Valid Vendor ID	{'v_id': '1'}	The system should correctly delete the vendor and provide a success message.
Invalid Vendor ID	{'v_id': '0'}	Error message indicating that the vendor ID does not exist.

Table 6.4.2.23: Test Data of Update Order Status (Admin)

Test Scenario	Input Data	Expected Result
Valid Order Update	{'order_id': 'ORDER-366913', 'status': 'closed', 'remark': 'Order completed successfully.'}	The system should correctly update the order status and manage payment status accordingly.
Invalid Order ID	{'order_id': '9999', 'status': 'closed', 'remark': 'Order completed successfully.'}	Error message indicating that the order ID does not exist.

Table 6.4.2.24: Test Data of View Sales Reports (Admin)

Test Scenario	Input Data	Expected Result
Valid Year and Month	{'year': '2023', 'month': 'January'}	The system should correctly retrieve and display sales-related data, including total vendors, products, customers, and

		orders for the selected month.
Invalid Year	{'year': '9999', 'month': 'January'}	Error message indicating that the year does not have any data.
No Data for Selected Year	{'year': '2022'}	The system should indicate that no data is available for the selected year and handle the scenario gracefully.

Table 6.4.2.25: Test Data of View Product Sales (Admin)

Test Scenario	Input Data	Expected Result
Valid Month	{'month': 'January'}	The system should correctly retrieve and display product sales data for the selected month, with accurate pagination and sorting.
Invalid Month	{'month': 'InvalidMonth'}	Error message indicating that the month is invalid.
No Sales Data for Month	{'month': '2024-09'}	The system should indicate that no sales data is available for september 2024 and handle the scenario gracefully.

Table 6.4.2.26: Test Data of View Revenue Reports by Payment Method (Admin)

Test Scenario	Input Data	Expected Result
Valid Month and Year	{'month': 'January', 'year': '2023'}	The system should correctly calculate and display the total revenue and revenue per payment method for the selected month.
Invalid Month	{'month': 'InvalidMonth', 'year': '2023'}	Error message indicating that the month is invalid.

Table 6.4.2.27: Test Data of Backup and Recover Database (Admin)

Test Scenario	Input Data	Expected Result
Valid Backup and Recovery	{}	The system should successfully create a backup of the database and recover it from the latest backup file.
Backup Failure	{invalid database connection details}	The system should terminate with an error message if the database connection fails.
Recovery Failure	{no backup file available}	The system should display an error message if no backup file is available for recovery.

**Table 6.4.2.28: Test Data of Power BI Dashboard
Integration (Admin)**

Test Scenario	Input Data	Expected Result
Valid Power BI Integration	{}	The Power BI dashboard should be correctly embedded and visible within the admin interface without errors.
Data Accuracy Verification	{}	The Power BI dashboard should display accurate real-time data that matches the live database.
Unauthorized Access	{non-admin user}	The Power BI dashboard link should not be visible or accessible to unauthorized users or users without admin privileges.

6.5 Test Result and Analysis

Table 6.5.1: Test Result of Register (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC1 _1	Missing Username	{'username': '', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	Error message indicating that the username field is required.	Error message displayed correctly, registration did not proceed.	Pass
	Missing Name	{'username': 'Roslan', 'name': '', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	Error message indicating that the name field is required.	Error message displayed correctly, registration did not proceed.	Pass
	Invalid Email Format	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer	Error message indicating that the email format is invalid.	Error message displayed correctly, registration did not proceed.	Pass

		Keroh Jaya, Melaka'}			
Valid Registrati on Data	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	User should be successfully registered, and a confirmation message should be sent.	User registered successfully, confirmation message sent.	Pass	

Table 6.5.2: Test Result of Login (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC2 _1	Missing Username	{'username': '', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username field is required.	Error message displayed correctly, login did not proceed.	Pass
	Missing Password	{'username': 'Roslan', 'password': ''}	Error message indicating that the password field is required.	Error message displayed correctly, login did not proceed.	Pass
	Invalid Username	{'username': 'ruslan', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username does not exist.	Error message displayed correctly, login did not proceed.	Pass

	Valid Credentials	{'username': 'Roslan', 'password': 'cc079201c0405996aca524961b6e17de'}	User should be successfully logged in, and a session should be created.	User successfully logged in, session created, and user redirected to the dashboard.	Pass
--	-------------------	--	---	---	------

Table 6.5.3: Test Result of Reset Password (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC3_1	Missing Email	{'email': ''}	Error message indicating that the email field is required.	Error message displayed correctly, password reset did not proceed.	Pass
	Invalid Email Format	{'email': 'Roslan.com'}	Error message indicating that the email format is	Error message displayed correctly, password reset did	Pass

			invalid.	not proceed.	
Email Not Registered	{'email': 'roslan@hotmail.com'}	Error message indicating that the email is not registered.	Error message displayed correctly, password reset did not proceed.	Pass	
Valid Email	{'email': 'roslan@gmail.com'}	A reset link should be sent to the provided email.	Reset link successfully sent to the provided email, and a success message was displayed.	Pass	

Table 6.5.4: Test Result of Order Product (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC4_1	Valid Product Selection	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '2', 'price': '14', 'size': 'medium',	The system should correctly add items to the cart and display an accurate order	Items added to the cart successfully, total calculated accurately, and order	Pass

		'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option': 'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '28', 'grandtotal': '28', 'orderID': 'ORDER-663474', 'payment_id': '9', 'v_id': '21', 'p_id': '101'}	summary.	summary displayed correctly.	
	Invalid Product ID	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '1', 'price': '14', 'size': 'medium', 'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option':	Error message indicating that the product does not exist.	Error message displayed correctly, product was not added to the cart.	Pass

		'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '14', 'grandtotal': '14', 'orderID': 'ORDER- 663474', 'payment_id': '9', 'v_id': '21', 'p_id': '999'}			
	Expired Product	{'co_id': '24', 'c_id': '112', 'name': 'Coklat B40 Coklat', 'quantity': '1', 'price': '14', 'size': 'medium', 'status': 'in process', 'date': '24/1/2023 11:14', 'order_prepare': 'in process', 'delivery_option': 'pickup', 'order_time': '24/1/2023 11:14', 'totalamount': '14', 'grandtotal': '14', 'orderID':	Error message indicating that the product is expired.	Error message displayed correctly, product was not added to the cart.	Pass

		'ORDER-663474', 'payment_id': '9', 'v_id': '21', 'p_id': '102'}			
--	--	--	--	--	--

Table 6.5.5: Test Result of Delivery Method (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC5_1	Valid Delivery Option	{'delivery_option': 'pickup'}	The system should correctly store the delivery option and redirect to the checkout page.	Delivery option stored successfully, redirected to the checkout page.	Pass
	Invalid Delivery Option	{'delivery_option': 'invalid_option'}	Error message indicating that the delivery option is invalid.	Error message displayed correctly, and the delivery option was not stored.	Pass

**Table 6.5.6: Test Result of Choose Payment Method
(Customer)**

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC6_1	Valid Cash Payment	{'payment_method': 'cash'}	The system should correctly process the payment and update the order status.	Payment processed successfully, order status updated, user redirected to your_orders.php.	Pass
TC6_2	Invalid Card Details	{'payment_method': 'card', 'card_number': '1234 5678 9012 3456', 'expiry_date': '01/20', 'cvv': '123'}	Error message indicating that the card details are invalid or expired.	Error message displayed correctly, payment was not processed, and order status was not updated.	Pass

Table 6.5.7: Test Result of Track Order Status (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC7_1	Valid Order ID	{'orderID': 'ORDER-478058'}	The system should correctly display the order details and allow cancellation if applicable.	Order details displayed correctly, and cancellation option available if applicable.	Pass

TC7_2	Invalid Order ID	{'orderId': 'ORDER-333333'}	Error message indicating that the order ID does not exist.	Error message displayed correctly, no order details shown, and no cancellation option available.	Pass
-------	------------------	-----------------------------	--	--	------

Table 6.5.8: Test Result of View and Download Receipt (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC8_1	Valid Order ID (View)	{'orderId': 'ORDER-478058'}	The system should correctly generate and display the receipt details.	Receipt details displayed correctly with all relevant information.	Pass
	Invalid Order ID (View)	{'orderId': 'ORDER-333333'}	Error message indicating that the order ID does not exist.	Error message displayed correctly, no receipt details shown.	Pass

TC8_2	Valid Order ID (PDF)	{'orderID': 'ORDER-478058'}	The system should correctly generate a PDF receipt with all relevant order details.	PDF receipt generated and downloaded successfully with all correct details.	Pass
	Invalid Order ID (PDF)	{'orderID': 'ORDER-333333'}	Error message indicating that the order ID does not exist, and no PDF should be generated.	Error message displayed correctly, no PDF generated.	Pass

Table 6.5.9: Test Result of Update Profile (Customer)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC9_1	Valid Address	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': 'No.33, Lorong Taman Ayer Keroh Jaya, Melaka'}	User's profile should be successfully updated with the valid address.	Profile updated successfully, valid address accepted, and success message displayed.	Pass

	Invalid Address	{'username': 'Roslan', 'name': 'Roslan Huzaimy', 'email': 'roslan@gmail.com', 'phone': '1234567859', 'password': 'cc079201c0405996aca524961b6e17de', 'address': '123 Invalid Address'}	Error message indicating that the address is invalid.	Error message displayed correctly, profile not updated due to invalid address.	Pass
--	-----------------	--	---	--	------

Table 6.5.10: Test Result of Login (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC10_1	Missing Username	{'username': '', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username field is required.	Error message displayed correctly, login did not proceed.	Pass
	Missing Password	{'username': 'faezah', 'password': ''}	Error message indicating that the password field is required.	Error message displayed correctly, login did not proceed.	Pass

Invalid Username	{'username': 'fazah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the username does not exist.	Error message displayed correctly, login did not proceed.	Pass
Valid Credentials	{'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Vendor should be successfully logged in, session should be created, and the user should be redirected to the dashboard.	Vendor logged in successfully, session created, and redirected to dashboard.php.	Pass

Table 6.5.11: Test Result of View Sales on Dashboard (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC11_1	Valid Vendor ID	{'v_id': '10'}	The system should correctly retrieve and display all required data on the vendor's dashboard.	Dashboard displayed all required data correctly, including total products, inventory, sales, and	Pass

				top-selling products.	
	Invalid Vendor ID	{'v_id': '00'}	Error message indicating that the vendor ID does not exist.	Error message displayed correctly, no data retrieved or displayed on the dashboard.	Pass

Table 6.5.12: Test Result of Update Profile (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC12_1	Valid Profile Update	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly update the vendor's profile and display a success message.	Profile updated successfully, success message displayed.	Pass

Invalid Email Format	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': 'fiyahx12@gmail', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the email format is invalid.	Error message displayed correctly, profile not updated.	Pass
Missing Required Fields	{'v_id': 10, 'name': 'Nur Faizah', 'icnum': '870612000000', 'phone': '01128364927', 'email': '', 'gender': 'Female', 'address': 'No.10 , Jalan Tasik Utama 36 , Taman Tasik Utama Ayer Keroh', 'vaccine': 'Yes', 'brandname': 'Faezah Bakery', 'image': '6680cf9bb3691.jpg', 'date': '19/6/2024 12:18', 'username': 'faezah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that required fields are missing.	Error message displayed correctly, profile not updated.	Pass

Table 6.5.13: Test Result of Add Product (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC13_1	Valid Product Data	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024', 'expirydate': '30/8/2024', 'img': '6680c26209b64.jpeg', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]'}	The system should correctly validate all inputs, handle image uploads, and store the product details in the database.	Product added successfully, inputs validated, image uploaded, and data stored in the database.	Pass
	Invalid Image File Type	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum,	Error message indicating that the image file type is invalid.	Error message displayed correctly, product not added due to invalid image file	Pass

		Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024', 'expirydate': '30/8/2024', 'img': 'invalid_file.txt', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]}'		type.	
--	--	--	--	-------	--

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 6.5.14: Test Result of Update Product (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC13_1	Valid Product Data	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024', 'expirydate': '30/8/2024', 'img': '6680c26209b64.jpeg', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]}'}	The system should correctly validate all inputs, handle image uploads, and store the product details in the database.	Product added successfully, inputs validated, image uploaded, and data stored in the database.	Pass
	Invalid Image File Type	{'p_id': 1, 'v_id': 10, 'name': 'Kek Pandan Cheese', 'description': 'Gula, Telur, Ovelatte, Minyak Masak, Pewarna	Error message indicating that the image file type	Error message displayed correctly, product not added due to	Pass

		Hijau, Perisa Pandan, Tepung Gandum, Baking Powder, Cream Cheese Tatura, Susu Cair, Susu Pekat Manis', 'producedate': '20/8/2024', 'expirydate': '30/8/2024', 'img': 'invalid_file.txt', 'flavour': 'Pandan', 'date': '30/6/2024', 'category_id': 2, 'img_blob': '[BLOB - 7.5 KiB]}'	is invalid.	invalid image file type.	
--	--	---	-------------	--------------------------------	--

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 6.5.15: Test Result of Delete Product (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC15_1	Valid Product ID	{'p_id': '1'}	The system should correctly delete the product and provide a success message.	Product deleted successfully, success message displayed, and user redirected to all_product.php with delete_success=1.	Pass
	Invalid Product ID	{'p_id': '0'}	Error message indicating that the product ID does not exist.	Error message displayed correctly, no product deleted, and appropriate feedback provided to the user.	Pass

Table 6.5.16: Test Result of Receive Notification for Low Stock & Expired Products (Vendor)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC16_1	Valid Vendor ID	{'v_id': '1'}	The system should correctly fetch and display notifications for low stock and expired products.	Notifications for low stock and expired products fetched and displayed correctly, including notification count in the bell icon.	Pass
	Invalid Vendor ID	{'v_id': '0'}	Error message indicating that the vendor ID does not exist.	Error message displayed correctly, no notifications fetched or displayed.	Pass

**Table 6.5.17: Test Result of Receive Request from Admin
(Vendor)**

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC17_1	Valid Product and Vendor Details	{'name': 'Kek Pandan Cheese', 'vendor_name': 'Nur Faezah', 'brandname': 'Faezah Bakery'}	The system should correctly process the request, fetch the correct product and vendor details, and send an email to the vendor.	Request processed successfully, correct details fetched, and email sent to the vendor.	Pass
	Invalid Product Name	{'name': 'Kek Pandan', 'vendor_name': 'Nur Faezah', 'brandname': 'Faezah Bakery'}	Error message indicating that the product does not exist.	Error message displayed correctly, request not processed due to invalid product name.	Pass

