

StallSync: Intelligence Management System for Food Stalls

By

Yeoh Zhe Heng



**FACULTY OF COMPUTING AND
INFORMATION TECHNOLOGY**

**TUNKU ABDUL RAHMAN UNIVERSITY OF
MANAGEMENT AND TECHNOLOGY
KUALA LUMPUR**

**ACADEMIC YEAR
2025/26**

StallSync: Intelligence Management System for Food Stalls

By

Yeoh Zhe Heng

Supervisor: Ms. Anurehka A/P Magheswaran

A project report submitted to the
Faculty of Computing and Information Technology
in partial fulfillment of the requirement for the
Bachelor of Software Engineering (Honours)

Faculty of Computing and Information Technology
Tunku Abdul Rahman University of Management and Technology
Kuala Lumpur

Copyright by Tunku Abdul Rahman University of Management and Technology.

All rights reserved. No part of this project documentation may be reproduced, stored in retrieval system, or transmitted in any form or by any means without prior permission of Tunku Abdul Rahman University of Management and Technology.

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University rules and regulations and would be subjected to disciplinary actions.



Yeoh Zhe Heng

Bachelor of Software Engineering (Honours).

ID: 23WMR07630

Abstract

This project introduces *StallSync: Intelligent Management System for Food Stalls*, designed to address key challenges in the Food and Beverage (F&B) industry. It resolves issues like a lack of data-driven insights, manual billing errors, and inefficient staff management. The system leverages real-time data analytics to provide actionable insights into sales, product trends, and customer behavior, empowering managers to make informed decisions. Automated digital billing and payment systems reduce errors, support modern payment methods, and streamline promotional management. Staff management tools enable effective scheduling, task assignment, and performance tracking, improving operational efficiency and employee satisfaction. Built using JavaScript frameworks such as Node.js and Express.js, with a modular API-driven architecture, the project follows the Agile development model for iterative testing and improvement. The expected outcome is a robust, user-friendly system that enhances operational efficiency, improves customer satisfaction, and supports sustainable business growth in canteens and food courts.

Acknowledgement

I want to thank everyone who helped with this project. First, I am very thankful to my project supervisor, Mrs. Anurehka A/P Magheswaran. She gave me guidance and feedback. She helped shape the results of this project.

I also thank the lecturers and tutors at Tunku Abdul Rahman University of Management and Technology. They provided a good learning environment. They also gave us the resources we needed to finish the project.

I am thankful to those who participated in the research analysis. Their input was helpful. I also thank my teammate, Loo Wai Kit, for working with me. His hard work and support were important for this project.

Table of Contents

1 Introduction.....	2
1.1 Objective.....	2
1.1.1 To provide data management for records-keeping.....	2
1.1.2 To use the time series modeling for predictive analysis.....	2
1.1.3 To develop StallSync application.....	2
1.2 Project Background.....	3
1.2.1 Problem Statement & Target Market.....	3
1.2.2 Solution.....	5
1.3 Advantages and Contribution.....	7
1.3.1 Advantages.....	7
1.3.2 Contributions.....	7
1.4 Project Plan.....	8
1.4.1 Project Scope.....	8
1.4.2 Project Milestones.....	8
1.5 Project Team & Organization.....	11
1.5.1 Workload Distribution.....	11
1.6 Chapter Summary and Evaluation.....	13
2 Research Background.....	15
2.1 Project Background.....	15
2.2 Review of Existing Systems.....	17
2.2.1 FeedMe.....	17
2.2.2 EasyEat.....	19
2.2.3 ShopeePartner.....	21
2.2.4 Comparison of the Three Existing Systems.....	23
2.3 Literature Review.....	24
2.3.1 Technical model development stack.....	24
2.3.2 Time Series Algorithm Choosing.....	26
2.3.3 Design and Development Tools:.....	29
2.3.4 Deployment.....	29
2.4 Feasibility Studies.....	30
2.3.1 Economic Feasibility.....	30
2.3.2 Legal Feasibility.....	30
2.3.3 Operational Feasibility.....	31
2.3.4 Schedule Feasibility.....	31
2.3.5 Technical Feasibility.....	32
2.5 Chapter Summary and Evaluation.....	33
3 System Analysis.....	35
3.1 Proposed Software Process Model.....	35
3.1.1 Methodology.....	35

3.1.2 System Architecture.....	37
3.1.3 Technical Model Development Stack.....	39
3.1.4 ARIMA Model.....	40
3.2 System Requirements.....	48
3.2.1 Functional Requirements.....	48
3.2.2 Non-Functional Requirements.....	49
3.3 Other Requirements.....	52
3.3.1 Development Environment.....	52
3.3.2 Operational Environment.....	52
3.4 Chapter Summary and Evaluation.....	53
4 System Design.....	55
4.1 Process Design & Algorithm Design.....	55
4.2 Database design & Structural Models.....	57
4.2.1 Business Rules.....	57
4.2.2 Data Dictionary.....	59
4.2.3 ERD Diagram.....	66
4.2.4 Class Diagram.....	67
4.3 Interaction Models (Main Use Case and Modules Use Cases).....	68
4.3.1 Main Use Case.....	68
4.3.2 User Account Module.....	69
4.3.3 Merchant.....	77
4.3.4 Product and Inventory.....	84
4.3.5 Order.....	97
4.3.6 Reward.....	105
4.3.7 Announcement.....	113
4.3.9 Staff.....	119
4.3.10 Report.....	127
4.4 Behavioral Models(State Diagrams).....	133
4.4.1 User Account.....	133
4.4.2 Product.....	133
4.4.3 Order.....	134
4.4.4 Reward.....	134
4.5 Other designs (Architectural design & UI design).....	135
4.5.1 Architecture Design.....	135
4.5.2 Preliminary Interface Design.....	136
4.5.3 Detailed Interface Design.....	140
4.6 Chapter Summary and Evaluation.....	145
5 Implementation and Testing.....	147
5.1 Implementation.....	148
5.1.1 User Authentication (Security).....	148
5.1.2: Success/Error Response.....	150

5.1.3: Header Setup.....	152
5.1.4: Standardized List Function Structure.....	154
5.1.5: Standardized FindOne/View Function Structure.....	156
5.1.6: Standardized Create Function Structure.....	158
5.1.7: Standardized Update Function Structure.....	160
5.1.8: Standardized Delete Function Structure.....	162
5.1.9: Setting for Server.....	164
5.1.10: Network Protocol.....	166
5.1.11: Database.....	167
5.1.12: Forecasting (Machine Learning).....	168
5.2 Testing.....	171
5.2.1 Test Plan.....	171
Overview.....	171
Test Scope and Test Item.....	171
Test Strategy.....	174
Test Data Requirements (Test Strategy).....	174
Test Environment (Test Strategy).....	175
Test Completion Criteria (Test Strategy).....	176
Testing Activities and Estimates - STLC (Test Strategy).....	177
5.2.2 Test Cases.....	178
User Account Module.....	179
Staff Module.....	192
Merchant Module.....	199
Announcement Module.....	206
Product Module.....	212
Inventory Module.....	219
Reward Module.....	225
Member Module.....	231
Order Module.....	233
Transaction Module.....	238
General Type Module (Configuration).....	240
General Code Module (Configuration).....	246
Functions Module (Configuration).....	252
Roles Module (Configuration).....	258
Password Policy Module (Configuration).....	264
System Table Module (Configuration).....	271
System Table Key Module (Configuration).....	277
System Calendar Module (Configuration).....	282
Backup Module (Configuration).....	290
5.3 Chapter Summary and Evaluation.....	293
6 Discussions and Conclusion.....	295
Summary.....	295
Achievements.....	296
Contributions.....	297

Limitations and Future Improvements.....	299
Limitations.....	299
Future Improvements.....	300
.....	300
Issues and Solutions.....	301
References.....	303
Appendices.....	308
Appendix A: Time Series Models' Result and Source Code Link.....	308
Appendix B: Context Diagram.....	311
Appendix C: Portfolio 1 Plagiarism Report.....	313
Appendix D: Github Project Repository.....	314
Appendix E: StallSync Poster.....	315

Chapter 1

Introduction

1 Introduction

This chapter introduces StallSync, an intelligent management system for food stalls. It provides an overview of the system to explain its purpose and the reasons behind this project.

The chapter covers key aspects of the project. It includes the background, the problems it addresses, and the proposed solution. It also discusses the objectives, competitive advantages, and contributions of the system. Additionally, it outlines the project plan, schedule, team structure, and organization.

1.1 Objective

1.1.1 To provide data management for records-keeping

StallSync automates record-keeping. It organizes data on sales, inventory, and more. This removes manual work. Manual work often causes mistakes and delays. The system shows real-time data with clear dashboards. Managers can access this data easily to make decisions.

StallSync secures data and it uses strict access controls. This keeps sensitive information safe. It issues digital receipts to make transactions easier. These features make managing records faster and more reliable.

1.1.2 To use the time series modeling for predictive analysis

StallSync uses time series modeling to study past data. It predicts future sales, customer demand, and staffing needs. This helps managers plan. Accurate forecasts help use resources wisely. It reduces waste and improves efficiency. It also helps stall owners prepare for changes in customer needs.

1.1.3 To develop StallSync application

The StallSync application improves food stall management. It combines tools in one system. It fixes problems like manual record-keeping, slow billing, and poor staff management. It helps stall owners, managers, and administrators work better. StallSync simplifies operations. It has features like real-time sales tracking, order monitoring, and modern payment options. The platform is easy to use. It makes daily tasks simple and improves customer satisfaction.

1.2 Project Background

1.2.1 Problem Statement & Target Market

One big problem in managing food stalls is the lack of data-driven insights. Accurate data helps in making good decisions. [Kristoffersen et al. \(2020\)](#) say decisions are important at every stage of the product life cycle, like development, growth, and decline. Each stage needs quick and accurate choices. Without data, managers guess and often make poor decisions. [Ghasemaghaei \(2019\)](#) shows that businesses using data can solve problems faster and find new ideas. In food services, knowing customer behavior and tracking product trends is key to staying ahead.

StallSync solves these problems by giving real-time sales data and easy digital tools ([Patil et al., 2024](#)). Stall owners often deal with cash transactions and manual records. These tasks cause errors and delays. StallSync offers a dashboard that shows real-time sales and updates product details after each sale. It helps owners track sales and adjust stock and promotions. This reduces errors and improves operations.

Manual billing systems are full of errors like wrong prices or double billing. These mistakes upset customers and cost businesses money. Many customers now prefer using contactless cards, mobile wallets, and QR codes. Traditional cash-only systems fail to meet these needs. [Statista Research Department \(2024\)](#) confirms this trend. [Fuentes \(2024\)](#) also highlights that digital wallets are becoming popular for their convenience and security.

StallSync makes billing and payments easier. It replaces manual billing with digital billing. This speeds up service and reduces mistakes. Customers can pay with their preferred method and get digital receipts. This makes transactions faster and more accurate. Self-service technologies also make ordering easier ([Harnidah et al., 2024](#)).

Tracking discounts and promotions by hand takes time and leads to mistakes. StallSync automates these tasks and ensures discounts are applied correctly. This saves time and avoids pricing errors.

Managing employees with spreadsheets is hard. Supervisors often face staffing problems. They may have too few workers during busy times or too many during quiet hours. Both waste resources and hurt service quality. StallSync offers a tool to manage staff schedules and tasks. Balanced staffing cuts costs and improves service. It also makes both staff and customers happy.

Tracking staff performance without a system is difficult. Supervisors cannot easily monitor progress or give fair reviews. [McKinsey & Company \(2024\)](#) says a people-centered approach improves performance and engagement. StallSync makes performance tracking simple and clear. Managers can give feedback and check progress.

Canteen administrators have to manage multiple stalls. This is hard with manual systems. It slows work and reduces efficiency. StallSync helps by giving tools to track performance and organize tasks. Administrators can use reports to improve service. [Li](#)

[et al. \(2019\)](#) show that good management systems help operations run smoothly. StallSync makes these tasks easier.

IT and technical support teams also face problems with old systems. These systems break down often and take time to fix. StallSync uses modern tools with automatic backups and simple updates. [Zhu et al. \(2016\)](#) explain that system logs help solve problems faster. With StallSync, IT teams can focus on improvements rather than constant repairs.

1.2.2 Solution

StallSync offers a comprehensive solution designed to improve canteen operations by providing real-time insights, streamlining billing and payment processes, and enhancing staff management. One of its core features is the ability to collect and analyze real-time data on sales, products, and customer preferences. This data gives managers and stall owners a clear picture of business performance. The system presents this information on detailed dashboards and reports, making it easier to spot patterns in sales, identify peak hours, and track product demand. With this data at their fingertips, managers can make better decisions, such as adjusting stock levels to avoid shortages or reducing excess inventory to minimize waste. Knowing customer preferences allows canteen operators to optimize their menus, offering the most popular items while eliminating less-demanded products.

For example, identifying the busiest times of the day enables staff to prepare in advance, ensuring faster service and shorter waiting times. Product trends help business owners anticipate customer demand and avoid overstocking, which reduces spoilage and unnecessary costs. With accurate data insights, operations can be more responsive and efficient, ultimately improving the customer experience while reducing operational risks.

In addition to data-driven insights, StallSync introduces a fully digital billing and payment system that replaces outdated paper-based processes. Traditional manual billing often results in errors like double billing, incorrect change, and missed transactions. StallSync eliminates these problems by digitizing the entire payment process. The system uses machine learning to predict sales and purchasing trends, helping business owners better plan their budgets and inventory. It also supports modern payment methods, including contactless card payments, which make transactions faster and more convenient for customers.

The system automates the tracking and application of discounts and promotional offers, reducing the chances of pricing errors. This feature ensures that customers always receive the correct discounts, which improves customer satisfaction and builds trust. Digital receipts provide an easy way to track and review transactions, making it simpler to resolve disputes or answer customer inquiries. Overall, StallSync's advanced payment handling reduces delays, increases accuracy, and saves both time and money.

Another important feature of StallSync is its ability to help manage staff more effectively. Staffing issues, such as being short on staff during busy hours or having too many employees during quieter periods, can disrupt operations and increase costs. The system's admin dashboard provides real-time insights into staffing needs, helping managers allocate resources more efficiently. By avoiding understaffing, businesses can maintain high service quality even during peak hours. At the same time, preventing overstaffing helps reduce unnecessary labor costs.

Balanced staffing not only improves service quality but also boosts employee satisfaction. Staff members are less likely to feel overwhelmed during busy times and can perform their tasks more efficiently. Customers, in turn, benefit from faster service and a better experience. By integrating data insights, payment automation, and

staff management tools, StallSync creates a well-rounded solution that improves business performance at every level.

1.3 Advantages and Contribution

1.3.1 Advantages

1. Comprehensive Integration:

- Many systems focus on one part of the business, like billing or product. StallSync combines all these features in one system. It tracks sales, manages products, handles employee schedules, processes payments, and monitors performance. This makes it easier to manage everything in one place.

2. Data-Driven Insights:

- Existing systems often require manual tracking. The StallSync gives real-time data. It provides reports on sales, product, and customer behavior. This helps managers make better choices and avoid mistakes.

3. Modern Payment Methods:

- Traditional systems mostly accept cash or basic card payments. StallSync supports contactless payments, QR codes, and mobile wallets like Apple Pay and GrabPay. This allows customers to pay faster and more securely.

4. User-Friendly Interface:

- StallSync is simple to use. It has an easy design. This makes it possible for staff with little technical experience to learn the system quickly.

1.3.2 Contributions

1. Stall Owners/Managers:

- The StallSync helps stall owners save time. It reduces errors and simplifies tasks like products and payment processing. It also helps managers make smarter decisions with data. This makes the business run more smoothly and increases profits.

2. Customers:

- Customers get faster service with accurate billing. They can pay with mobile wallets and cards. This improves their overall experience and satisfaction.

3. Administrators:

- The StallSync gives operators a clear view of all stalls. It helps them track performance, manage staff, and keep operations running smoothly. This ensures consistent service across the food court.

4. Community Impact:

- StallSync helps small food businesses grow. It offers simple and affordable technology that boosts service quality. This leads to better competition in the community.

1.4 Project Plan

1.4.1 Project Scope

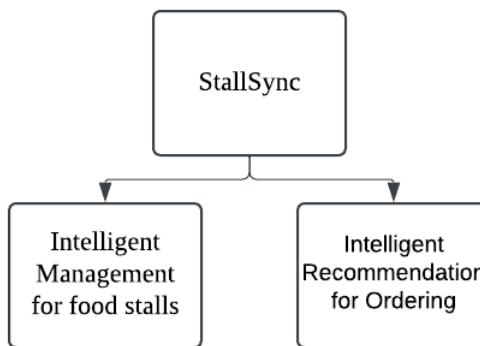


Figure 1.1: System Structure

1.4.2 Project Milestones

The proposed milestones are as follows:

Table 1.1: Project Milestones

Milestone	Milestone Goal	Deadline (DD/MM/YYYY)
Concept Approval	Proposed system concept are approved	2/12/2024
Proposal Submission	Submission of proposal (Form1, Form2 & Form3)	3/1/2025
Introduction	Complete the introduction precisely and clearly that explaining the project is approved	21/2/2025
Research Background	Submission of research background	6/3/2025
Methodology and Requirement Analysis	Submission of methodology and requirements analysis	28/3/2025
System Design	Comprehensive system design submission	11/4/2025
Project 1 Portfolio	Submission of Project 1 portfolio	20/4/2025
System Development	Implementation and development of the proposed system	2/7/2025

Test plan / cases	Preparation of test plan/cases	11/7/2025
System Preview	System preview with supervisor	15/7/2025
Final System Testing	Final system testing with supervisor and moderator	28/7/2025
Draft report	Submission of draft FYP report	25/8/2024
Final Report	Submission of final FYP report and all associated deliverables	4/9/2025

The gantt chart of the project is as follow:

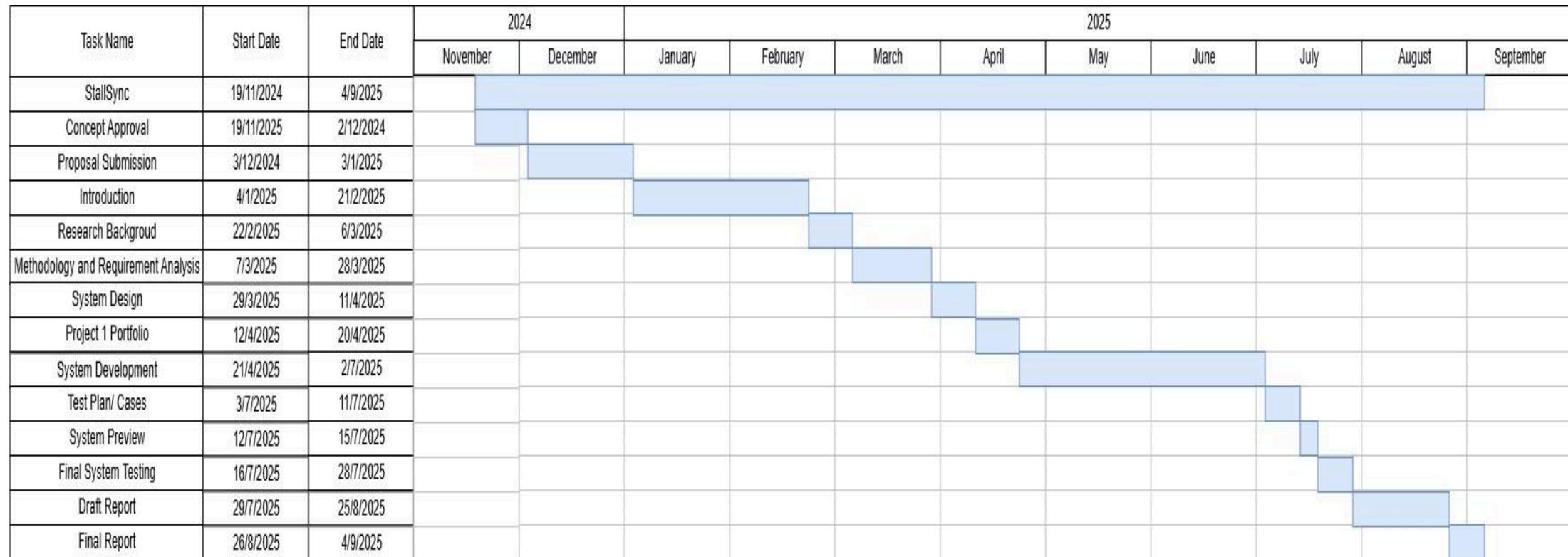


Figure 1.2: Gantt Chart

1.5 Project Team & Organization

1.5.1 Workload Distribution

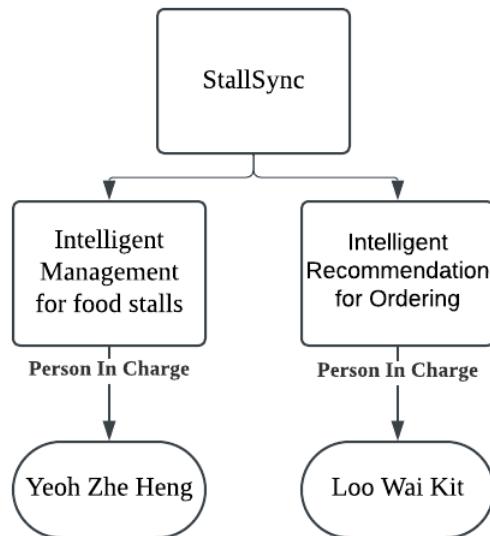


Figure 1.3: Project Team Structure

The system consists of StallSync: Intelligent Management food stalls and StallSync: Intelligent Recommendation for Ordering, the first is targeting the stalls owners, staff and administrator and the second is targeting the customers respectively. Both applications will be developed by both of us respectively and integrated together to form a complete system. The StallSync: Intelligent Management food stalls application will be developed by Yeoh Zhe Heng, while the StallSync: Intelligent Recommendation for Ordering will be developed by Loo Wai Kit.

Table 1.2: Modules' functions in charged by the team members

Modules	Yeoh Zhe Heng	Loo Wai Kit
User Account	✓	
Customer		✓
Admin Dashboard	✓	
Member HomePage		✓
Merchant	✓	
Cart		✓
Product	✓	

Transaction		✓
Order	✓	
Reward	✓	
Review		✓
Announcement	✓	
Inventory	✓	
Staff	✓	
Report	✓	

1.6 Chapter Summary and Evaluation

This chapter introduces StallSync, an intelligent system designed to manage food stalls. It explains the purpose of the system and the key problems it aims to solve. The chapter also outlines the solution provided by the system and identifies the target users it is meant to serve.

The chapter highlights the main goals of StallSync. These goals define the purpose of the system and its intended impact. It describes how the system helps improve the management of food stalls. It also shows how StallSync fits into the current market and benefits both stall owners and customers.

Planning and organizing tasks are also discussed in this chapter. It explains how work is divided among team members. This ensures the project stays on track and meets deadlines. It also guarantees that the quality of the work remains high.

Chapter 2

Literature Review

2 Research Background

Chapter 2 will have several sections. These sections will talk about the project's background. They will also look at current software systems or apps with similar features. The chapter will check if the proposed system is feasible from different points of view.

2.1 Project Background

The food and beverage (F&B) industry has transformed significantly since the pandemic, with many institutions, including school canteens, adopting digital solutions to enhance efficiency. Online food ordering and virtual queuing systems have become essential in reducing wait times, managing food demand, and preventing overcrowding during peak hours. These technologies also help minimize food waste by optimizing inventory and meal preparation. The cloud kitchen industry, valued at \$43.1 billion in 2019, experienced rapid growth during the COVID-19 pandemic ([Research and Markets, 2020](#)). Beyond school canteens, digital food management solutions are also widely used by major restaurant chains and high-end establishments. Brands such as Starbucks ([Akcam, 2022](#)) have invested in advanced ordering systems, allocating dedicated resources to streamline digital transactions and improve customer experiences.

School canteens are also moving toward healthier meal options. Many are replacing processed foods with fresh, organic ingredients to promote student health and enhance concentration in class. However, rising costs and staff shortages make it difficult to maintain these standards. To address these challenges, some canteens are considering privatization or seeking government subsidies to remain operational. Despite advancements in the industry, many school canteens still rely on outdated methods, such as paper-based ordering, cash payments, and manual record-keeping. These traditional processes lead to frequent errors, lost records, and long queues, frustrating both students and staff. A study by [Simuka et al. \(2024\)](#) found that manual systems in canteens contribute to inefficiencies, resulting in extended waiting times and poor customer satisfaction.

StallSync aims to solve these issues by introducing a digital ordering and payment system ([Patil et al., 2024](#)). It provides a user-friendly platform where students can browse menus, place orders, and make payments online. This eliminates the need for multiple stall-specific apps and simplifies the ordering process. By streamlining operations, StallSync helps canteen staff manage orders more efficiently, reducing workload and errors ([Harnidah et al., 2024](#)).

The goal is to build a simple online platform for students and staff to order food. Users can place orders and pay online using a secure system. The orders will go straight to the kitchen. This will reduce mistakes and save time. Waiting times will be

shorter, and the canteen will run more smoothly. The main users are students and staff. They will get faster service and convenience. Canteen operators will have fewer errors and easier management. Schools will also benefit by offering better canteen services. The developers will design and maintain the platform ([Pandey et al., 2024](#)).

There are some steps that need to be done before StallSync can start. First, it is important to collect required information which is helping to define what the system needs. Next, is to create a design for the system to make sure it works well and is secure. Testing the system is also necessary because it helps find and fix problems.

StallSync is expected to deliver clear results. The main outcome is an online food ordering platform. It will have a payment gateway and real-time order management. Users will have an easy way to order food. Canteen operators will get better tools to manage orders and finances. There will be fewer errors, and operations will run better. The system will help improve the overall experience for both users and staff.

2.2 Review of Existing Systems

2.2.1 FeedMe



Figure 2.1 : FeedMe

Source:

<<https://www.google.com/url?sa=i&url=https%3A%2F%2Fmipos.my%2Fpos-system-feedme%2F&psig=AQovVaw1bUw7fwv5LCA0s9NOOg8Dh&ust=1741444294539000&source=images&cd=vfe&opi=89978449&ved=0CBYQjRxqFwoTCNiWdiX-IsDFOAAAAAdAAAAA>
BAE>

The first system to review is FeedMe. FeedMe is a Point of Sale (POS) system for restaurants and food service businesses. It helps simplify operations with features like order management, table reservations, and inventory tracking. It also has tools for customer relationship management (CRM). FeedMe is cloud-based. This allows real-time updates and remote access. It is especially useful for businesses with multiple locations. It works with third-party platforms like delivery services and accounting software. This makes it more flexible. FeedMe helps businesses handle food orders better by improving operations and lowering costs ([FeedMe, 2018](#)).

FeedMe's interface is easy to use. The design helps staff learn and use the system quickly. FeedMe supports many payment methods, including cash, credit/debit cards, and mobile payments ([Aditya & Suhirman, 2024](#)). This is good for both merchants and customers. The system also gives detailed analytics and reports. These help businesses track sales, monitor inventory, and make better decisions ([Uriawan et al., 2024](#)).

FeedMe has some drawbacks that cause disrupted operations. The system needs the internet to work. If the internet has problems, the system may not work well. FeedMe has many features, but it may not have some advanced tools found in more expensive POS systems. This could be a problem for larger or more complex businesses ([Sachidananda Kangovi, 2017](#)).

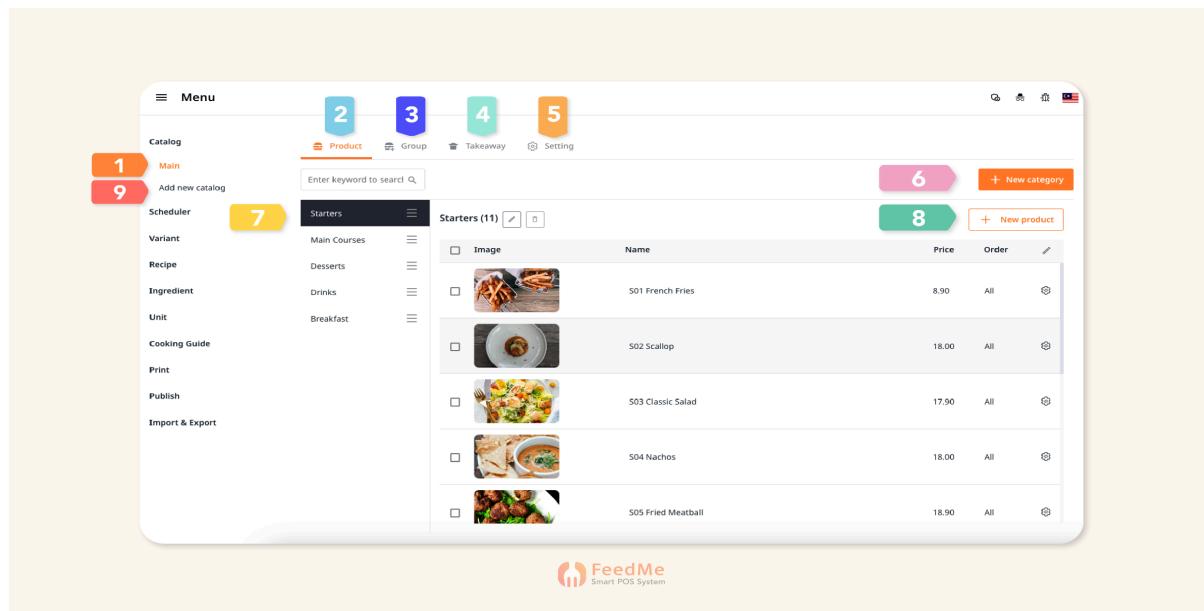


Figure 2.2: Example FeedMe Merchant Side Interface

FeedMe also enables F&B businesses to keep their menu up-to-date and ensure that their customers always have access to the latest offerings. By allowing businesses to update their menu anytime, FeedMe ensures that their customers are always presented with the most current menu options ([Maier et al., 2024](#)).