Table 6.5.18: Test Result of Login (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC18_1	Missing Username	{'username': '', 'password': 'cc079201c0405996 aca524961b6e17de'}	Error message indicating that the username field is required.	Error message displayed correctly, login did not proceed.	Pass
	Missing Password	{'username': 'ccbd', 'password': ''}	Error message indicating that the password field is required.	Error message displayed correctly, login did not proceed.	Pass
	Invalid Username	{'username': 'invalidadmin', 'password': 'cc079201c0405996 aca524961b6e17de'}	Error message indicating that the username does not exist.	Error message displayed correctly, login did not proceed.	Pass
	Valid Credentials	{'username': 'ccbd', 'password': 'cc079201c0405996 aca524961b6e17de'}	Admin should be successfully logged in, session should be created, and the user should be redirected to the dashboard.	Admin logged in successfully, session created, and user redirected to dashboard.php.	Pass

**Table 6.5.19: Test Result of View Dashboard Data
(Admin)**

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC19_1	Valid Admin ID	{'adm_id': '1'}	The system should correctly retrieve and display all required data on the admin's dashboard.	Dashboard displayed all required data correctly, including new vendors, products, customers, orders, and total earnings.	Pass
	Invalid Admin ID	{'adm_id': '0'}	Error message indicating that the admin ID does not exist.	Error message displayed correctly, no data retrieved or displayed on the dashboard.	Pass

Table 6.5.20: Test Result of Add Vendor (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC20_1	Valid Vendor Data	{'v_id': 12, 'name': 'Nur Fazilah', 'icnum': '761218-01-1837', 'phone': '0128472293', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery', 'image': '667993075cef7.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly validate all inputs, handle file uploads, and store the vendor details in the database.	Vendor added successfully, inputs validated, file uploaded, and data stored in the database.	Pass
	Invalid IC Format	{'v_id': 12, 'name': 'Nur Fazilah', 'icnum': '1234', 'phone': '0128472293', 'email': 'fiyahx12@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery', 'image': '667993075cef7.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah', 'password': 'cc079201c0405996aca524961b6e17de'}	Error message indicating that the IC number is in an incorrect format.	Error message displayed correctly, vendor not added due to invalid IC number format.	Pass

Table 6.5.21: Test Result of Update Vendor (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC21_1	Valid Vendor Update	{'v_id': 12, 'name': 'Nur Fazilah Updated', 'icnum': '761218011837', 'phone': '0128472293', 'email': 'fiyahx12_updated@gmail.com', 'gender': 'Female', 'vaccine': 'Yes', 'address': 'No.55 , Jalan Tasik Utama 28 , Taman Tasik Utama Ayer Keroh', 'brandname': 'Fazilah Bakery Updated', 'image': '667993075cef7_updated.jpg', 'date': '24/6/2024 23:38', 'username': 'fazilah_updated', 'password': 'cc079201c0405996aca524961b6e17de'}	The system should correctly update the vendor details in the database and provide a success message.	Vendor details updated successfully, inputs validated, optional password and image updated, and success message displayed.	Pass
	Missing Required Fields	{'v_id': 12, 'name': '', 'icnum': '', 'phone': '0128472293', 'email': 'vendorC@example.com', 'gender': '', 'vaccine': 'Yes', 'address': '789 Road', 'brandname': '', 'image': 'vendorC.jpg', 'date': '24/6/2024 23:38', 'username': '', 'password': 'VendorPassword123'}	Error message indicating that required fields are missing.	Error message displayed correctly, vendor details not updated due to missing required fields.	Pass

Table 6.5.22: Test Result of Delete Vendor (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC22_1	Valid Vendor ID	{'v_id': '1'}	The system should correctly delete the vendor and provide a success message.	Vendor deleted successfully, success message displayed.	Pass
	Invalid Vendor ID	{'v_id': '0'}	Error message indicating that the vendor ID does not exist.	Error message displayed correctly, vendor not found and no deletion occurred.	Pass

**Table 6.5.23: Test Result of Update Order Status
(Admin)**

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC23_1	Valid Order Update	{'order_id': 'ORDER-366913', 'status': 'closed', 'remark': 'Order completed successfully.'}	The system should correctly update the order status to "closed," set order preparation to "complete," manage payment status, and display a success message.	The order status was updated to "closed," order preparation status was set to "complete," payment status updated to "success," and a success message was displayed.	Pass
	Invalid Order ID	{'order_id': '9999', 'status': 'closed', 'remark': 'Order completed successfully.'}	The system should display an error message indicating the order ID is not found or invalid.	An error message was displayed indicating that the order ID "9999" was not found in the	Pass

				system.	
--	--	--	--	---------	--

Table 6.5.24: Test Result of View Sales Reports (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC25_1	Valid Year and Month	{'year': '2023'}	The system should retrieve and display accurate sales data for the selected year, including the total number of vendors, products, and customers, as well as monthly orders, earnings, order statuses, and payment methods. Data should be visually represented using Chart.js	The system successfully retrieved and displayed sales data for the year 2023. Total vendors, products, and customers were accurately shown. Monthly orders and earnings were correctly calculated and represented in bar and line charts, respectively. Order statuses and payment	Pass

			with error handling and Bootstrap and jQuery integration.	methods were visualized in pie charts. Bootstrap and jQuery were correctly included, and error handling was functional.	
	Invalid Year	{'year': '9999'}	The system should display an error message indicating that the year is invalid or no data is available for the selected year.	The system displayed an error message indicating that the year 9999 is invalid and no data is available.	Pass
	No Data for Selected Year	{'year': '2022'}	The system should indicate that no data is available for the selected year and handle the scenario gracefully.	The system displayed a message indicating that no data is available for the year 2022. The dashboard handled the scenario gracefully without crashing.	Pass

Table 6.5.25: Test Result of View Product Sales (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC25_1	Valid Month	{'month': '2024-08'}	The system should retrieve and display accurate product sales data for August 2024, including proper pagination, sorting, and chart visualization for the top 10 most purchased items.	The system successfully displayed accurate product sales data for August 2024. Pagination and sorting were correct. Top 10 items were visualized using Google Charts with accurate categorization.	Pass
	Invalid Month	{'month': '2024-13'}	The system should display an error message indicating that the month is invalid or no data is available for the selected month.	The system displayed an error message indicating that "2024-13" is an invalid month.	Pass

	No Data for Selected Month	{'month': '2023-07'}	The system should indicate that no data is available for July 2023 and handle the scenario gracefully.	The system displayed a message indicating that no data is available for July 2023 and handled the situation gracefully.	Pass
--	----------------------------	----------------------	--	---	------

Table 6.5.26: Test Result of View Revenue Reports by Payment Method (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC26_1	Valid Month and Year	{'month': 'January', 'year': '2023'}	The system should accurately calculate and display the total revenue and revenue by payment method (Card, Cash, In Store) for January 2023, using the correct date range. Visualization should be	The system correctly calculated and displayed the total revenue and revenue by payment method for January 2023. Bar and donut charts accurately represented the revenue data.	Pass

			provided through bar and donut charts.		
	Invalid Month	{'month': 'InvalidMonth', 'year': '2023'}	The system should display an error message indicating that the month is invalid.	The system displayed an error message indicating that "InvalidMonth" is not a valid month.	Pass

Table 6.5.27: Test Result of Backup and Recover Database (Admin)

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC24_1	Valid Backup and Recovery	{}	The system should successfully create a backup of the database, save it to a specified directory, and recover the database from the latest backup file. Feedback should indicate	The system successfully created a backup of the database, saved it with a timestamped name in the specified directory, and restored the database from the latest backup file. Success messages were displayed, and the user was redirected to the	Pass

			success or failure appropriately.	dashboard.	
	Backup Failure	{invalid database connection details }	The system should terminate with an error message if the database connection fails.	The system correctly terminated with an error message when the database connection details were invalid.	Pass
	Recovery Failure	{no backup file available }	The system should display an error message if no backup file is available for recovery.	The system displayed an error message indicating that no backup file was available for recovery.	Pass

**Table 6.5.28: Test Result of Power BI Dashboard
Integration (Admin)**

Test Case ID	Test Scenario	Input Data	Expected Result	Actual Result	Status
TC27_1	Valid Power BI Integration	{ }	The Power BI dashboard link should be correctly rendered on the admin panel and redirect to the specified Power BI report URL. The report should load without errors and be accessible only to logged-in admin users.	The Power BI dashboard link was correctly rendered on the admin panel and redirected to the specified Power BI report URL. The report loaded without errors and matched the live database data. The link was only accessible to logged-in admin users.	Pass
	Data Accuracy Verification	{ }	The Power BI dashboard should display accurate real-time data that matches the live database.	The Power BI dashboard displayed real-time data accurately, matching the live database.	Pass

	Unauthorized Access	{non-admin user}	The Power BI dashboard link should not be visible or accessible to unauthorized users or users without admin privileges.	The Power BI dashboard link was not visible or accessible to non-admin users.	Pass
--	---------------------	------------------	--	---	------

6.6 Conclusion

In conclusion, system testing is an important part of Agile development. Agile focuses on frequent testing and feedback to catch errors and defects early. For Sweet Scape Vendor Dessert Management System, white box testing is used to check the internal logic and code. This chapter covers how test cases, test data, and test results are designed and documented to ensure that the software is properly tested in each development cycle.

CHAPTER 7: PROJECT CONCLUSION

7.1 Introduction

This chapter provides an overall conclusion for Sweet Scape Vendor Dessert Management System (SSVDMS). It includes an analysis of the system's strengths and weaknesses. Additionally, it will propose improvements based on this analysis and outline the contributions of the project.

7.2 Observation Weakness and Strengths

Each system consists of its own strengths and weaknesses. Section 7.2.1 will explain in detail the system strength while 7.2.2 will explain the system weaknesses.

7.2.1 Strength

The strength of the Sweet Scape Vendor Dessert Management System (SSVDMS) is:

I. Improved Sales Overview with Business PowerBI.

The system makes it easy to see sales data using Business PowerBI. Users can access this information from anywhere with an internet connection. The system is designed to be user-friendly, with clear navigation buttons that show their functions. This helps users easily find and understand the sales data, making it simpler to analyze and make decisions.

II. High-Quality Standards Monitoring.

The system requires vendors to provide the production and expiry dates for each product. This ensures that staff have up-to-date information about the product's freshness and shelf life. By having this information readily available, staff can request vendor to check the quality of products before they expire, ensuring that only fresh and high-quality items are provided to customers.

7.2.2 Weaknesses

The weakness of the Sweet Scape Vendor Dessert Management System (SSVDMS) is:

7.2.2.1 Lack of Rating Product after purchase.

The system does not include a feature for customers to rate or leave feedback on products after purchase. This means there is no way to collect customer opinions or reviews on the quality of the products. Without this feedback, it is harder to understand customer satisfaction and make improvements based on their experiences

7.2.2.2 Profit Calculation for Seller

The system currently lacks a feature to calculate the profit for sellers who sell vendor products. Adding a profit calculation function would help sellers understand their earnings by subtracting the cost of products from their selling prices. This enhancement would provide sellers with clear insights into their financial performance, making it easier to manage their business and make informed decisions based on profitability.

7.3 Propositions of Improvement

The following are statements of improvement for Sweet Scape Vendor Dessert Management System based on the strengths and weaknesses above.

7.3.1 Add Rating and Feedback.

Add a feature that lets customers rate and leave feedback on products after they make a purchase. This will help collect customer opinions and understand their satisfaction levels. It will also provide valuable information to improve products based on real customer experiences.

7.3.2 Implement Profit Calculation Feature

Develop a module to automatically calculate and display the profit for each seller by subtracting the cost of goods from the selling price.

7.4 Conclusion

Sweet Scape successfully addresses key challenges in the dessert retail industry. The system helps sellers manage product quality and vendor relationships more effectively. By requiring vendors to input product details and allowing staff to monitor these details, Sweet Scape ensures that only high-quality desserts are offered to customers. The system also improves collaboration between vendors and sellers and provides valuable insights through Power BI reports.

Despite its strengths, there are still some weaknesses in the system that can be improved in the near future, such as adding features for customer feedback and enhancing tracking capabilities. Overall, Sweet Scape meets its objectives of improving quality management, simplifying vendor tasks, and providing useful analytics. It has the potential to significantly benefit sellers, vendors, and customers in the dessert industry. Future developments will focus on adding new features to keep up with industry needs and enhance user satisfaction. In conclusion, this PSM has been successfully completed and fulfilled the requirement for a Bachelor of Computer Science (Database Management).

REFERENCES

Kumar, G., & Bhatia, P. K. (2012). Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(4), 46-50.

Hutchinson, A. (2012). Card payment implementation guide for ASP. NET and PHP websites.

Davis, M. E., & Phillips, J. A. (2007). *Learning PHP & MySQL: Step-by-Step Guide to Creating Database-Driven Web Sites*. " O'Reilly Media, Inc."