Table 2.1: Pros and Cons of FeedMe

Pros	Cons
<ul style="list-style-type: none"> • User no need download application • Real-time order tracking • Easy Menu Updates • Good range of payment methods: TouchNGo, Cash, Cards and more • Easy-to-use interface • Integration with online delivery service • Good for multi-location businesses • Gives detailed analytics and reports • Offers demo system 	<ul style="list-style-type: none"> • Learning curve for new user • Not working to order when offline • Needs stable internet connection • Few advanced features for larger businesses • Limited customization options • Can be expensive for small businesses

2.2.2 EasyEat

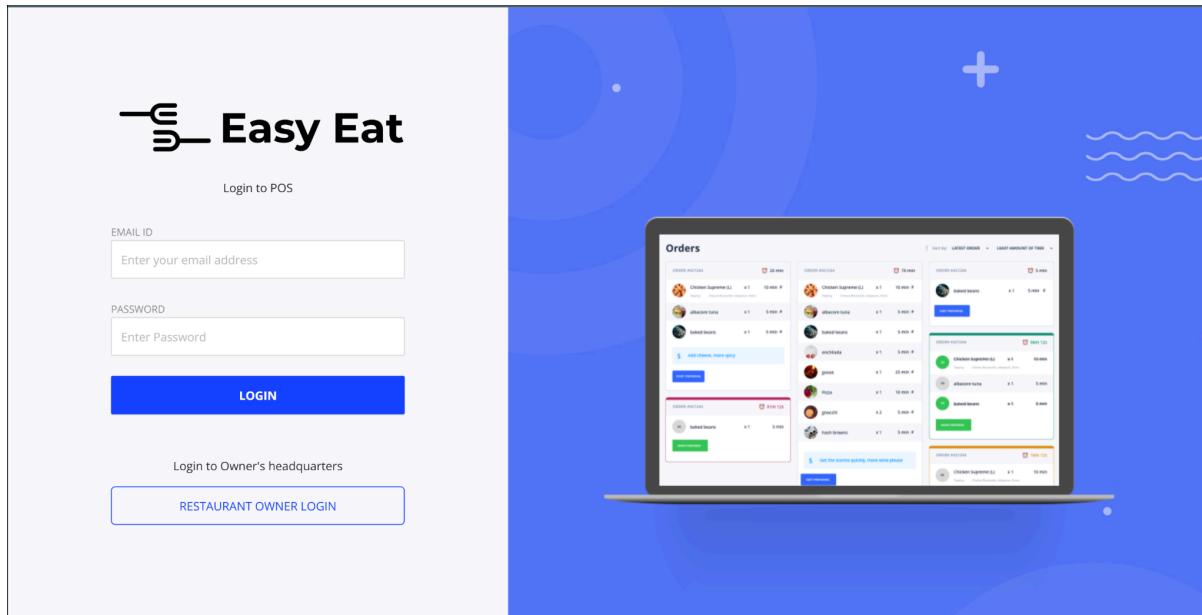


Figure 2.3: EasyEat Partner Portal

The second system to be introduced is EasyEat Partner. EasyEat is a Point-of-Sale (POS) system designed for food and beverage (F&B) businesses. It serves two main user groups: merchants and customers. This review focuses on the merchant side (EasyEat Partner), though some customer-related details are also mentioned. The system provides features such as multiple outlet management, QR ordering, a loyalty program, inventory and stock management, and real-time synchronization ([EasyEat, 2024](#)). These features help F&B businesses simplify operations and improve customer service.

Merchants can manage multiple outlets from a single platform. The QR ordering system allows customers to browse menus, order food, and make payments directly from their smartphones. This reduces waiting time and improves service efficiency. EasyEat also supports real-time inventory tracking, helping business owners monitor stock levels and avoid shortages. The loyalty program feature keeps track of customer preferences and rewards regular customers, encouraging repeat visits ([Adilla Faradila & Ichsan, 2025](#)).

The system also provides essential hardware products based on the nature of the business, such as iPads and printers. This flexibility allows businesses to customize and build their own POS setup. F&B establishments can easily reach out to EasyEat through a contact form on their website. They can also take advantage of a 30-day free trial and other benefits to test the system before committing ([Asrani et al., 2024](#)).

EasyEat's payment processing system supports various payment methods, including contactless payments, credit cards, and digital wallets. This reduces cash-handling errors and makes transactions faster and more secure. Real-time synchronization ensures that all data across multiple outlets is up to date. This allows business owners to access sales data, monitor performance, and make better decisions ([Google Patents, 2001](#)).

However, there are some challenges with the EasyEat Partner POS system. It requires a stable internet connection for optimal performance. Poor connectivity may cause delays. Setting up the system and training staff can take time, which may be difficult for small businesses with limited resources. Additionally, transaction fees for digital payments can add to operational costs ([Khando et al., 2022](#)).

Table 2.2: Pros and Cons of EasyEat

Pros	Cons
<ul style="list-style-type: none">• Free POS devices• Loyalty program for customer retention• QR ordering and multiple outlet management• Real-time inventory tracking• Offers 30-day free trial• Supports many payment options• Customizable hardware options	<ul style="list-style-type: none">• Needs a reliable internet connection• Setup and training take time• Transaction fees can increase costs• Less ideal for small vendors

2.2.3 ShopeePartner



Figure 2.4: ShopeePartner

Source: <<https://play.google.com/store/apps/details?id=com.shopeepay.merchant.my>>

The first system to be examined is ShopeePartner, a web and mobile application designed for the merchant side of ShopeeFood. ShopeeFood serves three key user groups: buyers, merchants, and delivery drivers. This review focuses mainly on ShopeePartner but also touches on some buyer-related concerns. ShopeePartner allows F&B businesses to manage their ShopeeFood delivery operations with ease, giving them more time to focus on preparing delicious meals for their customers ([ShopeeFood, n.d.](#)). It also provides merchants with an opportunity to promote their business and attract new customers.

A study conducted by [Fitri Yutika \(2023\)](#) on the intention of coffee shop entrepreneurs to use the ShopeePartner app found that Effort Expectancy and Facilitating Conditions are major factors influencing their decision to adopt the platform. These elements significantly shape their willingness to utilize the ShopeePartner application. Additionally, [Christine & Berlianto \(2022\)](#) highlights that Performance Expectancy has a positive effect on repurchase intention at ShopeeFood (t -statistic $2.315 > 1.65$), suggesting that users are more likely to reuse the platform based on its perceived benefits.

ShopeeFoodPartner is currently available only in major cities like Klang Valley, Johor Bahru, Penang, and Sabah. F&B businesses interested in becoming ShopeeFood partners must first submit an application form. Once approved, they receive an invitation to set up an account. Business owners can then register on the ShopeePartner web or mobile app and start managing their orders and operations. ShopeePartner promotes several useful features, including Real-Time Transaction Updates, Flexible Store Management, Business Hours Settings, and Easy Menu Management, which help merchants optimize their business processes ([Fini Anjela et al., 2024](#)).

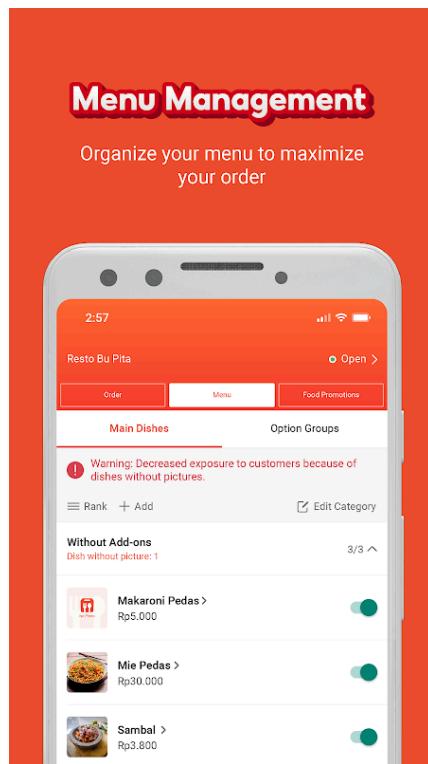


Figure 2.5: ShopeePartner Menu Management

ShopeePartner offers menu management based on merchant side preferences to provide the real-time updates and record tracking. These features increase the efficiency of menu changing compared to traditional ([Google Patents, 2013](#)).

Table 2.3: Pros and cons of ShopeePartner

Pros	Cons
<ul style="list-style-type: none"> • Wide Customer Base • Real-time transaction update • Flexible Store management • Easy Menu Management • Business Growth Support • Easy-to-use interface 	<ul style="list-style-type: none"> • Limited to Major Cities • High Commission Fees • Account Approval Process • Technical issues • Order are not allowed to cancel by F&B business • Intense Competition

2.2.4 Comparison of the Three Existing Systems

Table 2.4: Comparison of the three existing systems

Aspect	FeedMe	EasyEat	ShopeeFood
Core Services	In-premise order management	In-premise order management	Online food delivery
Fee type	Subscription fee	Subscription fee	Transaction fee
Cost Comparison	High	Lower compared to FeedMe	low
Menu management	Yes	Yes	Yes
Order Management	Yes	Yes	Yes
Management Devices	Required to buy	Free Ipad and printer only	Required to buy
Employee Management	Yes	Yes	No
Platform for businesses	POS system / Web	POS system / Web	Web/Mobile App
Small business friendly	No	No	Yes
Learning Curve	High	High	Lower
Payment Gateway Provided	Yes, all contactless payment	Yes, all contactless payment	ShopeePay (No charges)
Loyalty Program	Stages	Stages and cashback	Voucher and cashback
Data and Analytic	Yes, but for premium package and above	Yes	Yes

2.3 Literature Review

2.3.1 Technical model development stack

A key element in developing web applications is selecting the right technology stack. According to [Samuel \(2023\)](#), several technology stacks can improve the development process. Some common examples will be shown through Table 2.5:

Table 2.5: Technology stack

Stack Name	Components
MERN	<ul style="list-style-type: none"> - MongoDB (NoSQL, Database) - Express (Node.js web framework) - React (Frontend) - Node.js (Backend)
PERN	<ul style="list-style-type: none"> - PostgreSQL (SQL, Database) - Express (Node.js web framework) - React (Frontend) - Node.js (Backend)
MEAN	<ul style="list-style-type: none"> - MongoDB (NoSQL, Database) - Express (Node.js web framework) - Angular (Frontend) - Node.js (Backend)
LAMP	<ul style="list-style-type: none"> - Linux (Operating System) - Apache (Web Server) - MySQL (SQL, Database) - PHP (Logic Implementation)
MEVN	<ul style="list-style-type: none"> - MongoDB (NoSQL, Database) - Express.js (Node.js web framework) - Vue.js (Frontend) - Node.js (Backend)

These stacks serve as a guideline, not a strict rule for developers. The components in these stacks can be replaced with other alternatives if they are more suitable for the project. The choice of stack mainly depends on the developer's familiarity with specific programming languages, experience, and availability of relevant resources.

There is the comparison of the relational database having distinct advantages over Non-relational database techstack like MERN. According to [AWS \(2023\)](#), relational databases are better suited for data warehousing and online analytical processing, making them ideal for handling complex transactions commonly found in e-commerce, trading, and financial applications. One of its key strengths is its compliance with the ACID (Atomicity, Consistency, Isolation, and Durability) standard, ensuring that each transaction is processed

reliably and securely. This makes relational databases a robust choice for applications that require high data integrity and transactional accuracy. The comparison is summarized and shown through Table 2.6.

Table 2.6: Comparison of Relational Database and Non-Relational Database

Feature	Relational Database	Non-Relational Database
Data Structure	Structured, follows a tabular format with rows and columns	Unstructured or semi-structured, stores data as JSON-like documents
Best Use Case	Suitable for data warehousing, analytical processing, and complex transactions (e.g., e-commerce, trading, finance)	Ideal for flexible, schema-less storage, real-time applications, and large-scale distributed systems
Transactions	Supports ACID compliance, ensuring data integrity and secure transactions	Generally follows BASE (Basically Available, Soft state, Eventually consistent), prioritizing scalability over strict consistency
Scalability	Vertical scaling (increasing resources of a single server)	Horizontal scaling (distributing data across multiple servers)
Query Language	Uses SQL (Structured Query Language)	Uses NoSQL query languages (e.g., MongoDB Query Language)
Performance	Optimized for structured queries and complex joins	Faster for handling large, unstructured datasets and high read/write operations

2.3.2 Time Series Algorithm Choosing

In this project, Artificial Intelligence (AI) will be used to perform predictive analysis on certain reports. This is aimed at preventing the misallocation of employees at the wrong times and allowing the owner to prepare an adequate stock of raw materials based on predicted demand. Machine learning algorithms, specifically time series algorithms, will be employed for predictive analysis. Python will be used to develop the algorithm due to its simplicity and the powerful libraries it offers ([Türkmen et al., 2024](#)). According to [InfluxData \(2024\)](#), the proposed system's predictive analysis will focus on sales and quantity data, which exhibit trends, seasonal patterns, cyclic patterns, and regularities.

Various time series models are available for forecasting purposes. One widely used model is the Auto-Regressive Integrated Moving Average (ARIMA), which predicts future values by combining past data points and previous forecast errors. An extension of this model, the Seasonal ARIMA (SARIMA), incorporates seasonal patterns by including seasonal past values and forecast errors, making it more suitable for data with recurring seasonal trends.

To simplify the process of selecting optimal parameters, the Auto-ARIMA model automates the identification of key attributes such as the order of differencing, autoregressive (AR) terms, and moving average (MA) terms. It systematically explores different parameter combinations to find the best fit for the given time series data. Similarly, the Auto-SARIMA model applies this automated approach to seasonal time series data. Additionally, a more flexible version known as Custom Auto-ARIMA allows users to define a broader range for parameters (p, q, d), enabling a more tailored search for the best configuration.

Beyond ARIMA-based models, machine learning approaches like LightGBM (Light Gradient-Boosting Machine) and XGBoost (Extreme Gradient Boosting) offer powerful alternatives. LightGBM is an open-source framework built on decision tree algorithms, designed for high performance and scalability in tasks such as classification, ranking, and regression. On the other hand, XGBoost is a distributed gradient-boosted decision tree (GBDT) library known for its speed and accuracy. It supports parallel tree boosting and is widely used for solving complex regression, classification, and ranking problems.

All models will be evaluated through MAE (Mean Absolute Error). The importance of MAE lies in its ability to measure the forecasting accuracy in time series analysis, the lower MAE is indicating the better model ([Wikipedia Contributors, 2019](#)).

The advantages and disadvantages of the models mentioned above are listed below through Table 2.7.

Table 2.7: Advantages and Disadvantages of each models

Models	Advantages	Disadvantages	References
ARIMA	- Effective for	- Assumes linear	(GeeksforGeeks,

	<ul style="list-style-type: none"> - non-seasonal time-series forecasting - Captures trends and patterns in stationary data - Simple to interpret and implement 	<ul style="list-style-type: none"> - relationships - Not suitable for seasonal data - Requires stationarity, meaning data preprocessing is needed 	2024)
SARIMA	<ul style="list-style-type: none"> - Extends ARIMA by handling seasonality - Suitable for data with repeating patterns - Can model both trend and seasonality 	<ul style="list-style-type: none"> - Complex parameter tuning (p, d, q, P, D, Q) - Computationally expensive for large datasets - Requires domain expertise for effective parameter selection 	
Auto-ARIMA	<ul style="list-style-type: none"> - Automatically selects optimal ARIMA parameters (p, d, q) - Reduces manual effort in model tuning - Useful for data with unknown structure 	<ul style="list-style-type: none"> - Can be slow for large datasets - Sometimes selects suboptimal parameters - May not always generalize well to unseen data 	(Markos, 2020) (Prasad, 2021)
Auto-SARIMA	<ul style="list-style-type: none"> - Automates parameter selection for seasonal data - Reduces human intervention in choosing SARIMA parameters - Handles both trend and seasonality effectively 	<ul style="list-style-type: none"> - Computationally intensive - Risk of overfitting if not tuned properly - Still requires validation to ensure best model selection 	
Custom-Auto-ARIMA	<ul style="list-style-type: none"> - Allows flexible tuning for ARIMA model selection - Gives control over the search space for parameters - More adaptable to specific dataset requirements 	<ul style="list-style-type: none"> - Requires manual tuning of the search range - Can be computationally expensive for large datasets - May still require adjustments for best performance 	
LightGBM	<ul style="list-style-type: none"> - Highly efficient and scalable 	<ul style="list-style-type: none"> - Requires careful hyperparameter 	(Khadka, 2023)

	<ul style="list-style-type: none"> - Handles large datasets with high speed - Reduces memory usage due to leaf-wise tree growth 	<ul style="list-style-type: none"> - tuning - Can be sensitive to noise in small datasets - Not ideal for purely time-series data without feature engineering 	
XGBoost	<ul style="list-style-type: none"> - Powerful ensemble method for time-series forecasting - Strong predictive performance and generalization - Works well with missing values and non-stationary data 	<ul style="list-style-type: none"> - Computationally intensive - Prone to overfitting if not tuned properly - Requires feature engineering to perform optimally 	<u>(XGBoost Advantages and Disadvantages (Pros vs Cons) XGBoosting, 2024)</u>

2.3.3 Design and Development Tools:

For this project, Visual Studio Code (VSC) may be used as the primary source code editor. VSC provides a wide variety of extensions and plugins that support the various technology stack outlined and enhance the development process. It is an open-source tool, licensed under MIT, and is both lightweight and powerful, making it a preferred choice for many developers ([Sufyan bin Uzayr, 2022](#)).

Figma has the potential to be chosen for UI design and prototyping due to its real-time collaboration features and availability of a free version, which makes it suitable for teamwork and fast design iterations. For creating system design and documentation diagrams, Lucidchart will be the primary tool. Lucidchart is user-friendly and provides the necessary features for building clear and organized visual representations. Other than Lucidchart, there will be use of Draw.io also as backup use.

While these tools have the potential to be selected, the project team remains open to adopting other design and development tools that may better suit their needs as the project evolves. Additionally, Git will be utilized as the version control system. Git is an open-source tool that allows team members to work asynchronously, track changes, and merge their contributions efficiently. It also ensures that a complete authorship history is maintained for each modification ([Ram, 2013](#)).

2.3.4 Deployment

In the deployment phase, the web application of the proposed system needs to be made accessible online. Cloud computing services offer significant advantages over traditional on-premise solutions, as they eliminate the need for purchasing physical hardware, manually installing software, and configuring systems ([Armbrust et al., 2016](#)). This not only reduces overall costs but also minimizes the requirement for specialized technical expertise beyond the core scope of the project. Fortunately, there are various cloud service providers available, making it straightforward to deploy the application using their infrastructure. This streamlined process offers a more efficient and cost-effective alternative to conventional deployment methods.

According to the findings above, followed by the fact that current technical skills are matched with what is needed and the required technology to be used are available, stable and reliable. It can be concluded that this project is technically feasible.

2.4 Feasibility Studies

2.3.1 Economic Feasibility

The development of a software project includes several costs, such as hardware and software resources, design and development, and maintenance. These costs are checked to see if the project is financially possible. According to [Wilson \(2020\)](#), three main factors affect software costs: the type of software project, the size of the project, and the size of the development team.

The type of software project affects the costs. It could be a web-based project, a software update, or the creation of new software. The size of the project also matters. Projects can be small, medium, large, or enterprise-level. The size impacts both the development schedule and the costs. The third factor is the size of the development team. This is about how many people are working and their roles in the project.

For this project, there is no need for hardware purchases or subscription services. Instead, we can use open-source tools, which will lower costs. Tools like Visual Studio Code (VSC), Git, and the programming language frameworks are free to use. Using Visual Studio Code avoids the need to buy a source-code editor. Also, using cloud computing services for deployment can improve performance, scalability, and security while saving money [\(Promiso, 2024\)](#). Cloud technologies also let users pay based on their needs and build their own Cloud infrastructure with the suitable open source technologies [\(Meriam Mahjoub et al., 2011\)](#).

2.3.2 Legal Feasibility

Legal feasibility means following the laws and rules set by the government. For example, the Personal Data Protection Act 2010 (Act 709) ensures that personal data in commercial transactions is not misused or handled wrongly [\(MyGovernment, 2024\)](#). [PINTILIUC \(2018\)](#) says that protecting personal data means people can keep information that can identify them safe. The state must take steps to make sure this protection happens. The proposed system must meet these rules. It must have up-to-date user authentication, authorization, and proper data handling.

It is also important to follow payment rules. The system will handle payment transactions using credit/debit cards. It must follow the Payment Card Industry Data Security Standard (PCI DSS). The PCI DSS was created to reduce security risks in payment card transactions [\(Liu et al., 2010\)](#). The system must use a payment solution that follows both PCI DSS rules and the PDPA 2010 act.

2.3.3 Operational Feasibility

Operational feasibility is about checking if the system can meet long-term goals and fit smoothly into daily operations. It must also meet user expectations. The system should be efficient, effective, and reliable. Testing will make sure these qualities are present before the system is deployed. The testing will help achieve the project's goals, give benefits to users, and build trust in the software's ability to solve the identified issues. It will also fix problems seen in similar applications. The application's design and user interface will focus on the end-user to make sure they accept it.

Creating a user-friendly environment without losing efficiency in the ordering process is important for a better user experience. Also, consumers in Malaysia need to accept and adapt to digital ordering and payment methods. Many customers now prefer contactless cards, mobile wallets, and more instead of traditional cash systems.

[Statista Research Department \(2024\)](#) confirms this trend. This means the project is expected to have high user acceptance with little training needed.

As more people use the system, it needs to handle the increased demand. The system must be able to scale without problems once it is deployed. [Meriam Mahjoub et al. \(2011\)](#) say that cloud technologies can be adjusted to meet the developer's needs.

2.3.4 Schedule Feasibility

The StallSync project has a two-member team: Yeoh Zhe Heng and Loo Wai Kit. They will work on different parts of the project. One member will work on the management-side application (StallSync: Intelligent Management for Food Stalls). The other will develop the client-side application (StallSync: Intelligent Recommendation for Ordering). To complete the project on time and in an organized way, we will use a software development process that fits the project. We are using the Agile methodology because it allows flexibility and adaptability. The project is iterative. This means any changes in requirements or exploration of new code or libraries may need updates to the documentation.

2.3.5 Technical Feasibility

Technical feasibility involves assessing the availability of necessary resources, such as the suitability of chosen technologies and the development team's capability to meet project requirements ([Nandaniya, 2024](#)). The purpose is to evaluate the technical complexity and determine whether the required technology can effectively support the project's goals. Since the business-facing merchant portal will be developed as a web application, it is essential to analyze both software and hardware requirements for web-based development. Various technological aspects, including performance, usability, team expertise, scalability, long-term support, and maintainability, are carefully considered. This evaluation ensures that the selected tools and technologies meet the project's needs while improving the development process and workflow for the project team. As a result of research in literature review, there are several choices to be selected to design and develop the proposed system.

2.5 Chapter Summary and Evaluation

This chapter covers key topics like the project background, review of existing systems, and a feasibility study. The project background explains the idea, its goals, and the benefits of the project. It also describes the current state of the food ordering and delivery industry and suggests areas for improvement.

The review of existing systems looks at technologies available in the market that offer similar services. Three systems—FeedMe, EasyEat, and ShopeeFood—are analyzed. These systems are examined to understand their features, limitations, and gaps in meeting consumer needs. We evaluate each system's advantages and disadvantages. This helps identify how the project can improve and contribute to the market.

The feasibility study lists the available technologies and tools. It also outlines guidelines that can help the team achieve the project's goals. The study covers the economic, legal, operational, schedule, and technical feasibility of the project.

Chapter 3

Methodology and Requirements Analysis

3 System Analysis

This chapter explains the methodology used to develop the system and the chosen software process model. It also lists all system requirements in a clear order. Functional and non-functional requirements are detailed separately. Additional requirements, including development and operational needs, are also outlined. This structure ensures a smooth and efficient implementation process.

3.1 Proposed Software Process Model

3.1.1 Methodology

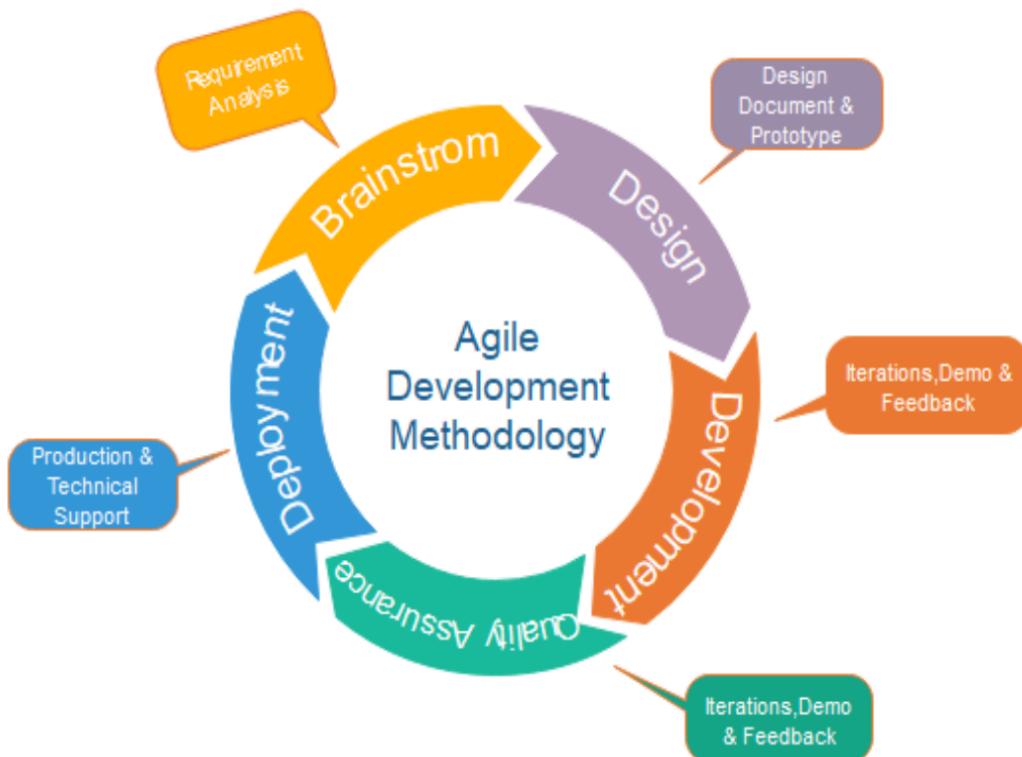


Figure 3.1: Agile Software Development Model

Source: <https://www.researchgate.net/figure/Outline-of-Agile-Software-Development-Methodology-9_fig1_362456799>

Agile Software Development is an iterative and flexible methodology. It focuses on adaptability, continuous improvement, and user collaboration. This approach ensures that software evolves efficiently through small, manageable cycles. It incorporates feedback and adapts to changing requirements.

The Agile process begins with Project Planning. In this phase, the team defines goals, scope, and constraints. Our project addressed this in Chapter 1, where we set clear objectives. These objectives focus on three key areas. First, they ensure the system's success and alignment with stakeholder needs. Second, they identify challenges and propose solutions. Third, they provide a structured timeline for development. We also created a Gantt chart to track progress.

This helps the team stay on schedule and meet deadlines. Unlike traditional methods, Agile allows for changes as the project evolves.

The next phase is Requirement Analysis. This step gathers detailed information about system needs. We conducted documentation research to compare three existing systems. This helped us understand their strengths and weaknesses. We used these insights to refine our own system. We also performed a feasibility analysis in five key areas. Technical feasibility assessed the available technology stack and tools. Schedule feasibility ensured realistic deadlines. Operational feasibility evaluated how well the system meets user needs. Legal feasibility checked compliance with regulations. Economic feasibility estimated costs and potential benefits. After gathering this information, we defined functional, non-functional, and technical requirements. Since Agile supports adaptive planning, we expect requirements to evolve throughout development.

The Design phase translates these requirements into a structured plan. This includes finalizing the architecture, methodology, and algorithms. We created high-level and low-level designs to document the system's structure. We also developed key diagrams. Entity-Relationship Diagrams (ERD) define database structure. Use Case Diagrams map user interactions. System Architecture ensures scalability and maintainability. Agile design is not rigid. It evolves as new insights emerge. Each iteration allows for adjustments, ensuring the design remains effective.