APPENDIX A

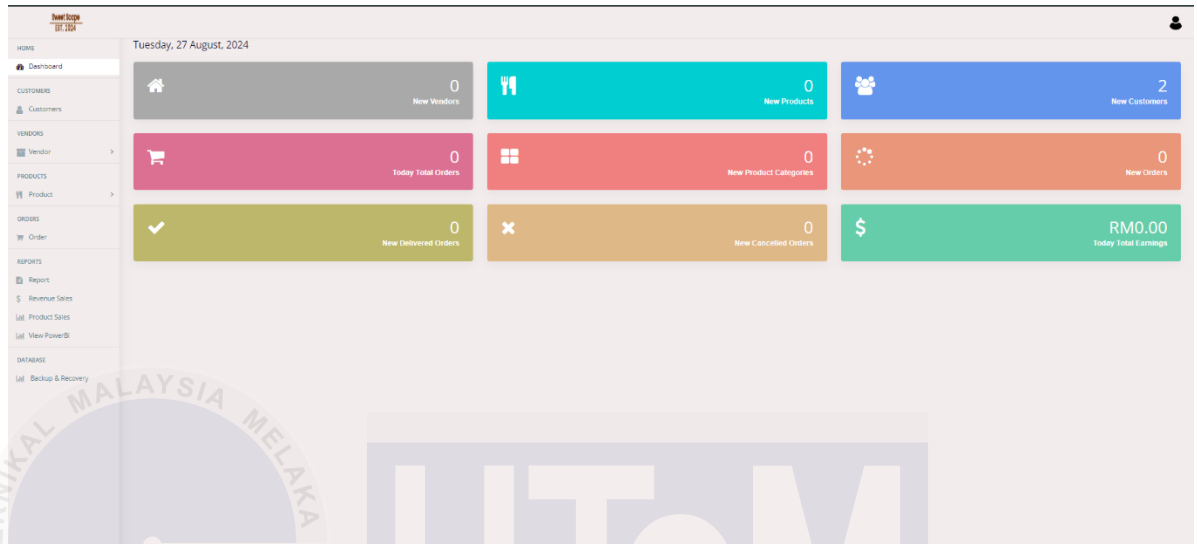


Figure 4.1.1.1: Admin Dashboard Page

اونيورسيتي تيكنيكل مليسيا ملاك

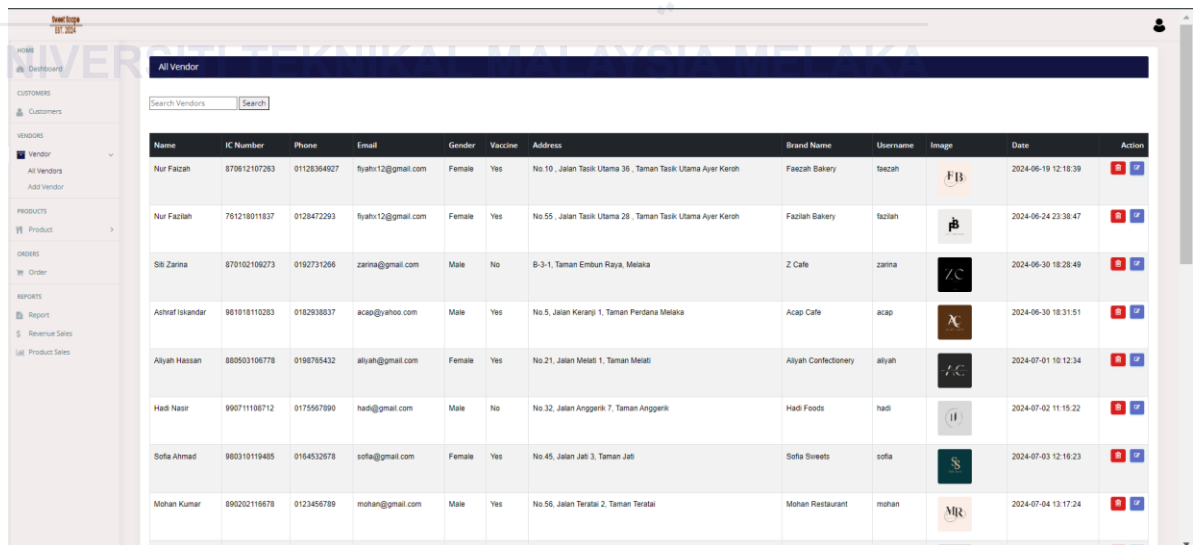


Figure 4.1.1.2: Admin All Vendor Page

Add Vendor

Vendor Name IC Number

Phone Number Email

Gender Vaccine

Address Brand Name

Image Username

Password

Figure 4.1.1.3: Admin Add Vendor Page

Update Vendor

Vendor Name IC Number

Phone Email

Gender Vaccine

Address Brand Name

Username Password

Image

Figure 4.1.1.4 : Admin Update Vendor Page

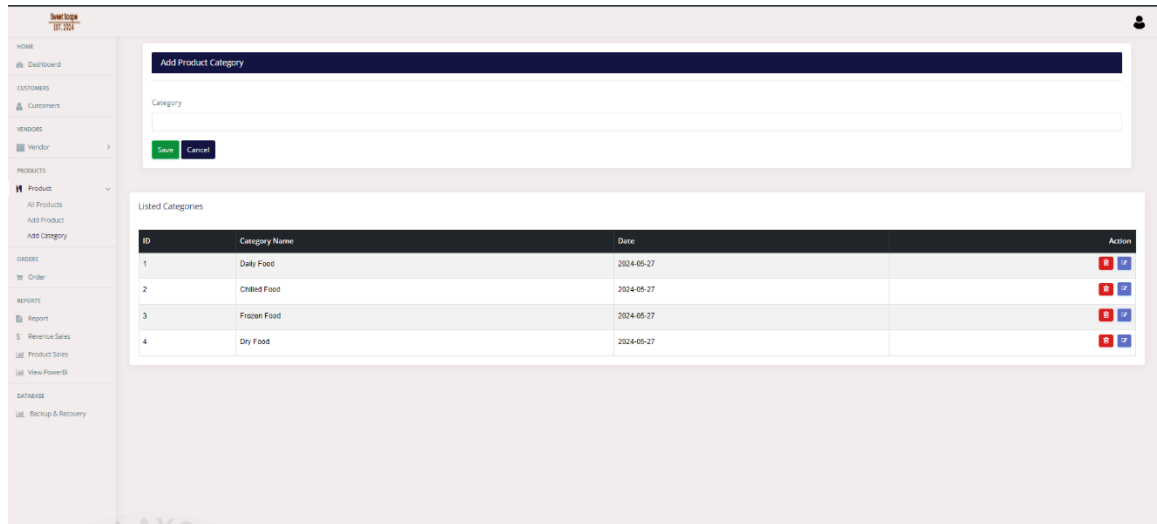


Figure 4.1.1.5: Admin Add Product Category Page



Figure 4.1.1.6: Admin Update Category Page

Name	Category	Ingredients	Status	Request Form	Action
Mini Burger Malaysia	Daily Food	Air Suam, Yis Kering, Gula Pasir, Tepung Gandum, Mentega, Susu Pekat Manis, Baking Powder, Baking Soda, Ikan Bilis, Bawang Besar, Ses Cili, Gula, Garam, Minyak Matak, Cili Kering, Bawang Besar, Bawang Putih	Available	<input type="checkbox"/>	<input type="checkbox"/>
Vietnamese Roll Kalam	Daily Food	Kulit Popia Vietnam (Rice Paper Wrapper), Suhun, Dada Ayam, Udang, Daun Salad, Timun, Lobak Merah, Daun Ketumbar, Daun Bawang, Minyak Sayur, Bawang Putih, Sos Hoin, Air Rebusan Ayam, Mentega Kacang, Gula, Kacang Tanah, C	Available	<input type="checkbox"/>	<input type="checkbox"/>
Vietnamese Roll Udang	Daily Food	Kulit Popia Vietnam (Rice Paper Wrapper), Suhun, Dada Ayam, Udang, Daun Salad, Timun, Lobak Merah, Daun Ketumbar, Daun Bawang, Minyak Sayur, Bawang Putih, Sos Hoin, Air Rebusan Ayam, Mentega Kacang, Gula, Kacang Tanah	Available	<input type="checkbox"/>	<input type="checkbox"/>
Crab Ragoon	Daily Food	Cream Cheese, Crabstick, Daun Bawang, Garlic, Minyak Biji, Garam, Kulit Wontan	Available	<input type="checkbox"/>	<input type="checkbox"/>
Boko Ubi	Daily Food	Ubi, Pati Ubi, Air, Gula Merah, Gula Pasir, Daun Pandan	Available	<input type="checkbox"/>	<input type="checkbox"/>
Tauhu Bergedil	Daily Food	Tauhu Kering, Bawang Besar, Daging Kisar, Kentang, Cili Merah, Daun Sup, Bawang Goreng, Lada Sulah, Garam, Telur, Bawang Merah, Bawang Putih, Cili Padi, Minyak, Kicap Manis ABC, Kicap Manis Cap Kipas Udang	Available	<input type="checkbox"/>	<input type="checkbox"/>
Creamput Custard	Chilled Food	Air, Mentega, Gula, Creamy Vanilla, Tepung Gandum, Telur Gred A, Susu Full Krim, Tepung Kastard, Non-Dairy Whipping Cream	Available	<input type="checkbox"/>	<input type="checkbox"/>
Pizza Homemade Frozen	Frozen Food	Roti Pizza, Tomato Cherry, Cili Hijau, Soes, Sos Tomato Prego, Kaju Mozzarella	Available	<input type="checkbox"/>	<input type="checkbox"/>
Mantao Frozen	Frozen	Mantao	Available	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4.1.1.7: Admin All Product Page

Add Product

Product Name:

Sizes, Prices, Quantities:

Select Size	Price	Quantity
<input type="text"/>	<input type="text"/>	<input type="text"/>

Production Date:

Expiry Date:

Flavour:

Category:

Image:

Brand Name:

Figure 4.1.1.8 : Admin Add Product Page

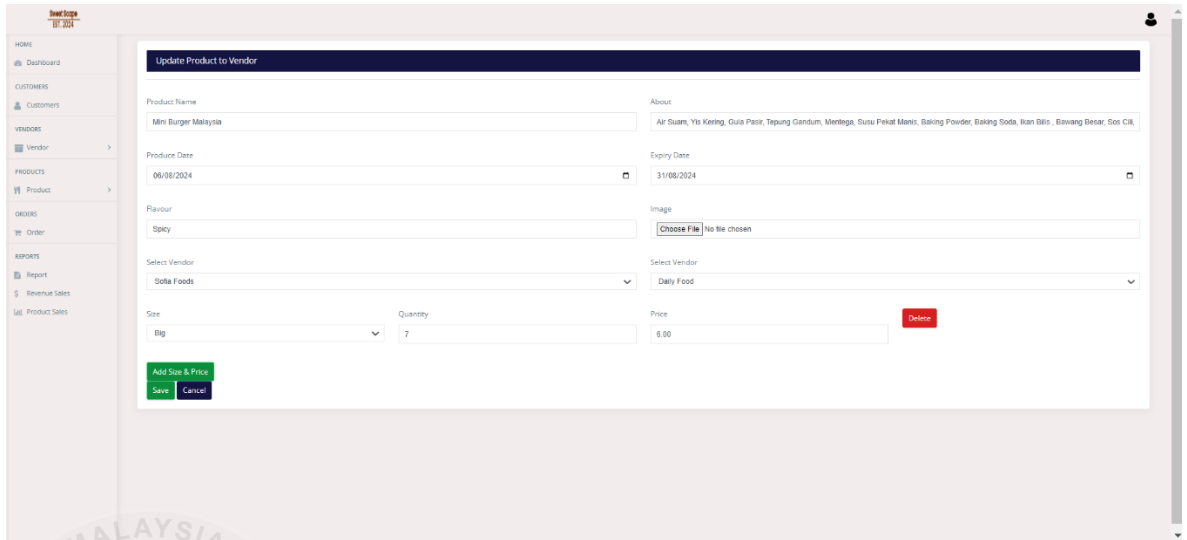


Figure 4.1.1.9: Admin Update Product Page

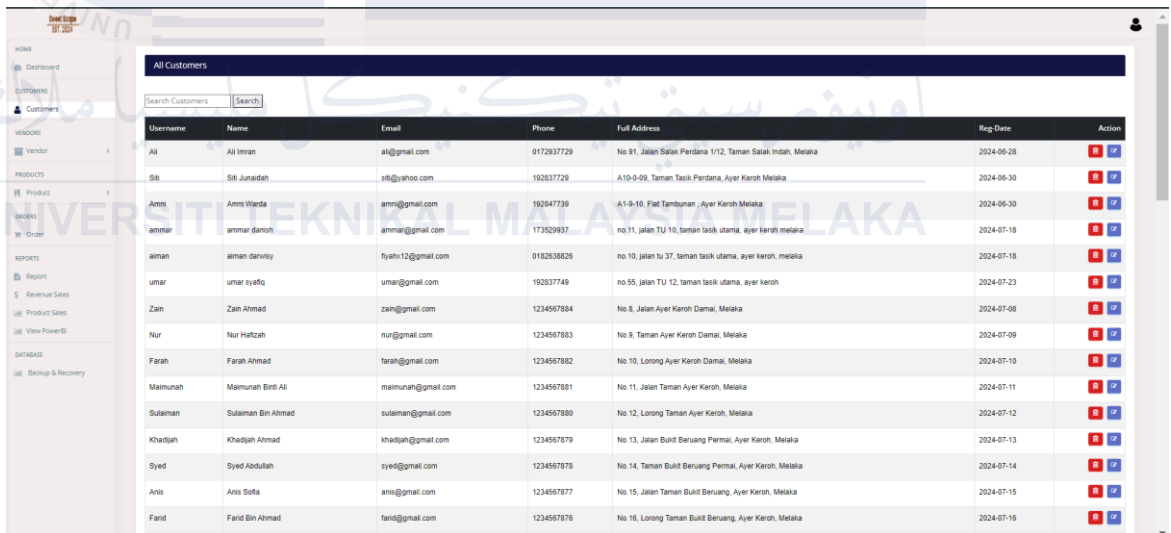


Figure 4.1.1.10: Admin All Customer Page

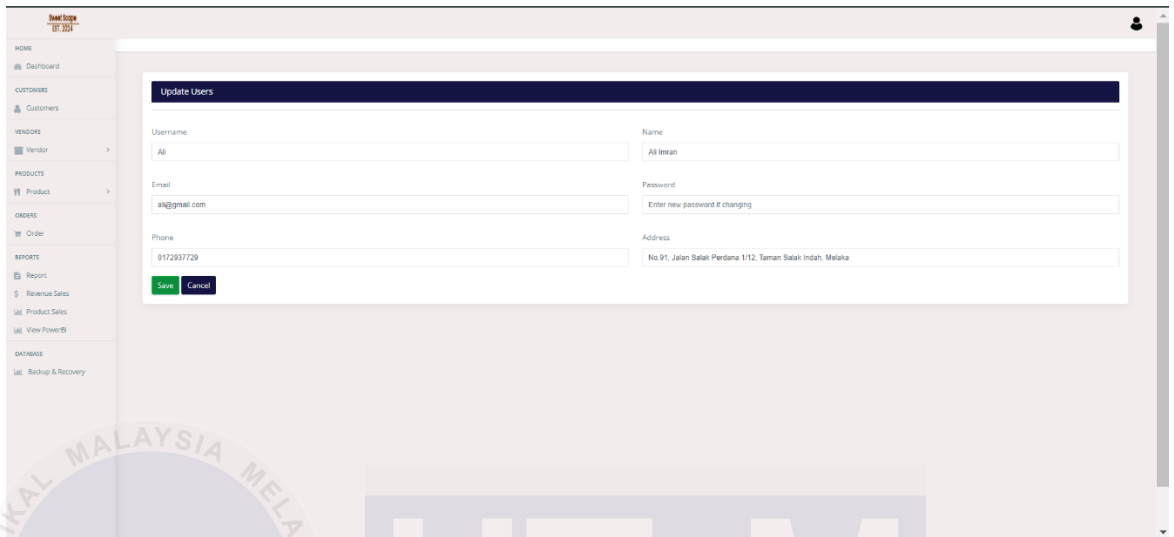


Figure 4.1.1.11: Admin Update Customer Page

Customer	Products	Payment Option	Delivery Option	Total Price	Address	Status	Order Date	Action
aiman danwily	Mochi Frozen	cash	delivery	RM35.00	no 10, jalan tu 37, taman tasik utama, ayer keroh, melaka	Delivered	2024-08-25 21:04:52	[Red] [Blue]
aiman danwily	Ayam Susu Thai Frozen	cash	delivery	RM35.00	no 10, jalan tu 37, taman tasik utama, ayer keroh, melaka	Delivered	2024-08-24 05:09:42	[Red] [Blue]
aiman danwily	Lai Chee Kang	In Store	pickup	RM10.00	no 10, jalan tu 37, taman tasik utama, ayer keroh, melaka	Delivered	2024-08-24 04:54:02	[Red] [Blue]
Khadjan Ahmad	Udang Mental	Card	delivery	RM85.00	No 53, Jalan Bukit Beruang Permai, Ayer Keroh, Melaka	Delivered	2024-07-24 21:02:00	[Red] [Blue]
Anis Sofia	Ayam Korea	Card	delivery	RM69.00	No 55, Jalan Taman Bukit Beruang, Ayer Keroh, Melaka	Cancelled	2024-07-24 19:51:00	[Red] [Blue]
Azhanna binti Khalid	Kerepek Ubi Pedas, Crab Ragoon	Cash	delivery	RM35.00	No 24, Jalan Bukit Ayer Keroh Indah, Melaka	Cancelled	2024-07-24 18:52:00	[Red] [Blue]
Zain Ahmad	Kek Tapak Kurda Nubelia, Daging Harimau Menangis Bakar, Mini Chocpai Coklat, Kerepek Cheese	Cash	delivery	RM176.00	No 8, Jalan Ayer Keroh Damai, Melaka	Delivered	2024-07-24 18:36:00	[Red] [Blue]
Ismail Bin Ahma	Vietnamese Roll Ketam, Kek Tornado	In Store	pickup	RM100.00	No 114, Jalan Taman Ayer Keroh Megah, Melaka	Delivered	2024-07-24 17:43:00	[Red] [Blue]
Salmah Binti Ahmad	Condough Cheese Frozen, Cheese Tart Blueberry, Latok Frozen, Mairtao with Butter Cream Clams	In Store	pickup	RM225.00	No 66, Lorong Taman Ayer Keroh Maju, Melaka	On The Way!	2024-07-24 17:17:00	[Red] [Blue]
Ismail Bin Ahma	Vietnamese Roll Ketam, Koktail Buah, Mochi Coklat, Rori Sarang Lebah	In Store	pickup	RM191.00	No 114, Jalan Taman Ayer Keroh Megah, Melaka	Delivered	2024-07-24 16:51:00	[Red] [Blue]
Sulaiman Bin Ahmad	Salmon Mental	In Store	pickup	RM45.00	No 52, Lorong Taman Ayer Keroh, Melaka	Delivered	2024-07-24 15:57:00	[Red] [Blue]
Hakim Bin Ahmad	Hass Ek Botol Bandung Pink, Ayam Bakar, Kek Tornado, Bangkit Cheese	In Store	pickup	RM177.00	No 20, Taman Bukit Ayer Keroh Utama, Melaka	Delivered	2024-07-24 14:21:00	[Red] [Blue]
Sarah Muhammad	Spagetti Cabonara	In Store	pickup	RM36.00	No 47, Lorong Ayer Keroh Haightt, Melaka	Delivered	2024-07-24 14:13:00	[Red] [Blue]
Shafiq Bin Ahmad	Coklat B40 Coklat	Cash	delivery	RM33.00	No 96, Lorong Bukit Ayer Keroh Damai, Melaka	Cancelled	2024-07-24 14:07:00	[Red] [Blue]