The Development phase follows Agile's core principles. Software is built in small, incremental cycles. Each feature is tested and integrated continuously. Our team consists of two developers (myself and Loo Wai Kit). We have two months to complete development. To manage time effectively, we adopted a modular approach. We break the system into small, manageable components. Each sprint focuses on specific features. After development, we review and gather feedback for improvements.

Once a module is complete, it undergoes Testing. Agile promotes continuous testing rather than a single final phase. Each sprint includes unit testing, integration testing, and user acceptance testing. This ensures defects are identified and fixed early.

After successful testing, we move to the Deployment phase. Agile encourages gradual releases. Early versions reach users quickly, allowing feedback collection. To support smooth adoption, we provide user guides, training sessions, and documentation.

Agile does not end with deployment. It emphasizes continuous review and feedback. After release, we collect user input to identify areas for improvement. This feedback informs future development cycles. Agile allows teams to revisit earlier stages when needed. This ensures the system adapts to new challenges and stakeholder needs. By following Agile's iterative approach, we create a flexible, efficient, and user-friendly system.

We choose Agile over traditional methods like Waterfall for several reasons. Agile enables continuous feedback, which improves software quality. It supports adaptive planning, making it ideal for changing requirements. Iterative development allows for early and frequent deliveries. Collaboration ensures the final product meets real-world needs. This structured but flexible approach helps deliver a high-quality system on time.

3.1.2 System Architecture

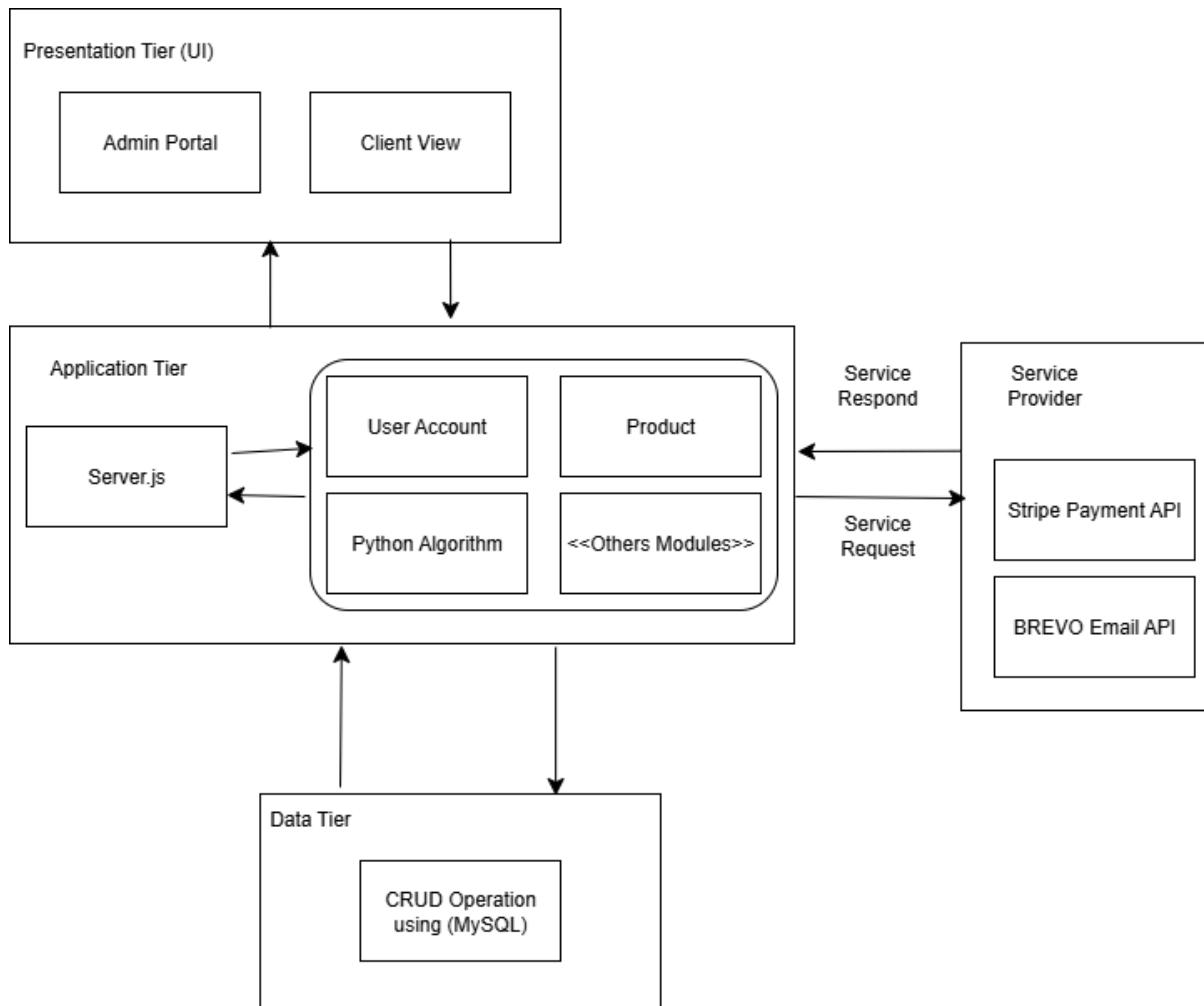


Figure 3.2: Architecture

StallSync adopts the hybrid architecture that combines 3-Tier architecture and Service-Oriented Architecture (SOA) which guarantee scalability, reliability, modularity, and interoperability. This approach allows for efficient system management, seamless integration with third-party services and improved performance and security.

In 3-Tier Architecture, due to the context for StallSync, it can provide scalable, reliable and maintainable structure that separates different components of the application into distinct tiers. These tiers work together to handle different aspects of the system, from user interaction and business logic to data management and security.

One of the key advantages of 3-Tier Architecture is scalability. Since each tier operates independently, they can be scaled separately to meet specific demands. For instance, if the number of users grows, the Presentation Tier (responsible for user interactions, such as browsing products and managing deliveries) can be scaled to handle increased traffic. Similarly, the Application Tier (handling business logic and request processing) can be expanded to support more transactions. If database queries

become a bottleneck, the Data Tier can be optimized through replication, indexing, and caching to improve performance.

Another critical benefit is reliability. Error handling is implemented at every tier to prevent failures. At the Presentation Tier, user inputs go through validation before submission. Once a request reaches the Application Tier, additional checks such as date comparisons, stock availability, and payment validations to ensure data accuracy. The Data Tier further validates information by enforcing data integrity constraints (e.g., string length, data types, and relational integrity via primary and foreign keys). This multi-layered validation prevents errors at the source and allows error messages to be displayed to users, ensuring a smooth experience and better system understanding.

The flexibility of 3-Tier Architecture also allows the use of different technologies for each tier. The Presentation Tier can leverage React, Angular, or Vue.js for web applications, while mobile apps can be developed using Flutter or React Native. The Application Tier can be powered by Node.js, Java, or Python, while the Data Tier can utilize relational databases (PostgreSQL, MySQL) or NoSQL databases (MongoDB, Firebase). This technology-agnostic approach enables developers to select the best tools for each function, improving performance and ensuring long-term adaptability.

Additionally, modularity and maintainability are significantly enhanced. Since each tier focuses on a specific function, developers can work on one tier without affecting the others. For example, a new dashboard feature for top consumers can be introduced in the Presentation Tier without modifying the underlying business logic or database. This modularity ensures faster development cycles, easier debugging, and smoother updates while minimizing system downtime.

SOA is integrated into the Application Tier to facilitate seamless communication with third-party services, ensuring the system remains scalable and modular. One of the main reasons for choosing SOA is its ability to integrate external services, which is the payment gateway integration. By adopting SOA, the system gains interoperability, allowing different services regardless of their technology stack to communicate effectively. For instance, the Payment Module can interact with FPX APIs and TNG eWallet APIs in the future, even if these payment services use different platforms or programming languages. This ensures smooth financial transactions without the need to develop payment processing mechanisms from scratch.

Another significant benefit of SOA is the development of reusable and maintainable services. Each service is modular and can be reused across different system components or future applications. This reduces development costs and effort, as updates to one service which do not disrupt the entire system. SOA also prevents ripple effects, meaning changes in one service do not unintentionally affect others, maintaining system stability.

In SOA implementation, services are designed as independent modules, with core business functions and integrations managed within the Application Tier. For example, the Payment Gateway Service handles secure payment processing using external payment gateway APIs. These services operate independently, ensuring scalability, security, and efficient system operations.

3.1.3 Technical Model Development Stack

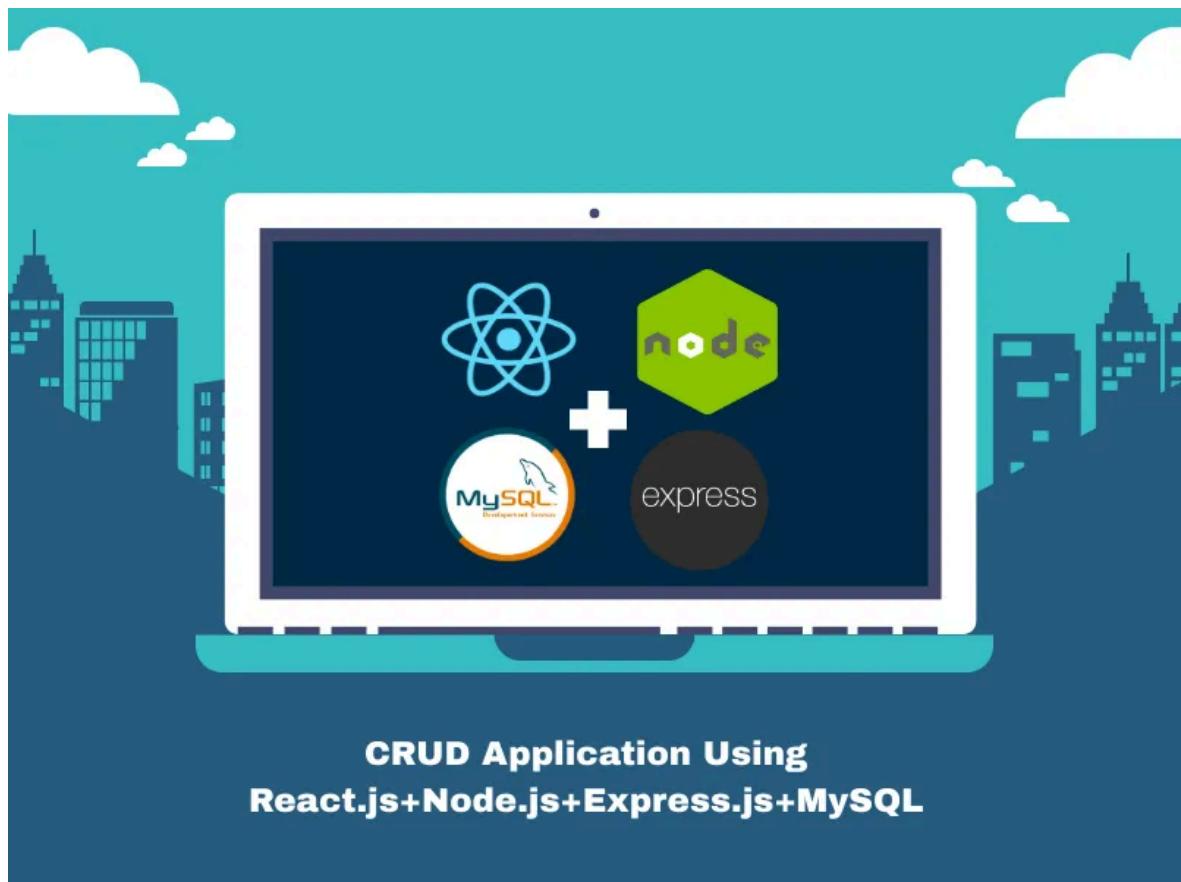


Figure 3.3: MERN with MySQL Tech Stack

Source: https://miro.medium.com/v2/resize:fit:1400/format:webp/1*xI-Cj8LCx2XFD_GswRxJ6Q.png

After evaluating tech stacks like MERN, MEAN, and MEVN, the stack combination MySQL , Express.js, React, and [Node.js](#) (MERN with MySQL) was chosen for the StallSync project. This decision was based on several factors. PostgreSQL offers a reliable and scalable database solution. The other technologies in the stack ensure strong performance and flexibility. Unlike the other stacks, MERN with MySQL supports REST APIs, which are crucial for efficient client-server communication.

Another important factor was the use of Python for the AI algorithm. While MERN with MySQL is based on JavaScript, it works well with Python. The REST API feature in the MERN with MySQL stack allows smooth communication between the

Node.js backend and Python-based services. This makes integrating the AI algorithm straightforward.

Additionally, MERN with MySQL fits the project's system architecture. The architecture combines a three-tier model and Service-Oriented Architecture (SOA). It divides the application into separate layers: presentation, logic, and data. This structure ensures clear separation of concerns and scalability. The MERN with MySQL stack supports this modular approach, making it an ideal choice for building a high-performance web application like StallSync.