Figure 4.1.1.12: Admin All Order Page

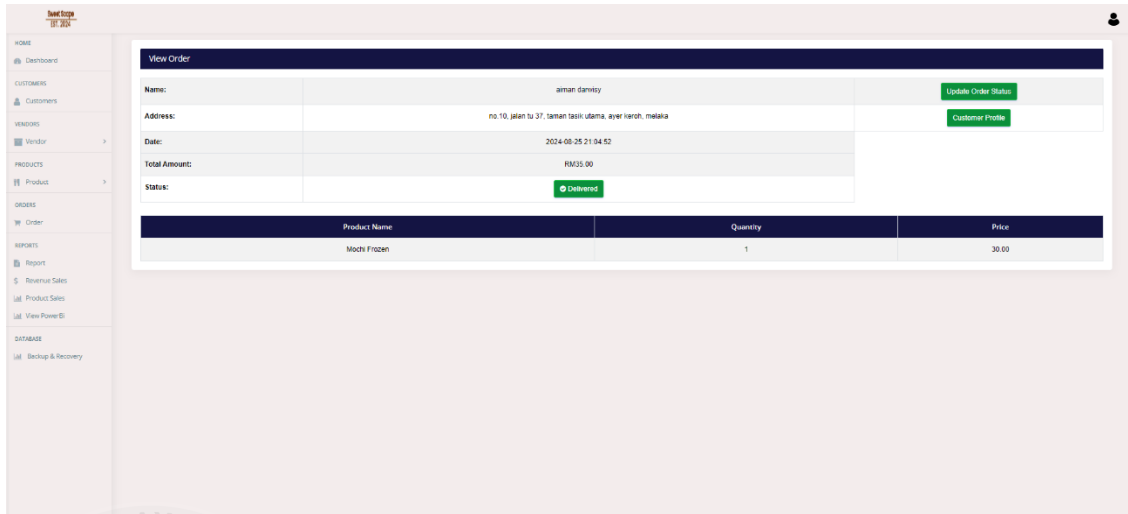


Figure 4.1.1.13: Admin View Order Page



Figure 4.1.1.14: Admin Order Update Page

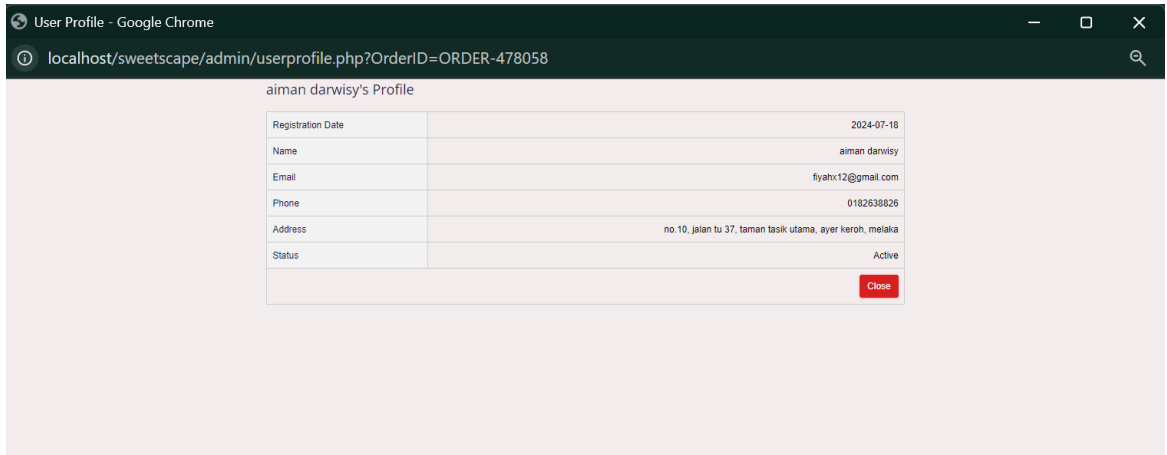


Figure 4.1.1.15: Admin Customer Profile by OrderID Page

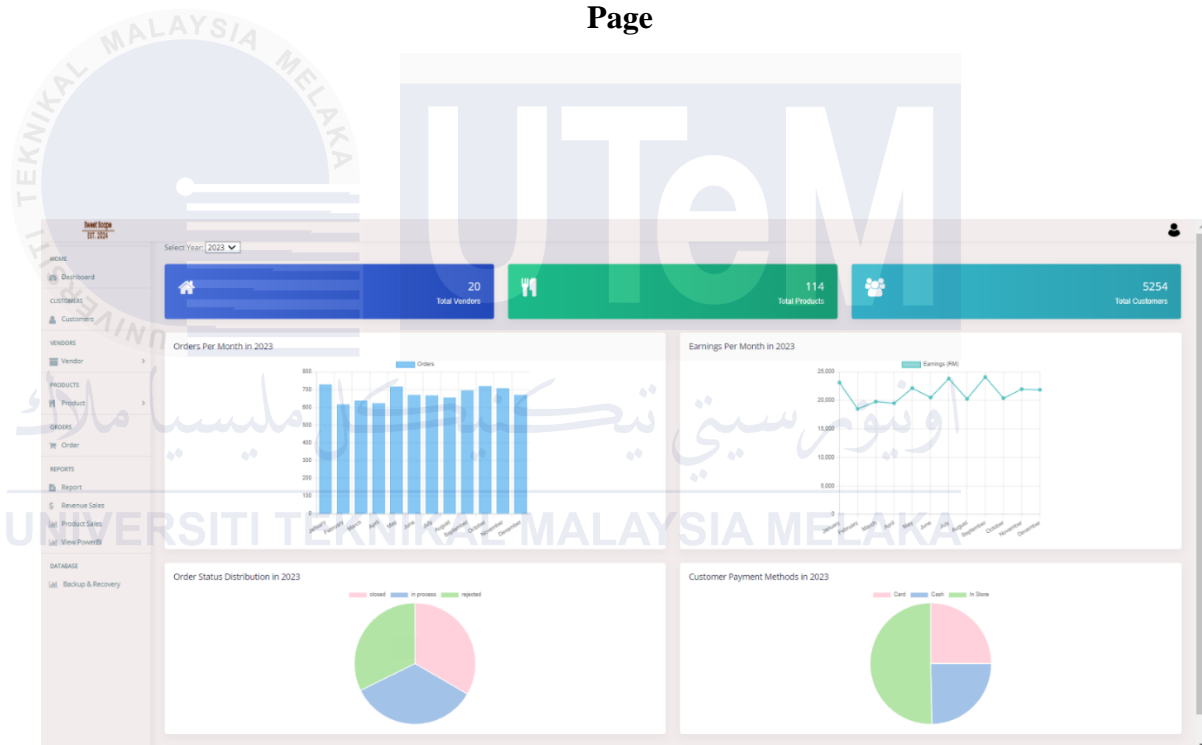


Figure 4.1.1.16: Admin Report Page

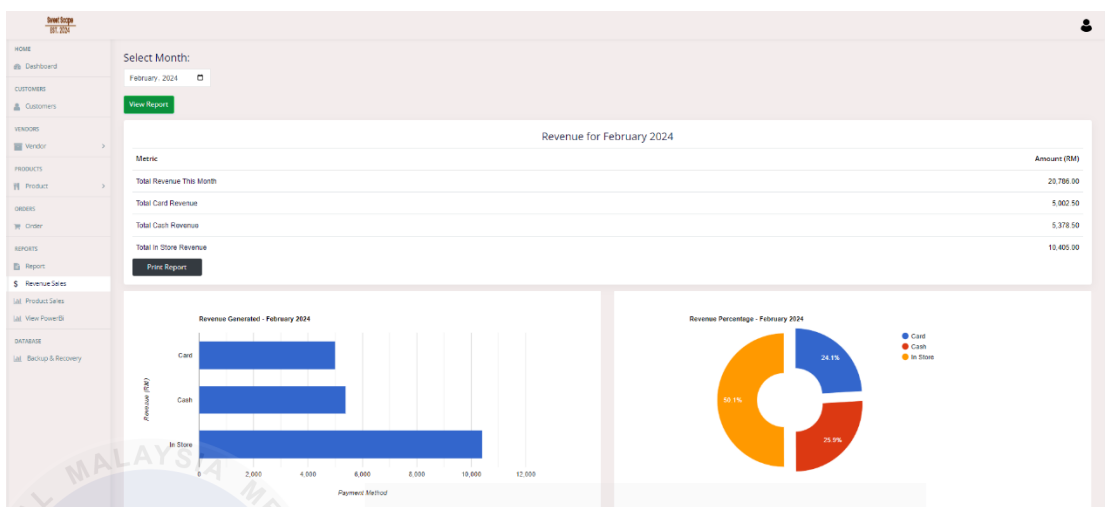


Figure 4.1.17: Admin Revenue Sales Page

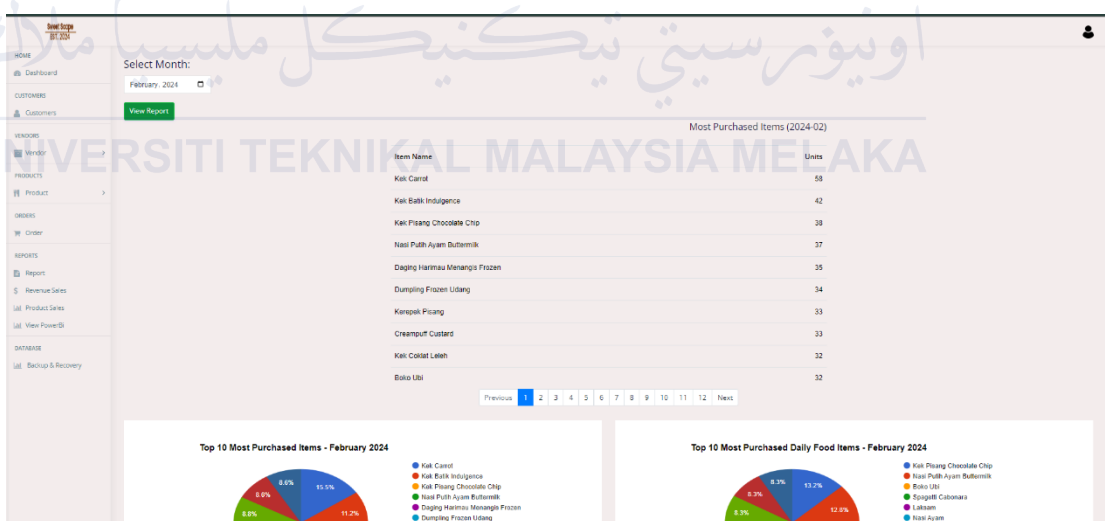


Figure 4.1.18: Admin Product Sales (1) Page

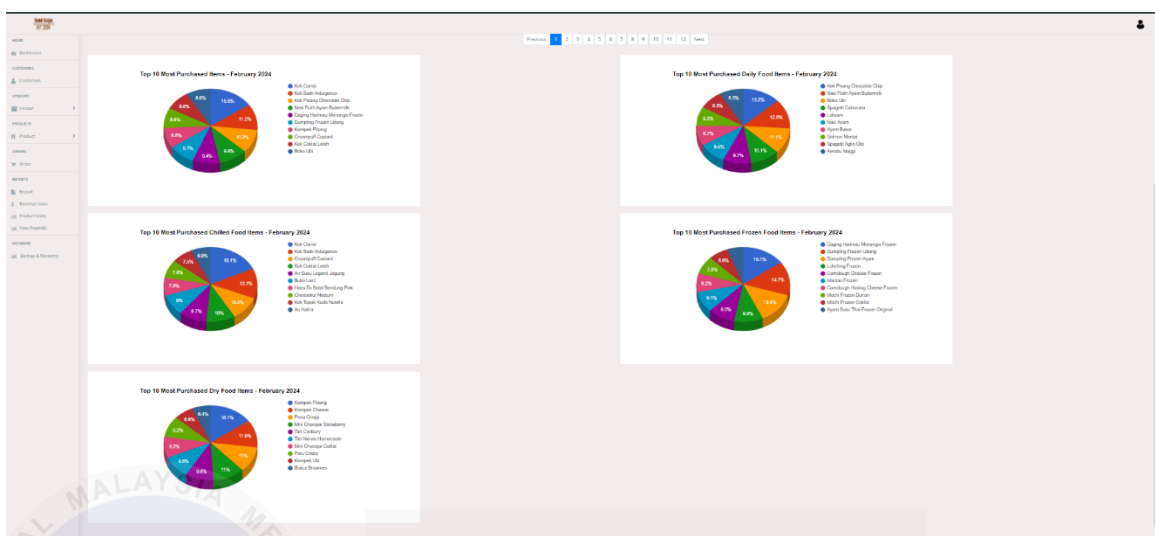


Figure 4.1.1.19: Admin Product Sales (2) Page

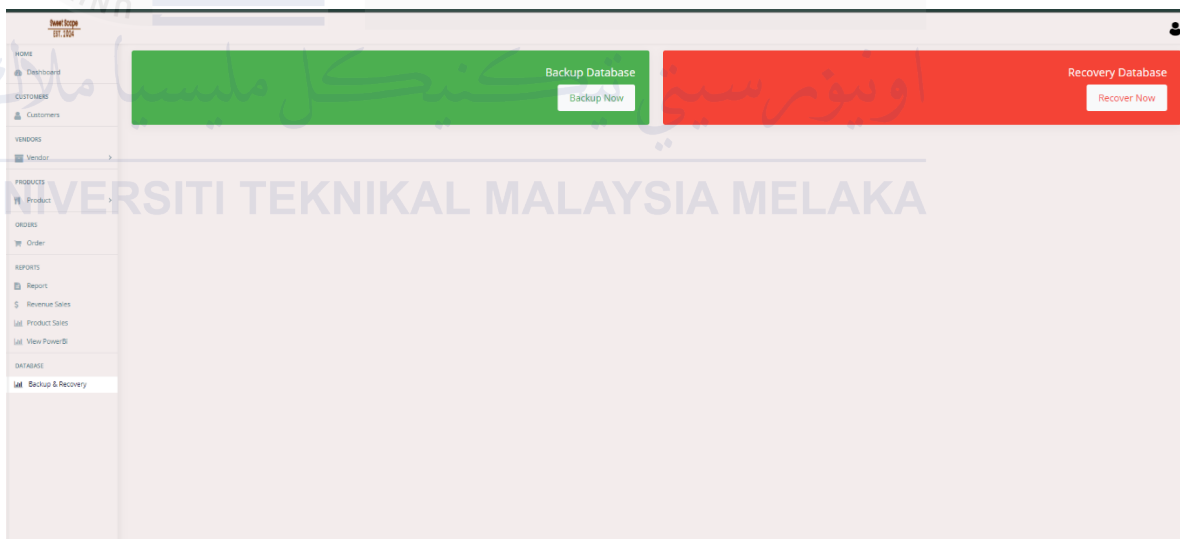
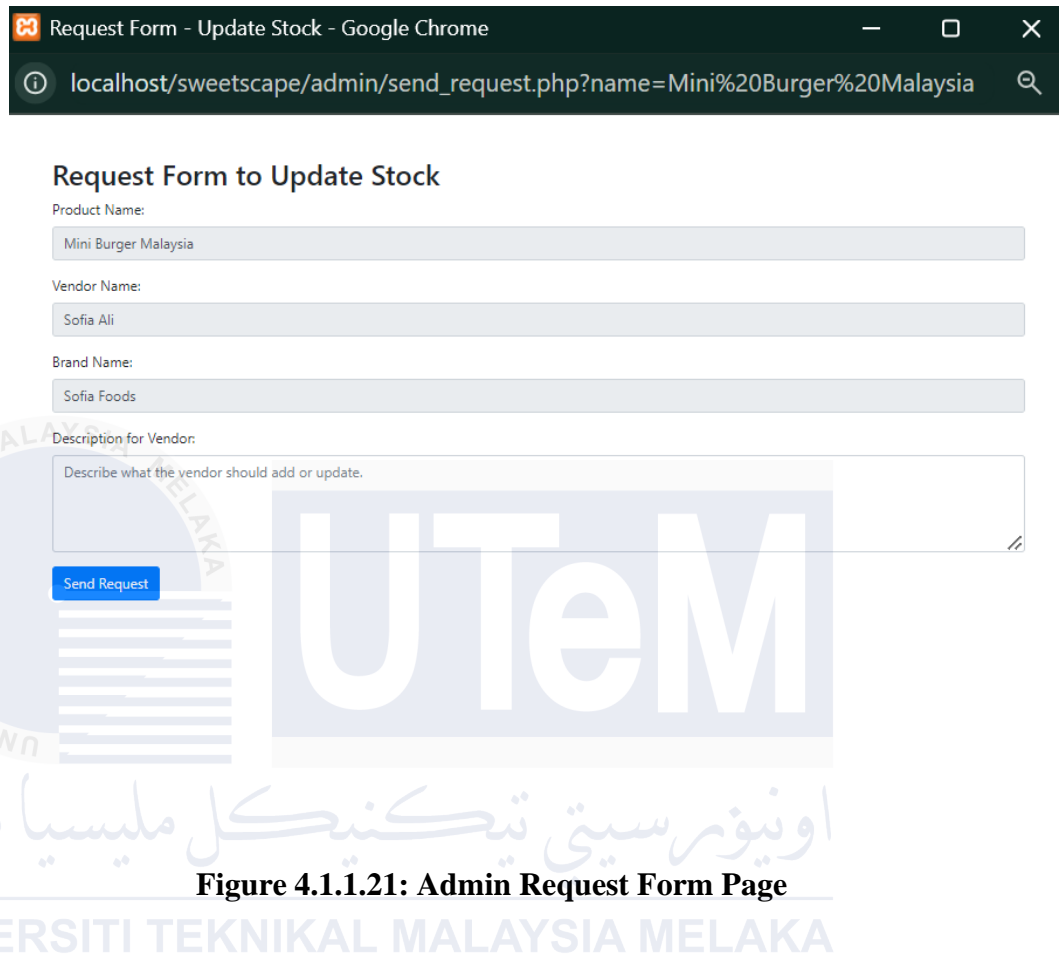


Figure 4.1.1.20: Admin Backup Recovery Page



Request Form - Update Stock - Google Chrome

localhost/sweetscape/admin/send_request.php?name=Mini%20Burger%20Malaysia

Request Form to Update Stock

Product Name:
Mini Burger Malaysia

Vendor Name:
Sofia Ali

Brand Name:
Sofia Foods

Description for Vendor:
Describe what the vendor should add or update.

Send Request

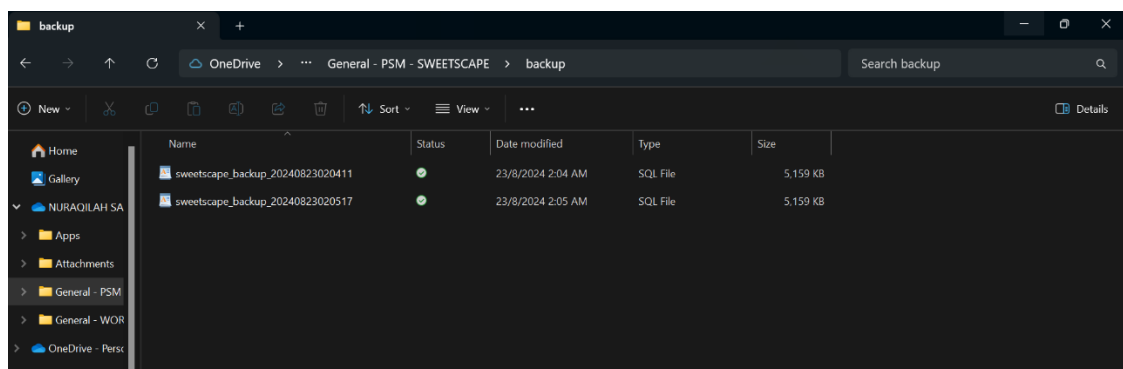
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UTeM

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 4.1.1.21: Admin Request Form Page



Name	Status	Date modified	Type	Size
sweetscape_backup_20240823020411	✓	23/8/2024 2:04 AM	SQL File	5,159 KB
sweetscape_backup_20240823020517	✓	23/8/2024 2:05 AM	SQL File	5,159 KB