3.1.4 ARIMA Model

The Reason on choosing ARIMA:

Firstly, the overview of the dataset will be shown with divided actual data (used to train) and tested data (used to evaluate):

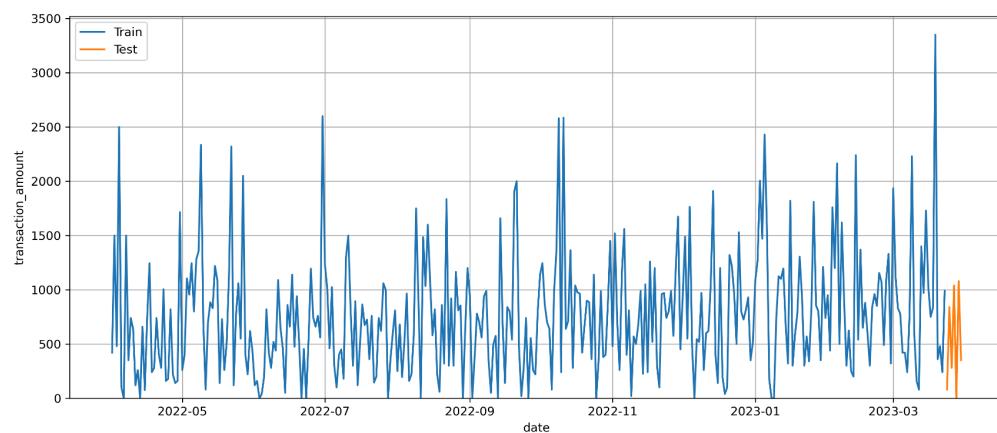


Figure 3.4: Dataset Overview Diagram

The source code and results will be displayed using figures and put inside Appendix A. These figures will show data over different time ranges, such as one month, three months, and one year. These time variations can affect the outcome and help determine the best Mean Absolute Error (MAE) value:

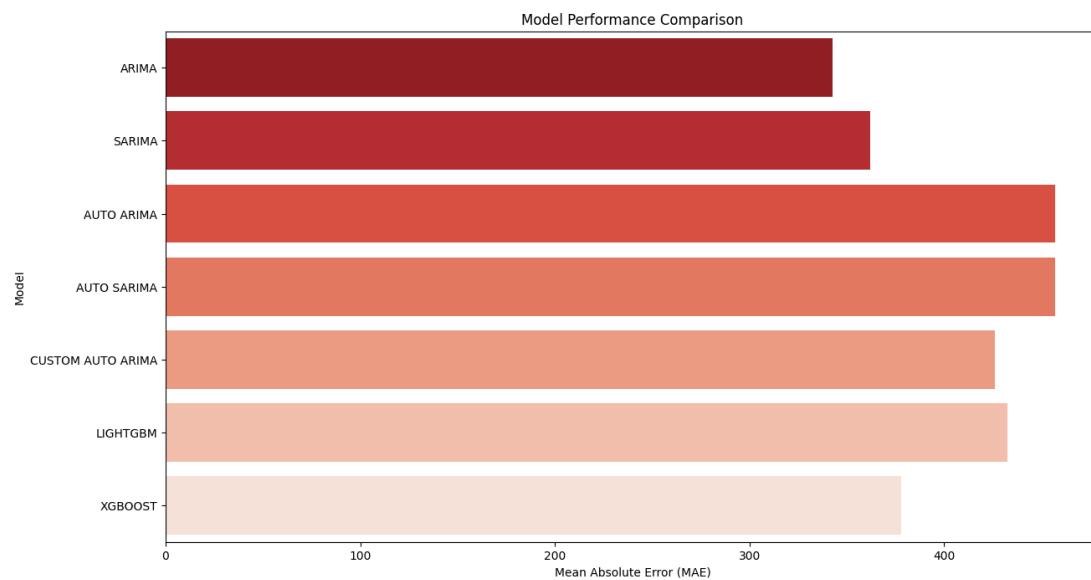


Figure 3.5: MAE Comparison Diagram for each model

Table 3.1: Model and MAE

Model	MAE
ARIMA	342.79
SARIMA	362.04
AUTO ARIMA	456.95
AUTO SARIMA	456.95
CUSTOM AUTO ARIMA	426.03
LIGHTGBM	432.64
XGBOOST	377.91

As a result, for each model's comparison, is shown by figure 3.5 or referring to Table 3.1. At the end the ARIMA will be chosen due to the lowest MAE value which is 342.79. The ARIMA result forecasting based on the overall dataset will be shown through figure 3.6.

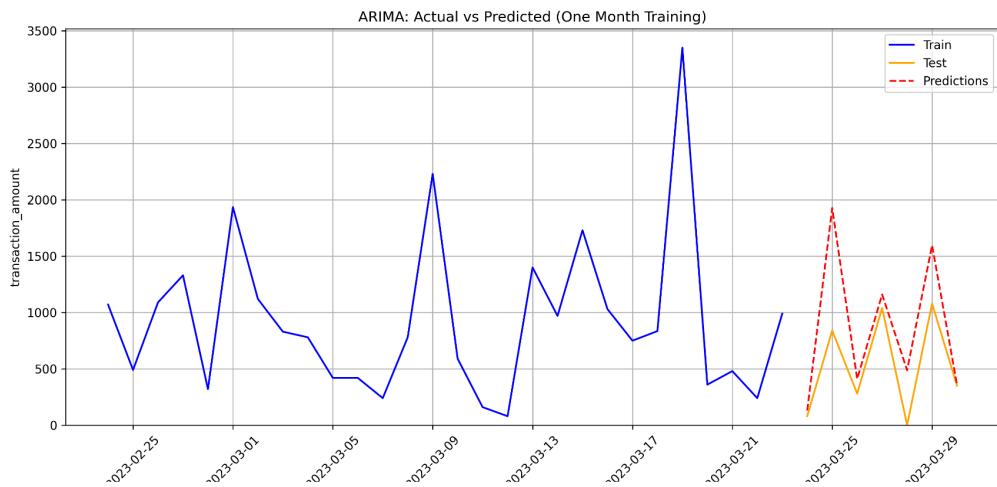


Figure 3.6: ARIMA Model Result

ARIMA Model Flows and Source Code explanation:

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import warnings
6 from sklearn.metrics import mean_absolute_error
7 from statsmodels.tsa.arima.model import ARIMA
8
9 warnings.filterwarnings('ignore')
10
11 # Load data
12 df = pd.read_csv("food.csv")
13
14 # Convert date column
15 df['date'] = df['date'].str.replace('/', '-')
16 df['date'] = pd.to_datetime(df['date'], format="mixed")
17
18 # Fill missing dates
19 date_range = pd.date_range(start=df["date"].min(), end=df["date"].max())
20 complete_dates = pd.DataFrame(date_range, columns=["date"])
21 df_by_date = df.groupby("date").agg({"transaction_amount": ["sum"]}).reset_index()
22 df_by_date.columns = ["date", "transaction_amount"]
23 df_complete = pd.merge(complete_dates, df_by_date, on="date", how="left").fillna(0)
24
25 # Define test size
26 test_size = 7
27 last_test_date = df_complete.iloc[-test_size]["date"]
28 one_month_before = last_test_date - pd.DateOffset(months=1)
29
30 # Keep only one month of training data
31 df_train = df_complete[(df_complete["date"] >= one_month_before) & (df_complete["date"] < last_test_date)]
32 df_test = df_complete.iloc[-test_size:]
33
34 # Train ARIMA model
35 p, d, q = 5, 0, 5
36 model = ARIMA(df_train['transaction_amount'], order=(p, d, q))
37 model_fit = model.fit()
38 test_predictions = model_fit.forecast(steps=len(df_test)).values
39 df_test["arima_pred"] = test_predictions
40
41 # Plot results (Only Last Month + Test Data)
42 plt.figure(figsize=(14, 6))
43 sns.lineplot(data=df_train, y="transaction_amount", x="date", label="Train", color="blue")
44 sns.lineplot(data=df_test, y="transaction_amount", x="date", label="Test", color="orange")
45 sns.lineplot(data=df_test, y="arima_pred", x="date", label="Predictions", color="red", linestyle="--dashed")
46 plt.title("ARIMA: Actual vs Predicted (One Month Training)")
47 plt.grid()
48 plt.ylim(0)
49 plt.xticks(rotation=45)
50 plt.savefig("arima_one_month.png", dpi=300)
51
52 # Calculate MAE
53 mae = mean_absolute_error(df_test["transaction_amount"], test_predictions)
54 print(f"Mean Absolute Error: {mae}")
55
56 # Save result to CSV
57 result_df = pd.DataFrame([{"model": "ARIMA", "mae": mae}])
58 result_df.to_csv("result.csv", mode='a', header=not pd.io.common.file_exists("result.csv"), index=False)

```

Figure 3.7: ARIMA Model Source Code

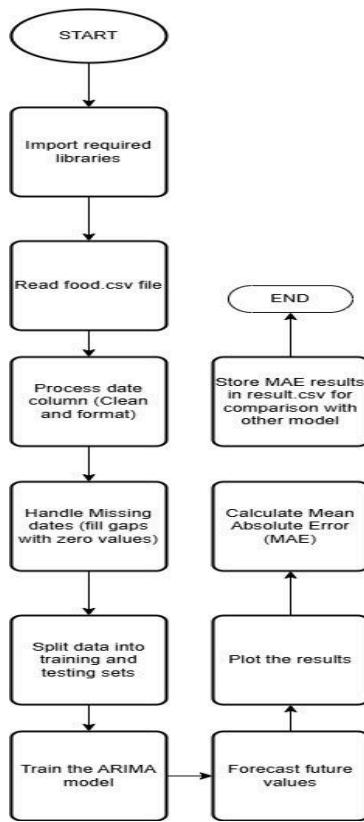


Figure 3.8: ARIMA Model Flow Chart

Figure 3.7 provides an overview of the ARIMA model source code used in this project. This Python script reads time-series data from a CSV file, processes missing values, applies the ARIMA model for forecasting, and visualizes the results. The following is a step-by-step breakdown of how the code works.

The first section (lines 1-7) imports the necessary libraries. NumPy and Pandas handle numerical computations and data manipulation. Matplotlib.pyplot and Seaborn help with data visualization. Warnings are used to suppress unnecessary messages. Mean Absolute Error (MAE) from sklearn.metrics evaluates prediction accuracy. Finally, the ARIMA model from statsmodels.tsa.arima.model is used for time-series forecasting.

Line 9 ensures unnecessary warnings are ignored. Next, the script reads the dataset (food.csv) into a Pandas DataFrame named df. Then, it cleans and formats the date column, making sure all dates are consistent.

The next step handles missing values. It creates a complete date range from the earliest to the latest date in the dataset. The script then merges the original data with this full range and fills any missing values with zero.

After cleaning the data, the script splits it into training and testing sets. It sets the test dataset size to four entries and identifies the last test date. The training data (df_train) includes all values up to four months before the last test date, while the testing data (df_test) contains the remaining values.

Next, the script applies the ARIMA model to forecast future values. It defines an ARIMA(4,0,4) model and trains it on the transaction_amount column. The model then predicts future values based on the test data.

After forecasting, the script visualizes the results. It plots three types of data:

- Blue Line – Actual training data
- Orange Line – Actual test data
- Red Dashed Line – Forecasted values

The generated plot is saved as "arima_forecast.png" and displayed.

Finally, the script calculates the Mean Absolute Error (MAE) to measure the accuracy of the forecast. This value is stored in a CSV file (result.csv) by creating a DataFrame that records the model type and MAE value. The script appends new results to the file for future comparisons with other forecasting models.

This structured approach ensures that raw data is cleaned, processed, analyzed, and used for accurate time-series forecasting, making the system effective for predicting future transaction trends. The overview of source code will be shown through Figure 3.8 which is a Flow Chart.

Adapting ARIMA in Project:

Our project uses the MERN with MySQL stack (MySQL, Express.js, React, Node.js) for development. However, the ARIMA (AutoRegressive Integrated Moving Average) model runs on Python. This model is used for time series forecasting. Since our backend is built with Node.js, we need a way to run Python scripts within it. We achieve this using the `child_process` module in Node.js. This module allows Node.js to execute Python scripts as separate processes.

When a user requests a forecast, the system triggers a Node.js function. This function calls a Python script that contains the ARIMA model. The script processes the input data and performs time series analysis. It then generates a forecasted output, such as a graph or numerical predictions. The system then stores this output on the server. Users can later retrieve and view the forecasted results.

We use ARIMA to predict future trends based on historical data. This helps users make informed decisions by analyzing patterns. Our system ensures smooth communication between Node.js and Python. Node.js handles user requests, while Python executes the forecasting algorithm. This setup makes the system efficient, scalable, and easy to integrate. Figure 3.9 shows an example of how the `child_process` module executes the Python script.

```
const { spawn } = require("child_process");

async function runPythonProcess(data, res) {
    try {
        const pythonProcess = spawn("python", ["./QL/app.py"]);

        pythonProcess.stdin.write(JSON.stringify(data));
        pythonProcess.stdin.end();

        let responseData = "";

        pythonProcess.stdout.on("data", (chunk) => {
            console.log("Raw Python Response:", chunk.toString());
            responseData += chunk.toString();
        });

        pythonProcess.stderr.on("data", (data) => {
            console.error("Python Error:", data.toString());
        });
    }

    pythonProcess.on("close", (code) => {
        if (!responseData.trim()) {
            console.error("No response received from Python script.");
            return res.status(500).json({ response_type: 3, response: "UnexpectedError" });
        }
    });

    try {
        const jsonResponse = JSON.parse(responseData.trim()); // Ensure complete JSON
        return res.json(jsonResponse);
    } catch (error) {
        console.error("JSON Parsing Error:", error);
        return res.status(500).json({ response_type: 3, response: "UnexpectedError" });
    }
};

} catch (error) {
    console.error("Error in processing:", error);
    return res.status(500).json({ response: "UnexpectedError" });
}
}
```

Figure 3.9 : Example source code for Nodejs child_process library

3.2 System Requirements

3.2.1 Functional Requirements

This proposed system management-side functional requirements are derived from the elicitation of requirements, it will be listed down based on the respective modules through Table 3.2. The user consists of admin, food stall owners, and staff.

Table 3.2: Functional requirements

Modules	Requirements
User Account	1) The system shall allow the user to login. 2) The system shall allow the register of a user account. 3) The system shall allow the user to logout. 4) The system shall allow the user for password recovery. 5) The system shall allow the user to edit profile. 6) The system shall allow the user to list user accounts. 7) The system shall allow the user to view the user profile.
Admin Dashboard	The system shall allow the user to view data statistic
Merchant	1) The system shall allow the user to create merchants. 2) The system shall allow the user to edit merchants. 3) The system shall allow the user to list out all the merchants. 4) The system shall allow the user to filter and sort out the merchants. 5) The system shall allow the user to view the merchant detail. 6) The system shall allow the user to delete merchants.
Product	1) The system shall allow the user to create products. 2) The system shall allow the user to edit products. 3) The system shall allow the user to view the product. 4) The system shall allow the user to list out the product. 5) The system shall allow the user to filter and sort the product list. 6) The system shall allow the user to delete products.
Reward	1) The system shall allow the user to create rewards. 2) The system shall allow the user to edit rewards. 3) The system shall allow the user to view the reward detail. 4) The system shall allow the user to list the rewards. 5) The system shall allow the user to filter and sort the reward list. 6) The system shall allow the user to delete rewards.
Order	1) The system shall allow the user to create orders. 2) The system shall allow the user to update order status. 3) The system shall allow the user to view the order detail. 4) The system shall allow the user to list the orders.

	5) The system shall allow the user to filter and sort the order list. 6) The system shall allow the user to cancel orders.
Announcement	1) The system shall allow the user to create announcements. 2) The system shall allow the user to view the Announcement details. 3) The system shall allow the user to list the Announcement. 4) The system shall allow the user to filter and sort the Announcement list. 5) The system shall allow the user to delete announcements.
Inventory	1) The system shall allow the user to stock in for product. 2) The system shall allow the user to stock out for products. 3) The system shall allow the user to list the inventory history.
Staff	1) The system shall allow the user to create staff. 2) The system shall allow the user to update staff. 3) The system shall allow the user to view the staff details. 4) The system shall allow the user to list the staff. 5) The system shall allow the user to filter and sort the staff list. 6) The system shall allow the user to delete staff.
Report	1) The system shall allow the user to create report 2) The system shall allow the user to list the report 3) The system shall allow the user to sort and filter the report 4) The system shall allow the user to forecast the report 5) The system shall allow the user to view the report

3.2.2 Non-Functional Requirements

The proposed system must uphold non-functional requirements to ensure quality, meet user expectations, and enhance user satisfaction, ultimately encouraging adoption, it will be shown through Table 3.3.

Table 3.3: Non-functional requirements

No.	Quality Attributes	Description
1	Reliability	Reliability refers to the system's ability to consistently perform its intended functions without failure. A reliable system is one that can handle errors effectively and continue operating smoothly even when unexpected issues arise. The system should be able for error handling and catch the unexpected error by displaying error messages. It should display clear error messages to inform users when something goes wrong. A reliable system ensures that

		users can trust the platform to function correctly, even during unforeseen circumstances. This builds user confidence and minimizes disruptions in their experience.
2	Usability	Usability refers to how easy and intuitive a system is for users to interact with. A system with good usability allows users to accomplish their tasks efficiently, without confusion or frustration. The system should be simple to use and easy to navigate. For example, it should not require users to navigate through more than three pages to complete a task. A user-friendly interface ensures that users can quickly learn how to use the system, leading to higher satisfaction and productivity. By focusing on usability, the system will reduce user errors and improve overall user experience.
3	Security	Security is an essential non-functional requirement that protects the system and its components from unauthorized access and malware. For StallSync, ensuring security is vital to safeguard user data and secure transactions. The system should include strong authentication mechanisms to verify user identities and prevent unauthorized access. The system must be able to identify different user roles. By focusing on security, the system aims to protect customer data and ensure safe interactions on the platform.
4	Maintainability	Maintainability refers to how easily a system can be updated, modified, or repaired over time. A maintainable system is designed to allow developers to make changes efficiently and without introducing errors. The system coding part should be structured well for easy maintenance and flexible for any components changing. This will make it simple to update or change components when needed. Well-organized code reduces the time and effort required for future modifications. It also makes it easier to fix issues or implement new features. A focus on maintainability helps the system adapt to changing requirements and technologies, ensuring long-term sustainability.
5	Availability	Availability refers to the system's ability to be accessible to users whenever needed. The proposed system will prioritize high availability, ensuring it is operational during business hours. The system should be available when it is needed in the business operating hours. To achieve this, the system must be

		designed with redundancy and fault tolerance in mind. This includes implementing backup servers, load balancing, and automated failover systems to minimize downtime and ensure continuous service. Additionally, the system should have a robust infrastructure and architecture to reduce the risk of disruptions and maintain reliable access for users.
6	Performance	Performance refers to how quickly a software system responds to user actions under a given workload. It measures the system's ability to handle requests and deliver responses promptly. For optimal performance, the system should respond within 1500 ms (1.5 seconds) . This ensures that users experience minimal delays and get timely feedback. The system must also handle peak loads during busy times, like lunchtime, while maintaining consistent performance.

3.3 Other Requirements

3.3.1 Development Environment

This project will incorporate a range of hardware, software, databases, programming languages, browsers, and diagramming tools to support the development process. These components are summarized in Table 3.1.

Table 3.4 Development process tools and environments

Hardware	Asus Zephyrus G14 - Ryzen 9-8945 HS - ROM: 1TB - RAM: 32GB
Operating System	Windows
Development Software	- Editor: VSC - Terminal: PowerShell - Database GUI: PG Admin 4 - Version Controlling: GIT
Browser	Google Chrome
Database	PostgreSQL
Programming language/ Framework	Nodejs, Reactjs, Python
Diagramming Tool	Lucidchart, Draw.io

3.3.2 Operational Environment

The proposed system is aimed to represent a web-based application, the devices with browsers and support network connection are able to use this application. The minimum requirements for using this application is shown through Table 3.2.

Table 3.5: Operational Environment and Requirement

Devices	- PC with installed browser - Tablet with installed browser - Smartphone with installed browser
Browser	- Modern browsers such as Chrome, Edge, Safari, and more
Pre-condition	Stable network connection

3.4 Chapter Summary and Evaluation

This chapter focuses on the methodology and requirements analysis for the StallSync project. First, it explains the software development process model that will be used. Then, it discusses the technology stack and the algorithm chosen for the project. The chapter also covers the development and operational environments in detail. It goes on to outline both functional and nonfunctional requirements for StallSync. To summarize the tools, the project will use the MERN with MySQL stack along with the ARIMA algorithm. Additionally, free and open-source tools like Git, Visual Studio Code (VSC), and Figma will be used in the development process.

Chapter 4

System Design

4 System Design

This chapter provides a clear overview of the system's design using diagrams and models to illustrate its structure, processes, and interactions. Process design outlines how data moves and how operations are executed, ensuring efficiency and clarity. Database design focuses on structuring and storing data effectively, defining relationships, and improving system performance. User interactions are mapped to show how different roles engage with the system, ensuring a user-friendly experience. Behavioral models capture system responses, state changes, and component interactions, helping to understand how the system processes requests. UI design ensures an intuitive interface that enhances usability. The chapter concludes with an evaluation, assessing how well the design meets system requirements and project objectives. A structured design approach ensures scalability, efficiency, and a strong foundation for development.

4.1 Process Design & Algorithm Design

This section outlines the implementation of a forecasting function to achieve the project's objectives.

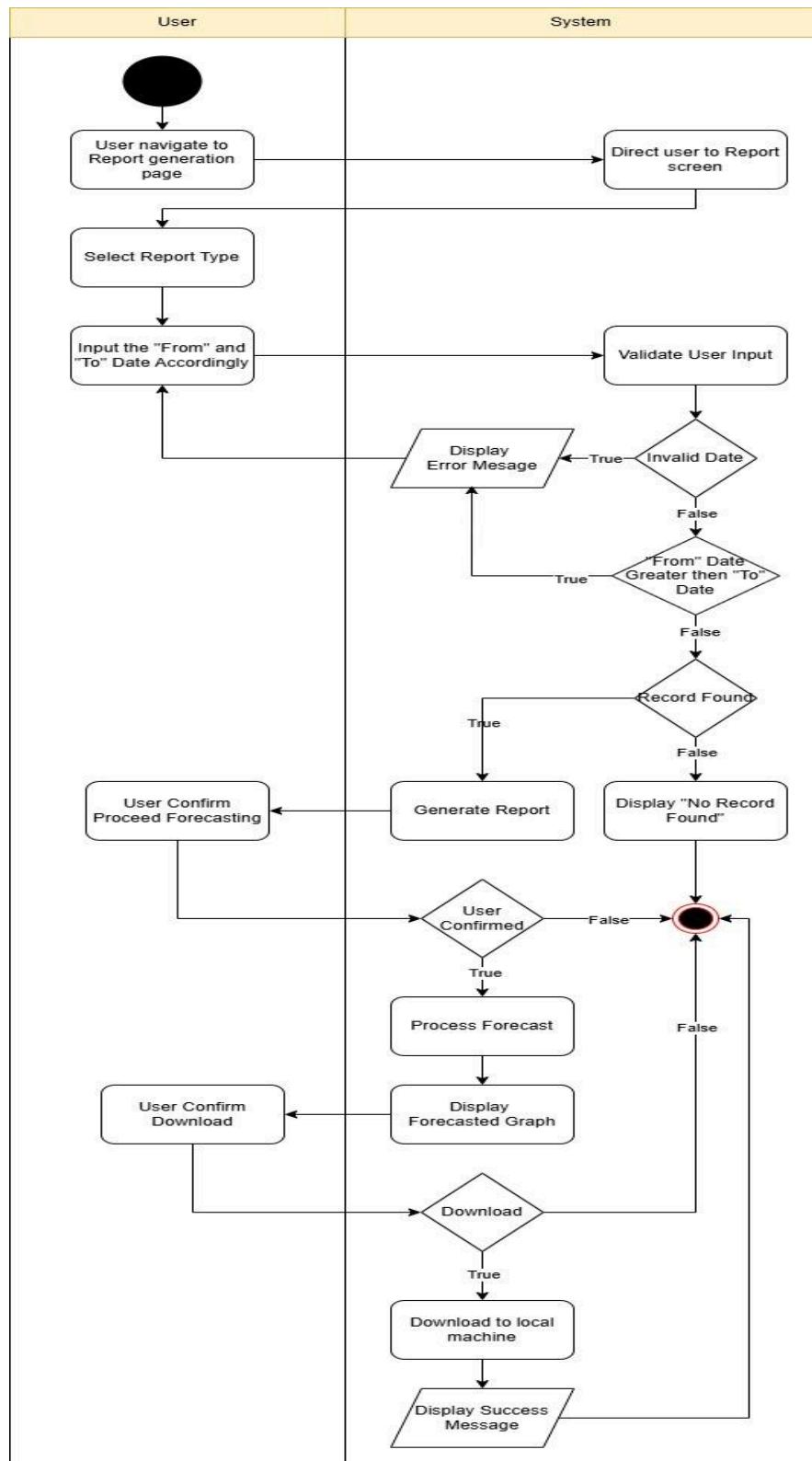


Figure 4.1: Activity Diagram on using Algorithm

Figure 4.1 illustrate the time-series forecasting process. The decision to utilize forecasting is presented as an optional choice. This is due to the potential for forecasting to consume significant time and produce results that may mislead food stall owners or the management team. Therefore, users can selectively engage the forecasting feature to inform their decision-making process.

4.2 Database design & Structural Models

4.2.1 Business Rules

Before Creating the Entity-Relationship Diagram, we will need the business rules as an abstraction to avoid us constructing the wrong diagram that may not meet requirements and user expectations. After the ERD is designed and constructed, the details for each table will be listed down in tables.

Business Rule:

- 1) The Customer have one user account and the user account is belongs to one customer
- 2) The Staff have one user account and the user account is belongs to one staff
- 3) The staff can create none, one or more Announcements and the announcement can be created by none or one staff
- 4) The customer has zero, one or many announcements and the announcement is sent to one or many customers.
- 5) The Customer have zero, one or many cart, cart is related to one customer
- 6) The cart will consist of one product where the product will exist in one or many carts.
- 7) The Product may have zero, one or many inventory records.
- 8) The staff consists of zero or one merchant (food stall) while the merchant has at least one staff member.
- 9) The staff can generate zero, one or many reports while the report can be generated by one staff.
- 10) A customer will have zero or at least one order while an order is belongs to one customer only
- 11) An order will have one or many products and the products will exist in one or many order
- 12) An order will have one related transaction and the transaction is belongs to only one order
- 13) An order can apply or not apply a reward, and a reward can be applied in zero, one or many orders.