Figure 4.1.1.22: File Backup Page

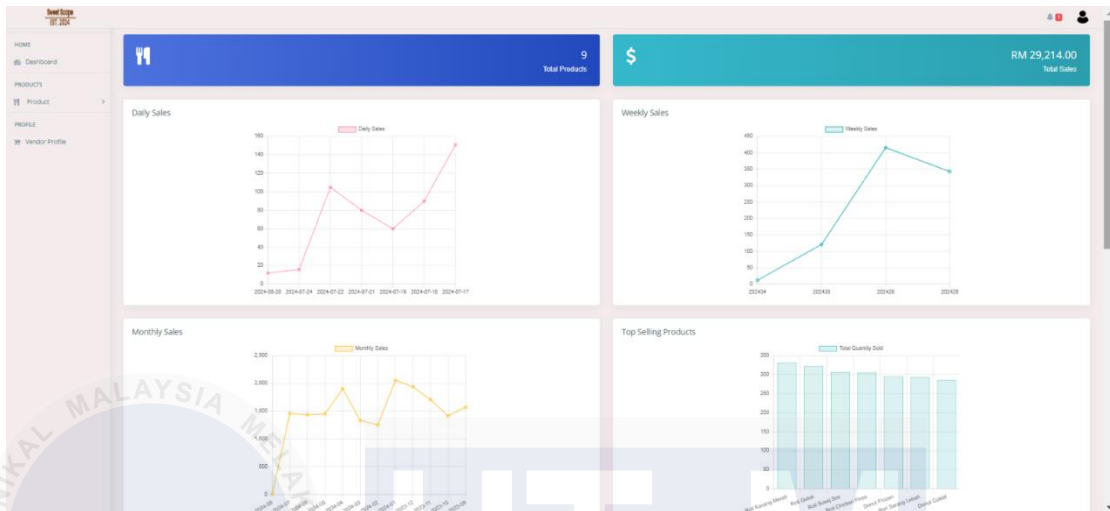


Figure 4.1.1.23: Vendor Dashboard (1) Page

اونيورسيتي تيكنيكل مليسيا ملاك

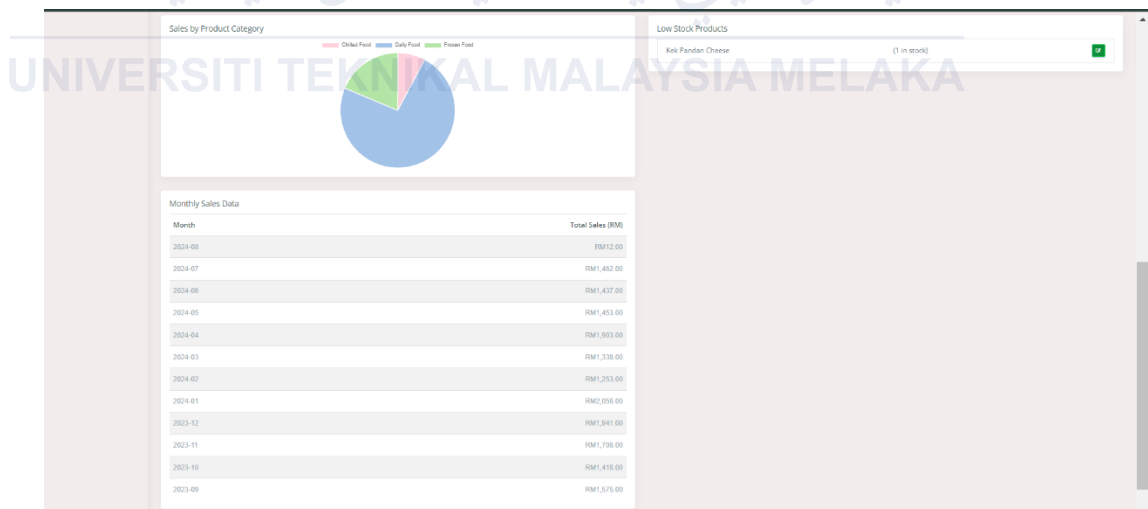


Figure 4.1.1.24: Vendor Dashboard (2) Page

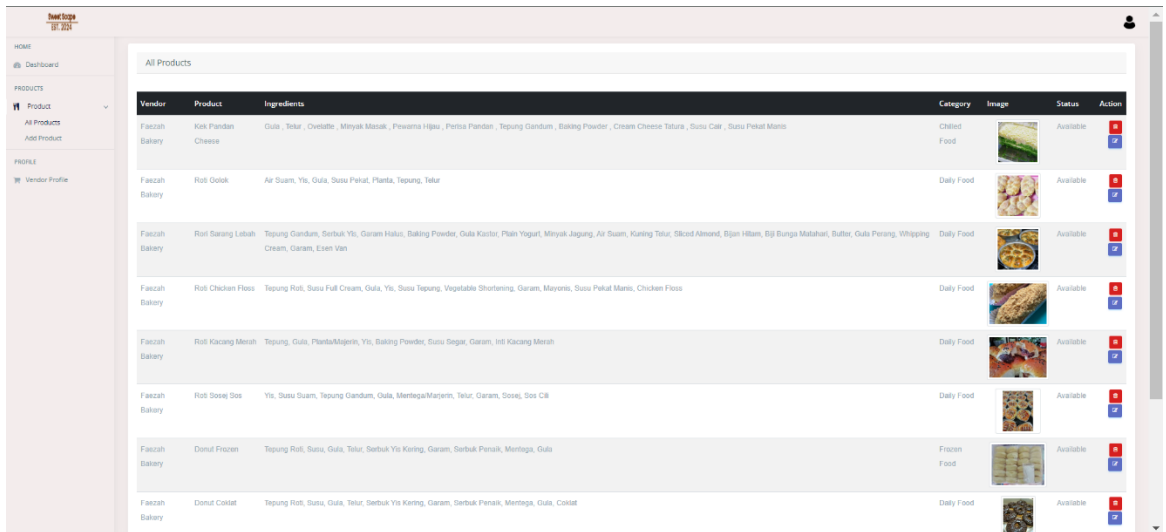


Figure 4.1.1.25: Vendor All Product Page

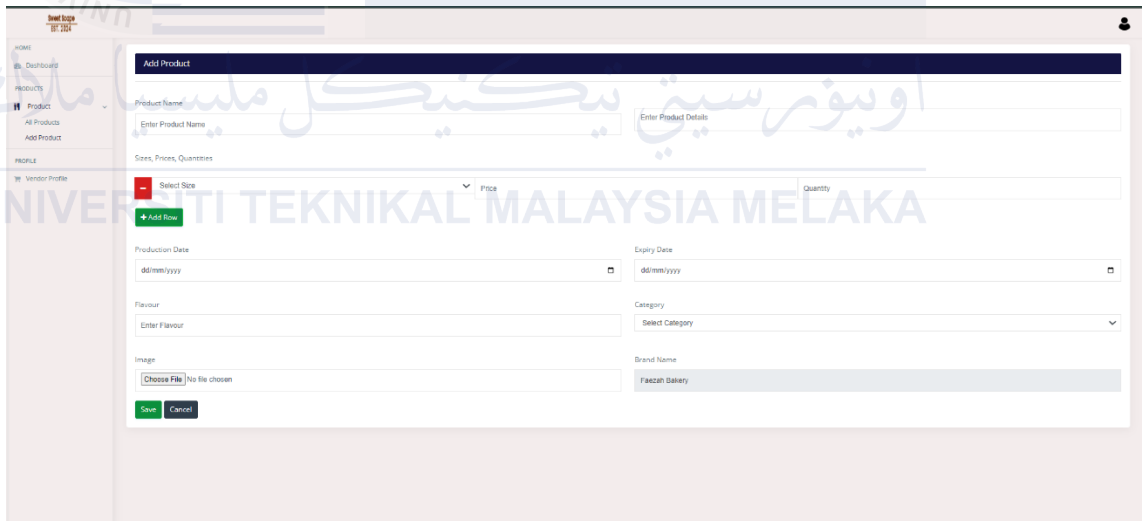


Figure 4.1.1.26: Vendor Add Product Page

Update Product to Vendor

Product Name: Roti Chicken Floss

Produce Date: 01/08/2024

Expiry Date: 31/08/2024

Flavour: Original

Brand Name: Faecah Bakery

Size: Small

Quantity: 8

Price: 20.00

Buttons: Add Size & Price, Save, Cancel, Remove

Figure 4.1.1.27: Vendor Update Product Page

Update Vendor

Vendor Name: Nur Fatmah

Brand Name: Faecah Bakery

Phone: 01128384027

Email: fyshv12@gmail.com

IC Number: 820612107263

Gender: Female

Address: No. 10, Jalan Tasik Utama 36, Taman Tasik Utama Ayer Keroh

Upload Image: Choose File

Username: faecah

Password: Leave blank if not changing

Buttons: Save, Cancel

Figure 4.1.1.28: Vendor Update Profile Page



Figure 4.1.1.29: Vendor Notification Page

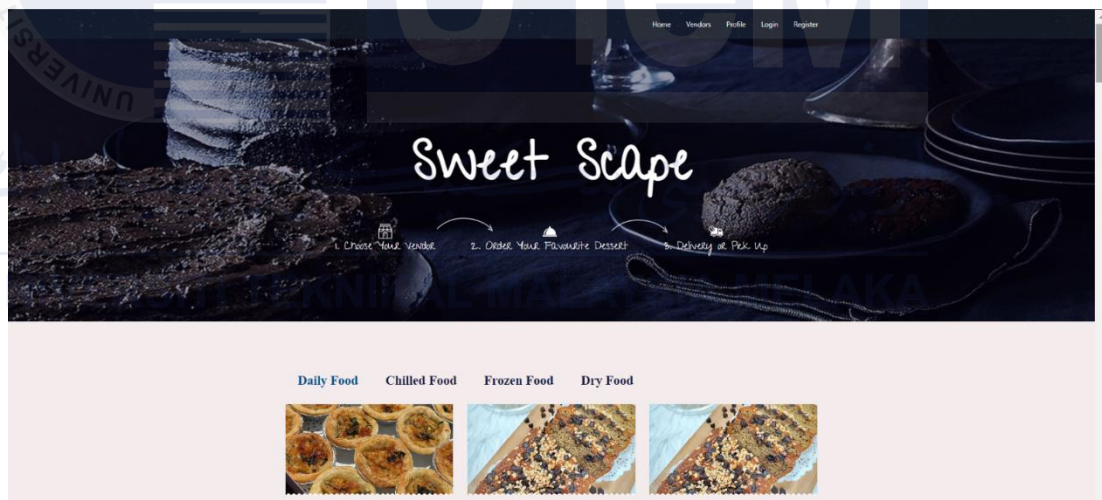


Figure 4.1.1.30: Customer Index (1) Page

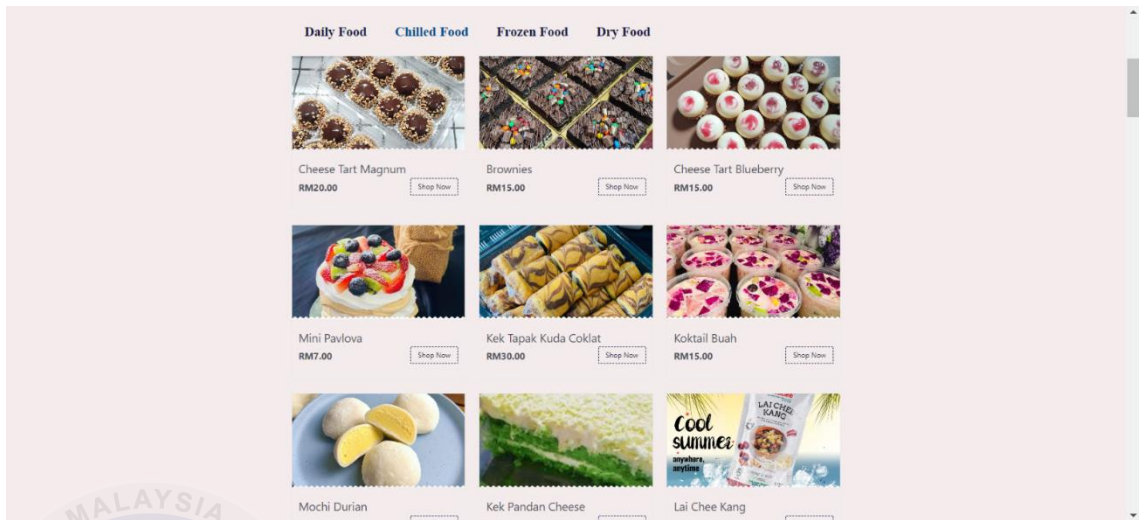


Figure 4.1.1.31: Customer Index (2) Page

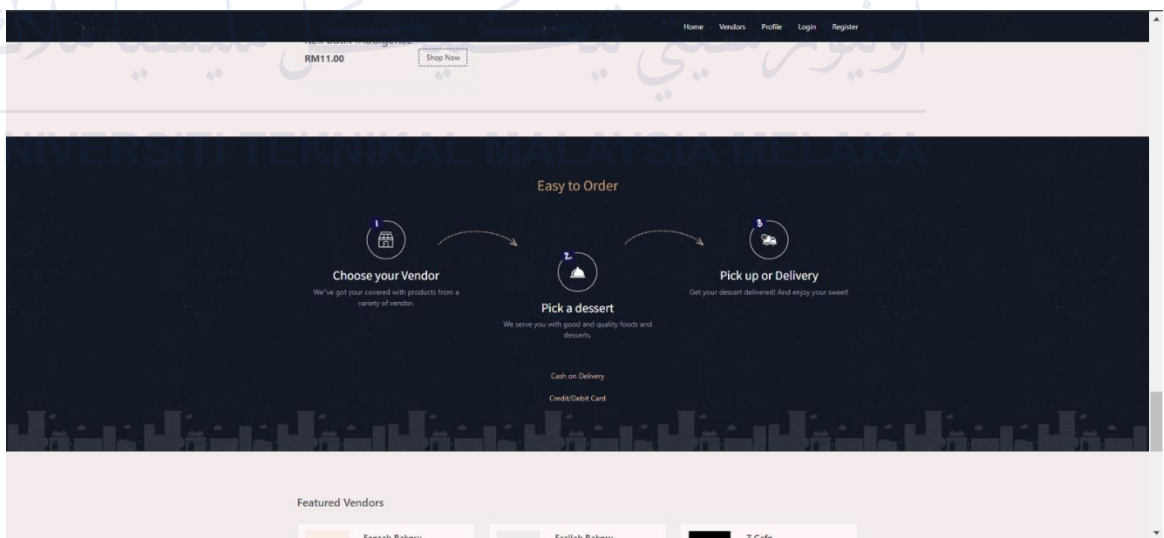


Figure 4.1.1.32: Customer Index (3) Page

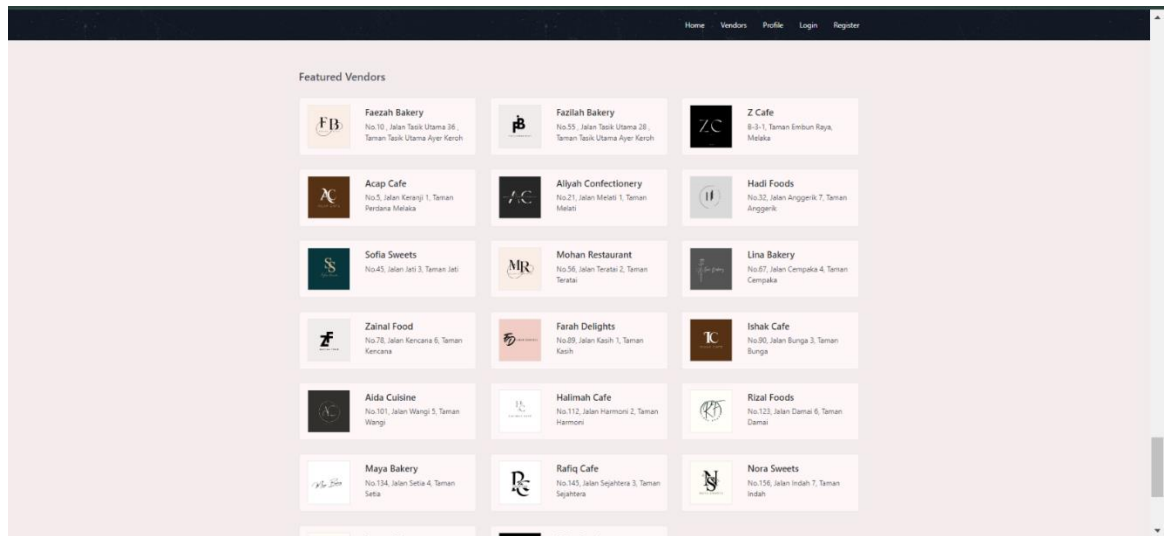


Figure 4.1.1.33: Customer Index (4) Page



Figure 4.1.1.34: Customer Index (5) Page

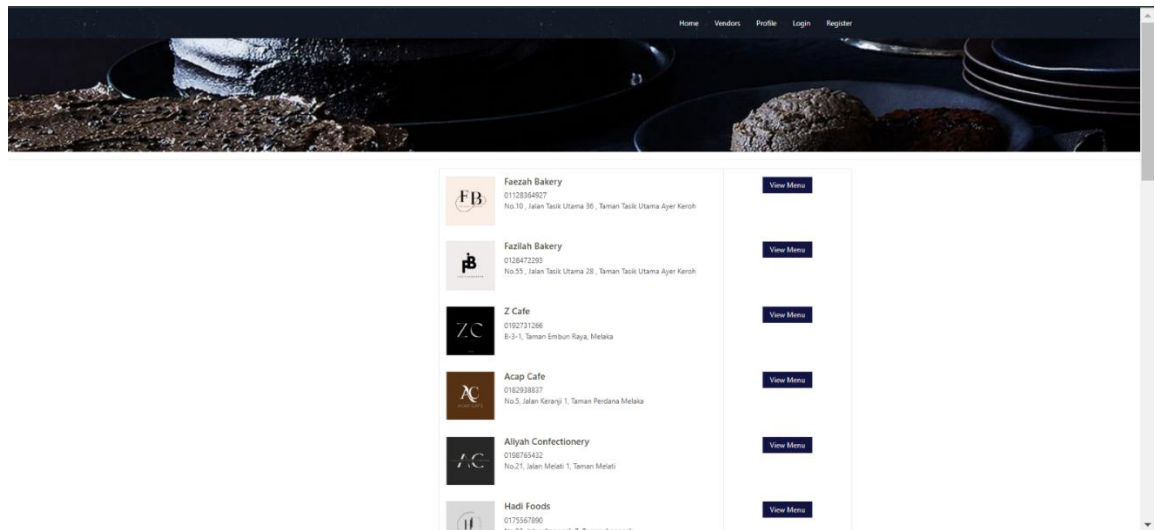


Figure 4.1.1.35: Customer Vendors Page

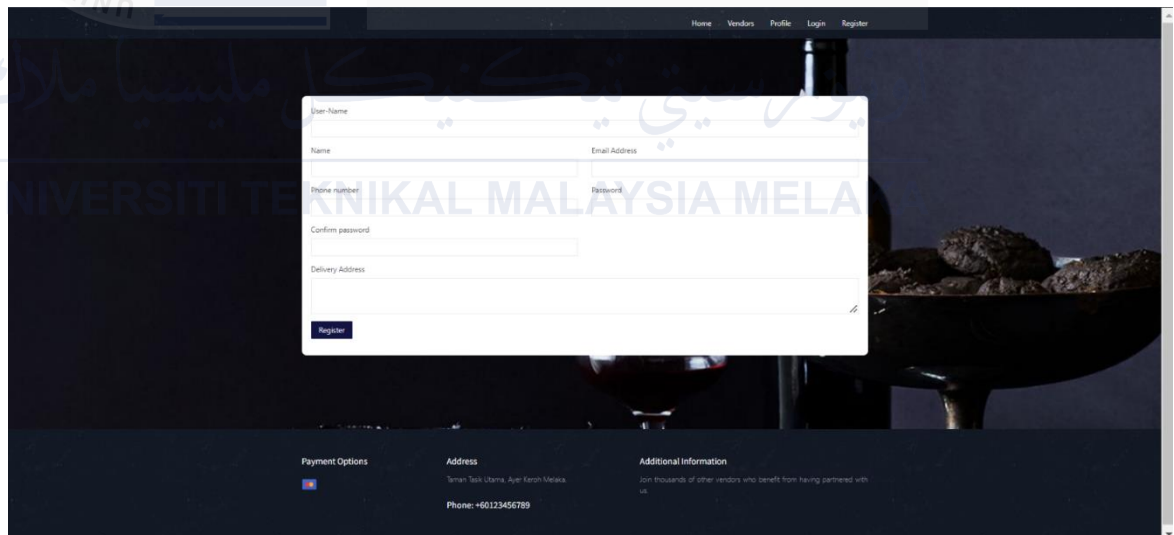


Figure 4.1.1.36: Customer Registration Page

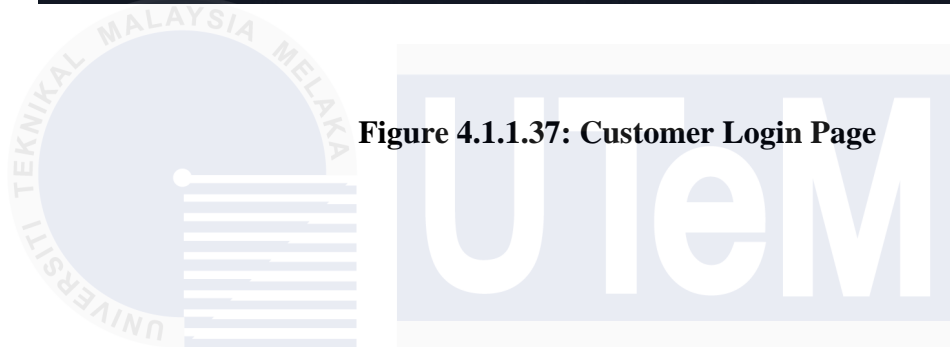
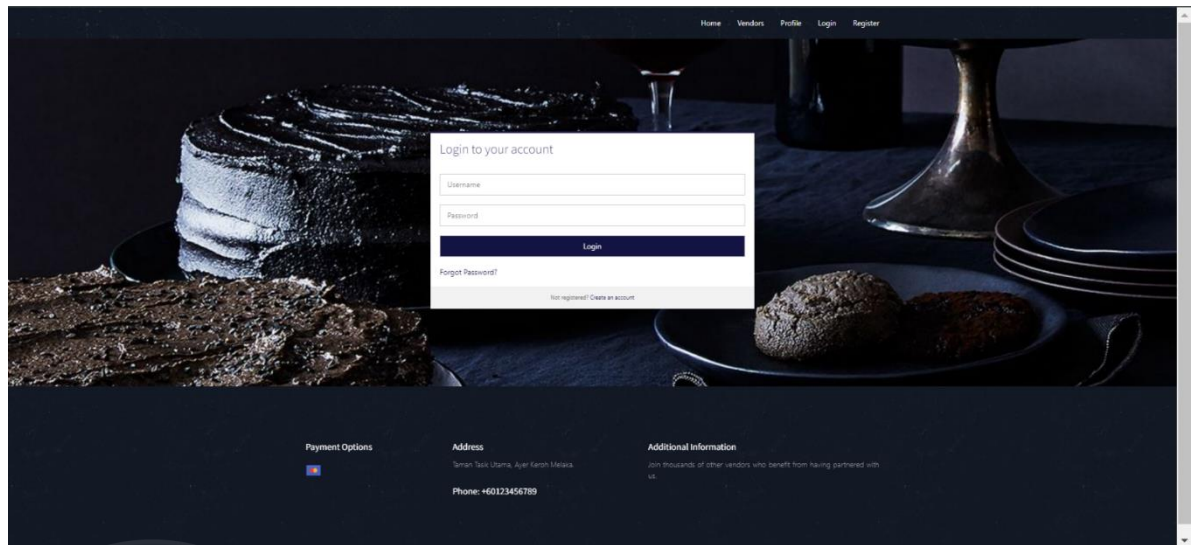


Figure 4.1.1.37: Customer Login Page

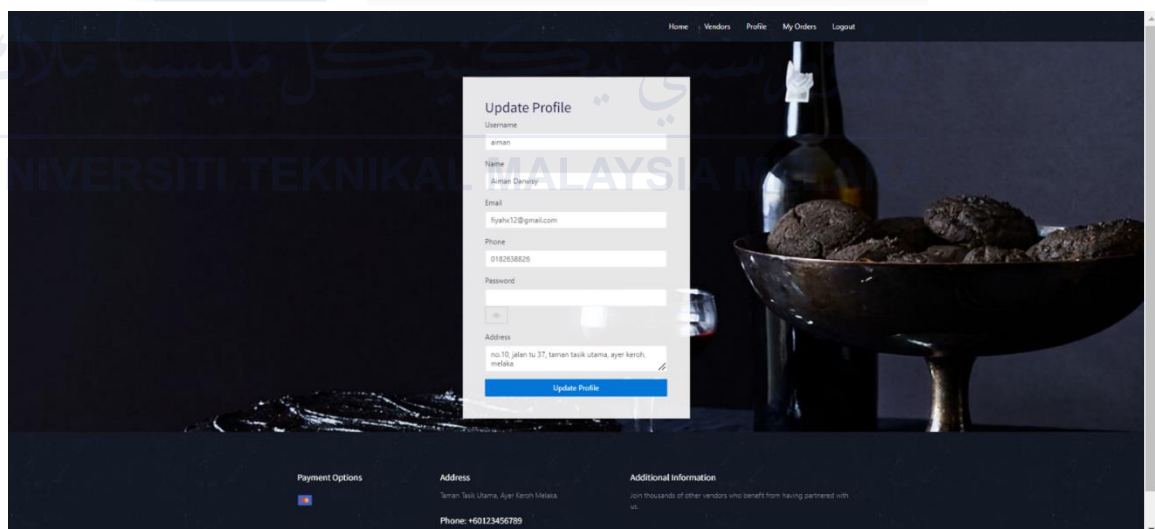


Figure 4.1.1.38: Customer Update Profile Page

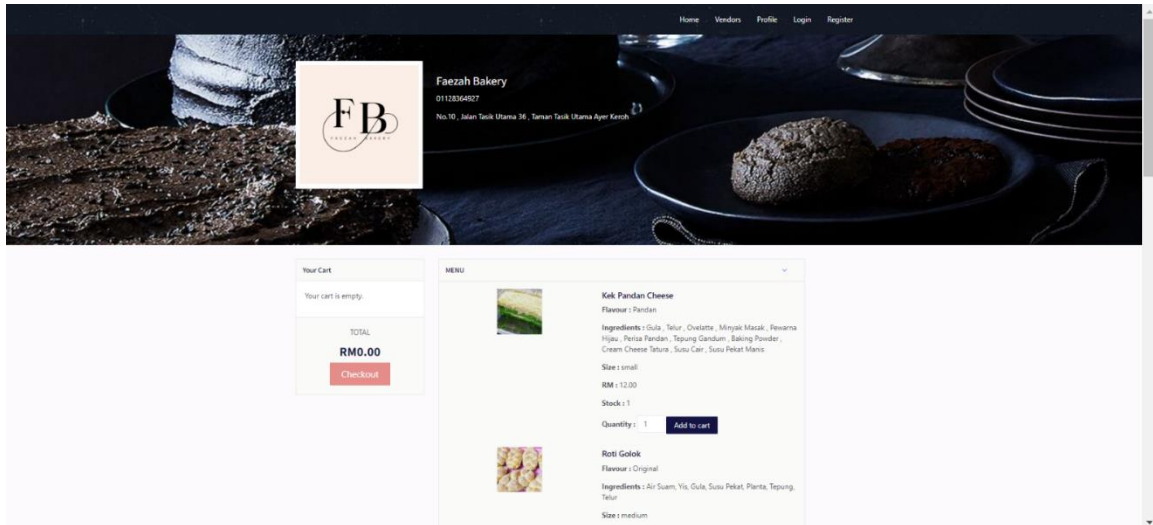


Figure 4.1.1.39: Customer Product by Vendor Page



Figure 4.1.1.40: Customer Product by Category Page

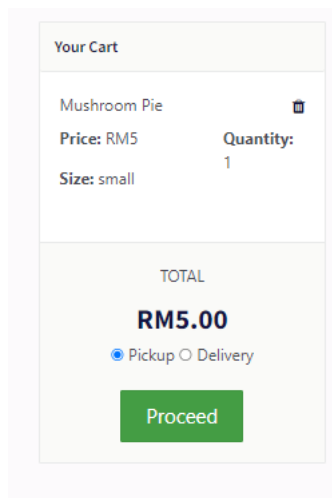


Figure 4.1.1.41: Customer Add to Cart & Delivery Option



Figure 4.1.1.42: List Order Track Page

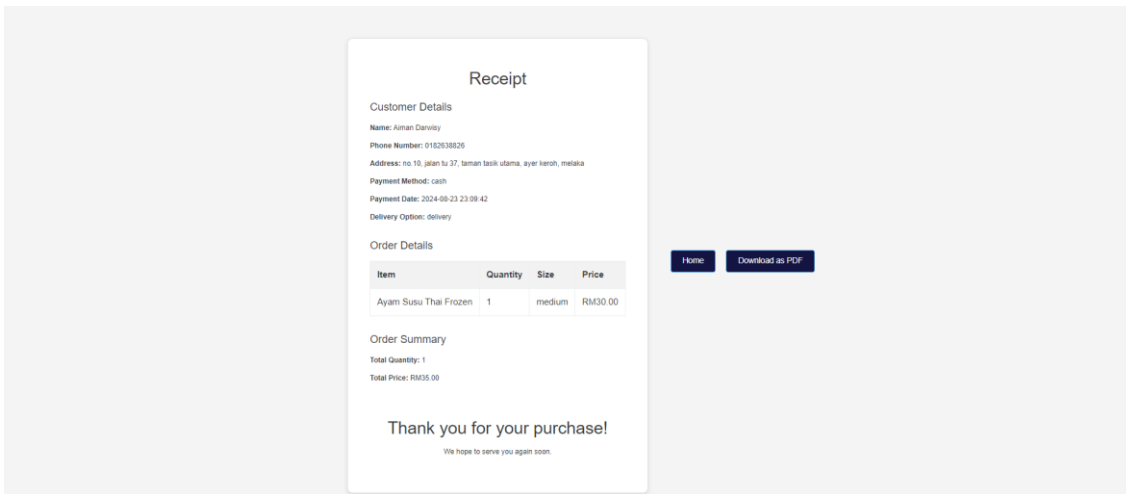


Figure 4.1.1.43: Receipt Page

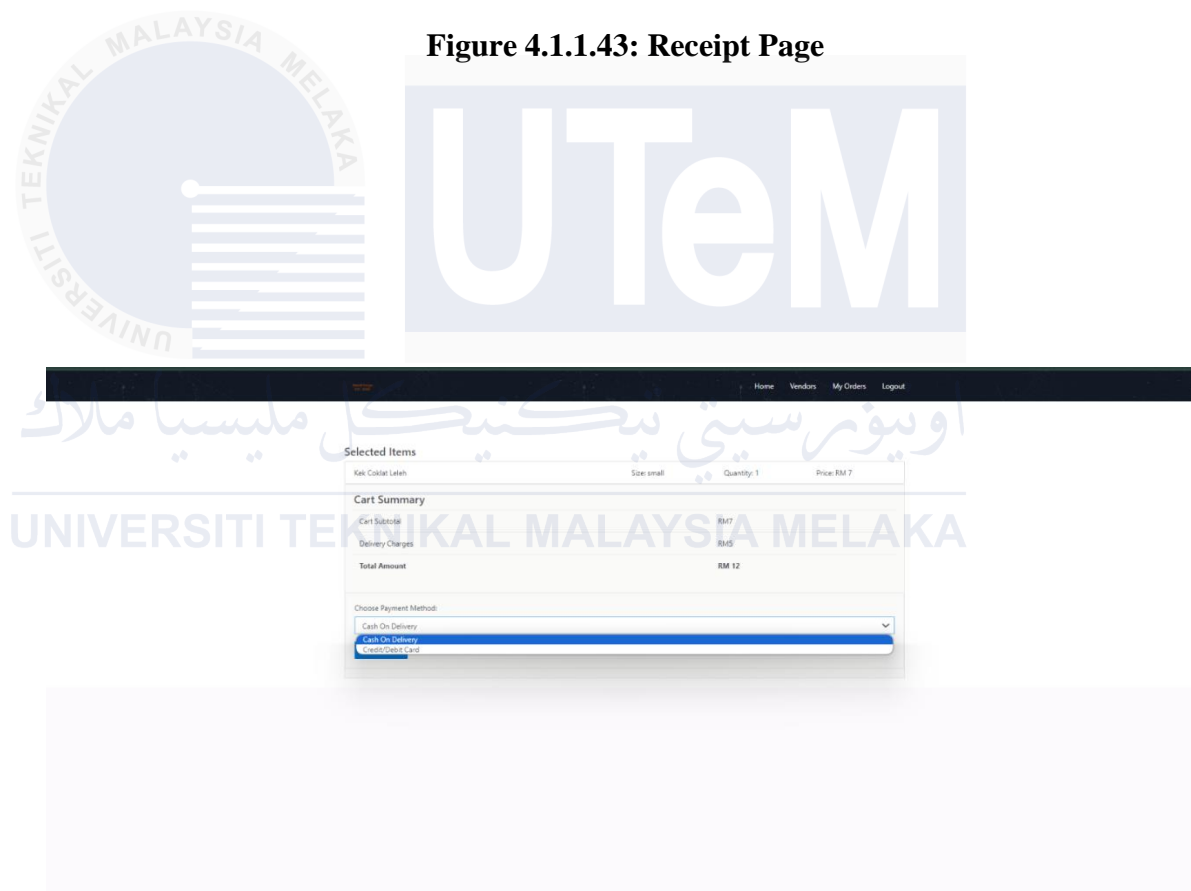


Figure 4.1.1.44 : Checkout Payment Method for Delivery

Bill (Cash Payment)			
Item Name	Price	Quantity	Total
Kek Pandan Cheese	RM 12.00	1	RM 12.00
Total: RM 12.00 Delivery Charge: RM 5.00 Grand Total: RM 17.00			
<input type="button" value="Confirm"/>			

Figure 4.1.1.45: Bill for Cash Payment Delivery Page

The screenshot shows the UTEM online payment interface. On the left, a 'Bill (Card Payment)' table lists 'Kek Cakle Lelah' for RM 7.00. Below the table, the totals are: Total: RM 7.00, Delivery Charge: RM 5.00, and Grand Total: RM 12.00. On the right, a 'Fill in your card details' form includes fields for Account Holder Name, Card Number (masked), Expiry Date (MM/YYYY), and Security Code (CVV). A checkbox for 'I agree to the Private Data Terms and Conditions' is also present. The background features the UTEM logo and the university name in Malay and English.