- 14) The review rating is out of 5.
- 15) An order can only be done within a merchant.
- 16) Customer Tier will drop with a month no transaction record
- 17) Customer Tier will affect the points reward.
- 18) Customer will be rewarded 500 points at the initial
- 19) 1 points can be converted to 0.01 Currency
- 20) 1 Currency can earn 1 points
- 21) The minimum total amount will be 0
- 22) The points will be used all when applied

This chapter uses several diagrams to explain the food ordering system from different angles. The system design follows object-oriented principles and adopts a 3-tier architecture. A UML class diagram shows how users interact with the system. Activity diagrams outline the steps involved in each main function. A use case diagram and use case descriptions provide detailed flow of the system's processes, including alternative flows when specific conditions occur. These functions are organized into separate modules. An Entity-Relationship Diagram (ERD) and a data dictionary describe how data is structured and outline the system's data requirements. A basic GUI design is also included. It offers a preview of how users will

interact with the system in practice. Together, these diagrams provide a clear view of the system's design and behavior. This understanding will help ensure a smooth and successful implementation in the next phase.

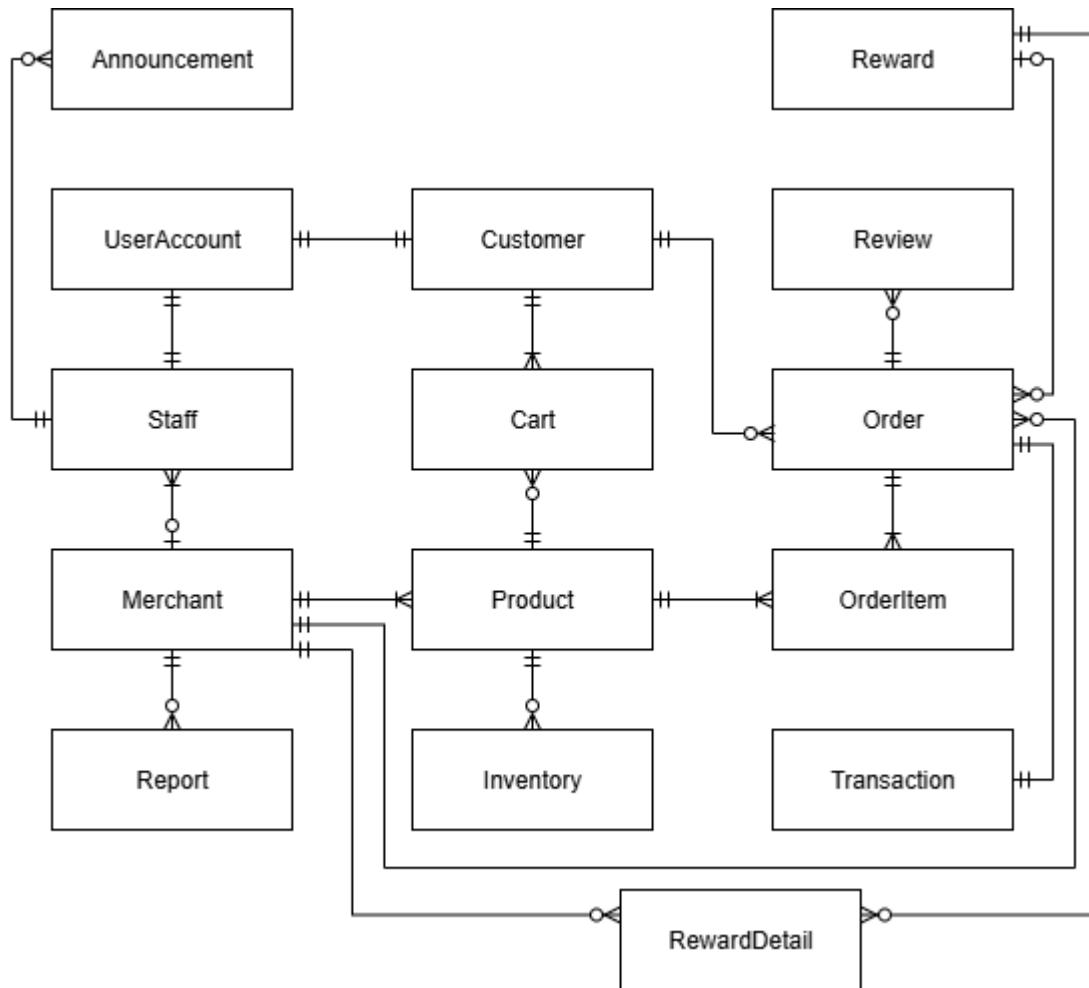


Figure 4.2: ERD overview 1

4.2.2 Data Dictionary

Announcement:

File Name:	pssysann						
File Description:	System Announcement						
No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psannuid	VARCHAR(25)	YES	Announcement Id			unique
2	psannttl	VARCHAR(50)	YES	Announcement Title			
3	psannmsg	TEXT	YES	Announcement Message			
4	psannimg	VARCHAR(255)	YES	Announcement Image			
5	psanndat	DATE	YES	Announcement Date		new Date()	
6	psannsts	VARCHAR(10)	YES	Announcement Status	YESORNO		Y - Yes, N - No
7	psanntyp	VARCHAR(10)	YES	Announcement Type	ANNTYP		EVT - Event, SYS - System, INF - Information
8	crtuser	VARCHAR(255)		Create User			
9	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.3: Announcement Table

User Account:

File Name:	prusrprf						
File Description:	User Profile						
No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psusrnun	VARCHAR(255)	YES	Username			unique
2	psusrnam	VARCHAR(255)	YES	Name			
3	psusrpwd	VARCHAR(255)	YES	Password			Encrypted with brcrypt salt
4	psusreml	VARCHAR(255)	YES	Email			
5	psusrrol	VARCHAR(3)	YES	User Role	USRROLE	MBR	ADM - ADMIN, MBR - MEMBER [This is for indicate login in different front end
6	psusrpre	VARCHAR(10)	YES	Phone No Prefix	HPPRE		MY - +60, US +1, CN +86, SG +65
7	psusrphn	VARCHAR(25)	YES	Phone No			
8	psusrsts	VARCHAR(10)	YES	User Status	USRSTS	A	A - Active, L - Locked, C - Close, E - Expired
9	psusrtyp	VARCHAR(10)	YES	User Type	USR_TYP	MBR	ADM - Admin, MCH - Merchant, MBR - Member
10	psusrdt8	DATE	YES	Status date		new Date()	
11	pspwdatm	INTEGER	YES	Password Attempts			
12	pschgpwd	BOOLEAN	YES	Changed Password			true for yes, false for no
13	psfstlgn	BOOLEAN	YES	First Login			true for yes, false for no
14	psmsgurd	VARCHAR(1)	YES	Message Unread	YESORNO	N	Y - Yes, N - No
15	crtuser	VARCHAR(255)		Create User			
16	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.4: User Account Table

Member:

File Name:	psmbrprf						
File Description:	Member Profile						
No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psmbuid	VARCHAR(25)	YES	Member Id			unique, Running No based on (customer)
2	psmbnam	VARCHAR(255)	YES	Member Name			
3	psmbreml	VARCHAR(255)	YES	Member Email			
4	psmbdob	DATE	YES	Date of Birth			
5	psmbpts	INTEGER		Member Points		500	Free 500 points, Spend RM1 = 1 point, Spend 100 points = RM1
6	psmbracs	DECIMAL(15,2)		Accumulated Spending			Accumulated when point added
7	psmbtyp	VARCHAR(10)		Membership Type	MBRTYP		B - Bronze, S - Silver, G - Gold
8	psmbrexp	DATE		Membership Expired		new Date() + 1 year	When [condition date] no spending will drop member tier, BATCH JOB
9	psmbjdt	DATE		Join Date		new Date()	
10	psmbrcar	VARCHAR(50)		Cart No		uuidv4()	auto creation
11	psusrname	VARCHAR(255)	YES	Username			link to psusrprf
12	psmbrpre	VARCHAR(10)	YES	Member Phone No Prefix	HPPRE		MY - +60, US +1, CN +86, SG +65
13	psmbrphn	VARCHAR(25)	YES	Member Phone No			
14	crtuser	VARCHAR(255)		Create User			
15	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.5: Member Table

Staff:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	pstfuid	VARCHAR(25)	YES	Staff Id			unique (running no MS0001, CS0001, AS0001)
2	pstfnme	VARCHAR(255)	YES	Full Name			
3	psmrcuid	VARCHAR(25)		Merchant Id			link to merchant id, when staffType != O or != O, then required
4	pstftyp	VARCHAR(10)	YES	Staff Type	STAFFTYP		A - Admin, S - Staff, O - Owner
5	pstfidt	VARCHAR(10)	YES	ID Type	IDTYPE		IC - Identity Card, PS - Passport
6	pstfidn	VARCHAR(25)	YES	ID No.			
7	pstfpfp	VARCHAR(255)		Profile Picture			
8	pstfnat	VARCHAR(10)	YES	Nationality	NATION	MY	MY - Malaysia, MM - Myanmar, VN - Vietnam, ID - Indonesia, PK - Pakistan
9	pstfdst	DATE		Status Date		new Date()	
10	pstfjdt	DATE		Join Date		new Date()	
11	pstfsts	VARCHAR(10)		Staff Status	YESORNO	Y	Y - Yes, N - No
12	pstfad1	VARCHAR(255)	YES	Current Address Line 1			
13	pstfad2	VARCHAR(255)	YES	Current Address Line 2			
14	pstfpos	VARCHAR(25)	YES	Current Postcode			
15	pstfcit	VARCHAR(25)	YES	Current City			
16	pstfssta	VARCHAR(25)	YES	Current State			
17	pstfchp	VARCHAR(25)	YES	Current Hp			
18	pstfsmam	VARCHAR(10)	YES	Same Address	YESORNO	N	Y - Yes, N - No
19	pstfha1	VARCHAR(255)	YES	Home Address Line 1			When Same address is YES, then use the Current one
20	pstfha2	VARCHAR(255)	YES	Home Address Line 2			When Same address is YES, then use the Current one
21	pstfhpo	VARCHAR(25)	YES	Home Postcode			When Same address is YES, then use the Current one
22	pstfhci	VARCHAR(25)	YES	Home City			When Same address is YES, then use the Current one
23	pstfhst	VARCHAR(25)	YES	Home State			When Same address is YES, then use the Current one
24	pstfeml	VARCHAR(25)	YES	Email Address			When Same address is YES, then use the Current one
25	pstfbnk	VARCHAR(10)	YES	Bank Name	BANK		HLBB - Hong Leong Bank, MBB - Maybank, CIMB - CIMB Bank, PBB - Public Bank
26	pstfacc	VARCHAR(25)	YES	Bank Account No			Only Accept numerical Value
27	pstfbnm	VARCHAR(255)	YES	Bank Account User Name			
28	pstfepr	VARCHAR(10)	YES	Emergency Contact Prefix	HPPRE		MY - +60, US +1, CN - +86, SG - +65
29	pstfeph	VARCHAR(25)	YES	Emergency Contact			
30	psusrnm	VARCHAR(255)	YES	Username			link to psusrprf
31	crtuser	VARCHAR(255)		Create User			
32	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.6: StaffTable

Merchant:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psmrcuid	VARCHAR(25)	YES	Merchant Id			unique
2	psmrcnme	VARCHAR(255)	YES	Merchant Name			
3	psmrcdsc	VARCHAR(255)	YES	Merchant Description			
4	psmrclds	VARCHAR(255)		Merchant Local Description			
5	psmrcsdst	DATE		Status Date		new Date()	
6	psmrcjdt	DATE	YES	Join Date		new Date()	
7	psmrcown	VARCHAR(25)		Merchant Owner			link to psstfpar
8	psmrcssm	VARCHAR(255)	YES	SSM No			
9	psmrcssc	VARCHAR(255)	YES	SSM Cert Image			
10	psmrcsts	VARCHAR(10)	YES	Merchant Status	YESORNO	Y	Y - Yes, N - No
11	psmrcbnk	VARCHAR(10)	YES	Merchant Bank	BANK		
12	psmrcacc	VARCHAR(25)	YES	Merchant Bank Account			
13	psmrbnm	VARCHAR(255)	YES	Merchant Bank Name			
14	psmrcsf1	VARCHAR(255)		Store Front Image			
15	psmrcppi	VARCHAR(255)		Profile Picture Image			
16	psmrcrtg	DECIMAL(2,1)		Rating			
17	psmrcrmk	TEXT		Remarks			
18	psmrcrtc	INTEGER	YES	Rating Counts		0	Auto increment when being reviewed
19	crtuser	VARCHAR(255)		Create User			
20	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.7: Merchant Table

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psmrcuid	VARCHAR(25)		Merchant Id			unique, link to psmrcpar
2	psmrcotyp	VARCHAR(10)		Merchant Type	MRCTYP		unique, LC - Local, VG - Vegetarian, KR - Korean, MX - Mixed, JP - Japan, TH - Thailand
3	crtuser	VARCHAR(255)		Create User			
4	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.8: Merchant Label Table

Product:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psprduid	VARCHAR(25)	YES	Product Id			unique, Running No (MP0001, CP0001, AP0001)
2	psprdnme	VARCHAR(255)	YES	Product Name			
3	psprddsc	VARCHAR(255)	YES	Product Description			
4	psprdimg	VARCHAR(255)		Product Image			
5	psprldds	VARCHAR(255)		Product Local Description			
6	psmr cuiid	VARCHAR(25)	YES	Merchant Id			link to psmrcpar
7	psprdtyp	VARCHAR(10)	YES	Product Type	PRODTYP		N - Noodle, R - Rice, B - Bread, BR - Beverage, T - Taco
8	psprdcat	VARCHAR(10)	YES	Product Category	PRODCAT		LC - Local, JP - Japanese, KR - Korean, WS - Western
9	psprdfvg	VARCHAR(10)	YES	For Vegetarian	YESORNO	N	Y- Yes, N - No
10	psprdhal	VARCHAR(10)	YES	Is Halal	YESORNO	N	Y- Yes, N - No
11	psprdsts	VARCHAR(10)	YES	Product status	PRODSTS	A	A - Available, L - Low Stock, S - Sold Out
12	psprdsdt	DATE	YES	Status Date		new Date()	
13	psprdcid	VARCHAR(10)	YES	Countable Indicator	YESORNO	N	Y- Yes, N - No
14	psprdlsr	INTEGER		Low Stock Reminder			10 required when countable
15	psprdstk	INTEGER		Stock			0 required when countable
16	psprdpri	DECIMAL(10,2)	YES	Discount Price			0
17	psprdcrd	DATE	YES	Creation Date		new Date()	
18	psprdrmk	TEXT		Remarks			
19	psprdrtg	DECIMAL(2,1)	YES	Rating			0
20	psprdtak	VARCHAR(1)	YES	Can Tabkeaway?		N	
21	psprdtpr	DECIMAL(10,2)		Takeaway Charge			0 Required when psprdtak == 'Y'
22	psprdtc	INTEGER	YES	Rating Counts			0 Auto increment when being reviewed
23	crtuser	VARCHAR(255)		Create User			
24	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.9: Product Table

Inventory:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psprdivn						
2	psinvsty	VARCHAR(10)	YES	Stock Type	STKTYP		I - StockIn, O - StockOut, S - Sold
3	psinvqty	INTEGER	YES	Quantity			
4	psinvsdt	DATE	YES	Stock Date			
5	psstkuid	VARCHAR(50)	YES	Stock Uid			
6	psinvven	VARCHAR(255)	YES	Vendor/Member Name			
7	psinvpri	DECIMAL(10,2)	YES	Inventory Price			
8	crtuser	VARCHAR(255)		Create User			
9	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.10: Inventory Table

Cart:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psmbrcar	VARCHAR(50)	YES	Cart Id			unique, link to psmbpar
2	psitmcno	INTEGER	YES	Cart Number			Increment ID, unique
3	psmrcuid	VARCHAR(25)	YES	Merchant Id			unique
4	psprduid	VARCHAR(25)	YES	Product Id			link to psprdpar
5	psitmqty	INTEGER	YES	Quantity			
6	psitmunt	DECIMAL(15,2)		Unit Price			Product Price
7	psitmsbt	DECIMAL(15,2)		Subtotal			
8	psitmrmk	TEXT		Remarks			

Figure 4.11: Cart Table

Order:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psorduid	VARCHAR(50)	YES	Order Id		uuidv4()	unique
2	psordodt	VARCHAR(25)	YES	Order Date		new Date()	
3	psordamt	DECIMAL(10,2)	YES	totalAmount			Before Reward Applied
4	psordrap	VARCHAR(10)	YES	Reward Applied	YESORNO	N	Y - Yes, N - No
5	psrwduid	VARCHAR(10)		Reward Id			
6	psorddpap	VARCHAR(10)	YES	Point Applied	YESORNO	N	Y - Yes, N - No
7	psordpdv	DECIMAL(10,2)		Point Discounted Value			
8	psordgra	DECIMAL(10,2)	YES	Grand Total			After RewardApplied
9	psmbruid	VARCHAR(25)		Member Id			
10	psordpre	VARCHAR(10)	YES	Phone No Prefix	HPPRE		When MemberId is available, use his phoneNo, else required to fill up
11	psordphn	VARCHAR(25)	YES	Member Phone No			
12	psmrcuid	VARCHAR(25)	YES	Merchant Id			
13	psordsts	VARCHAR(10)	YES	Order Status	ODRSTS	N	N - New, P - Paid, C - Cancelled, A - Preparing, D - Completed, G - Pending
14	psordcd	DATE		Order Completed Date			
15	psordsst	DECIMAL(10,2)	YES	SST Amount			

Figure 4.12: Order Table

Order Item:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psorduid	VARCHAR(50)	YES	Order Id			unique, link to psordpar
2	psprduid	VARCHAR(25)	YES	Product Id			unique, link to psprdpar
3	psitmcno	INTEGER	YES	Cart Number			unique Increment ID
4	psitmqty	INTEGER	YES	Quantity		1	
5	psitmrmk	TEXT		Remarks			
6	psitmsbt	DECIMAL(10,2)	YES	Subtotal			
7	psitmunt	DECIMAL(15,2)		Unit Price			

Figure 4.13 Order Item Table

Review:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psordrvw	VARCHAR(50)	YES	OrderId			unique, link to psodpar
2	psrvwrtg	INTEGER	YES	Rating			1 to 5 only
3	psrvwdsc	VARCHAR(255)	YES	Review Description			
4	crtuser	VARCHAR(255)		Create User			
5	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.14: Review Table

Transaction:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	pstrxuid	VARCHAR(50)	YES	Transaction Id		uuidv4	unique
2	psorduid	VARCHAR(50)	YES	Order Id			link to psodrpar
3	pstrxdat	DATE	YES	Transaction