Figure 4.1.1.46 : Bill for Debit/Credit Card Payment Delivery Page

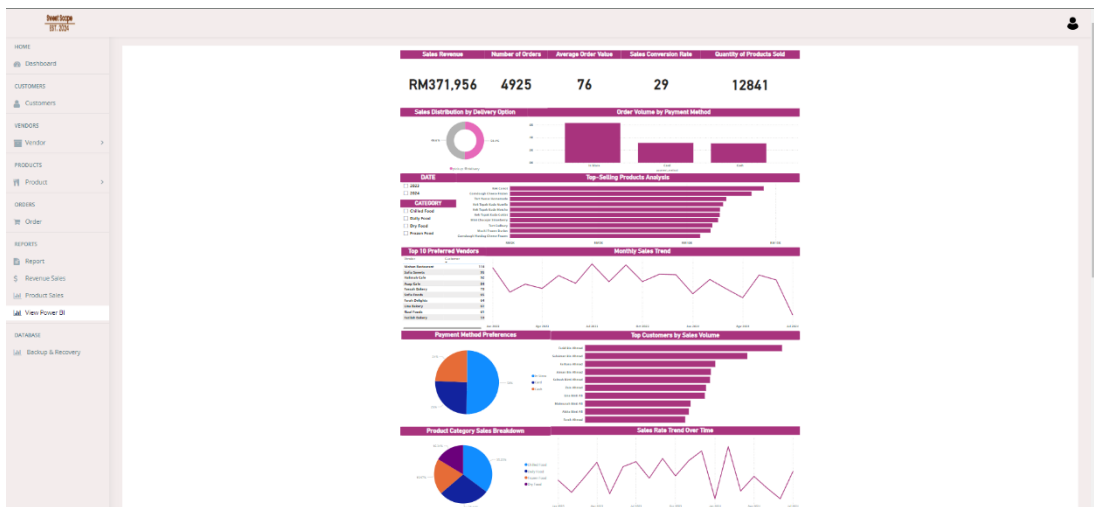


Figure 4.1.1.47 : Insight Sales on PowerBI

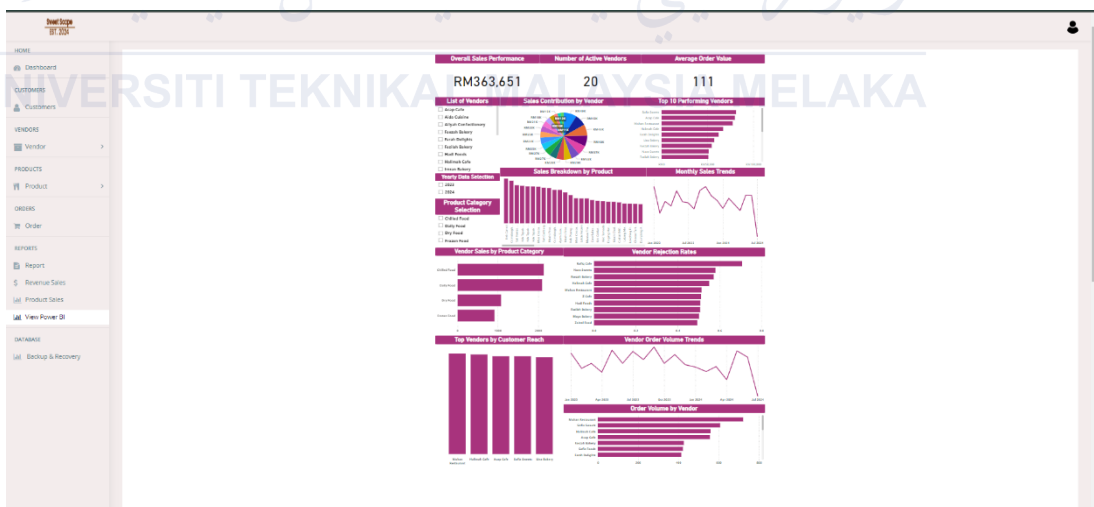
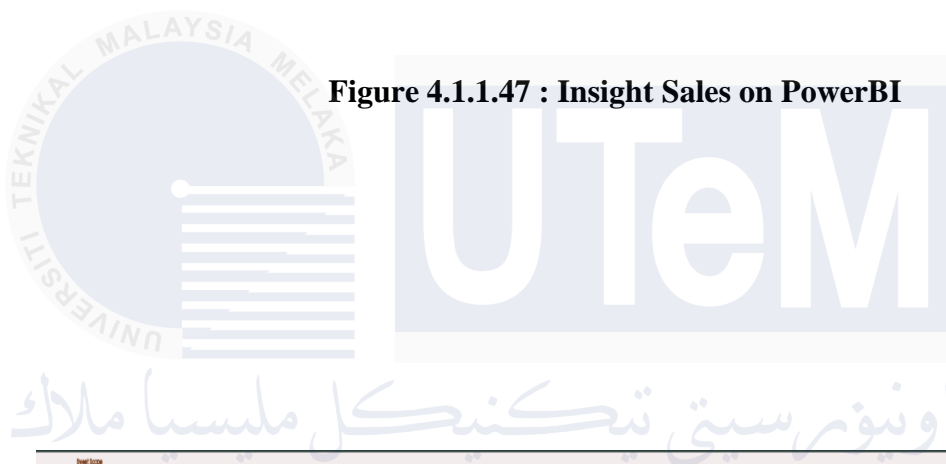


Figure 4.1.1.48 : Insight Vendor on PowerBI

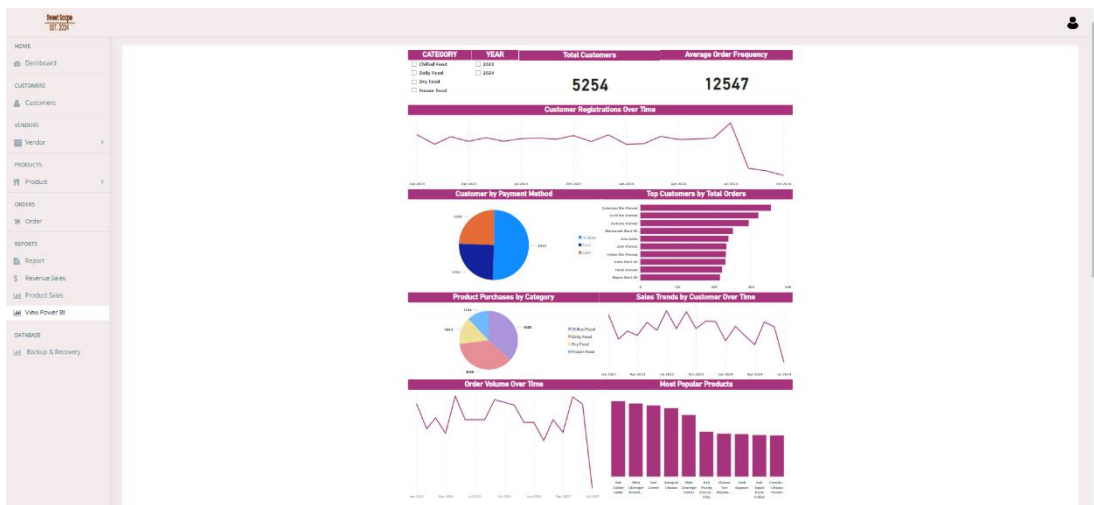


Figure 4.1.1.49 : Insight Customer on PowerBI

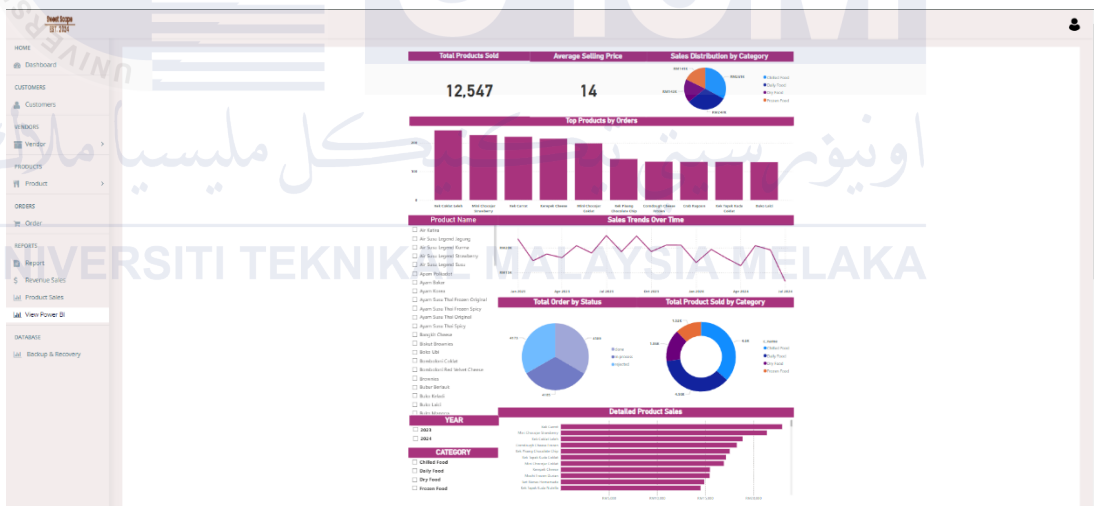


Figure 4.1.1.50 : Insight Product on PowerBI

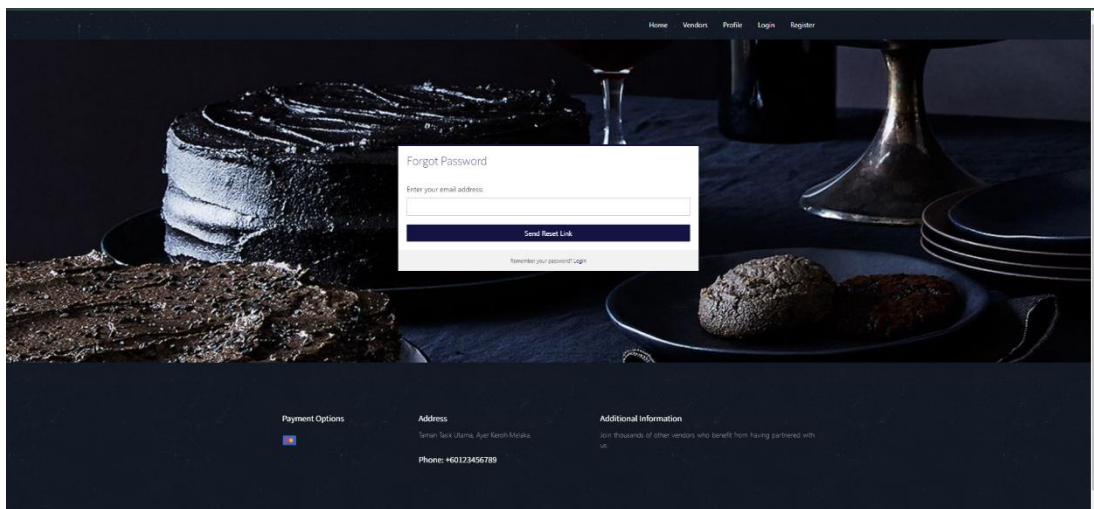


Figure 4.1.1.51 : Forgot Password Customer



Figure 4.1.1.52 : Reset Password Customer

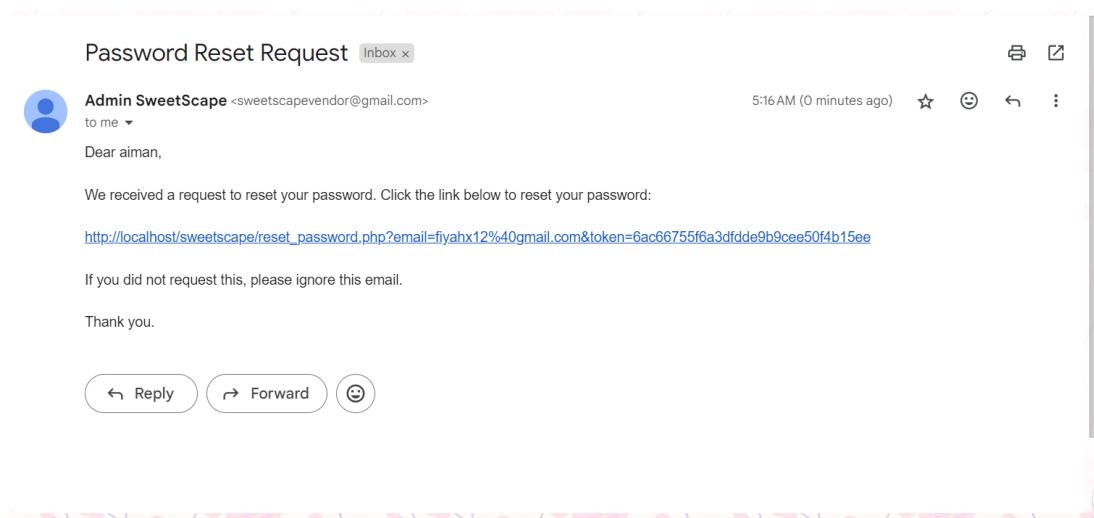


Figure 4.1.1.53 : Reset Password Email

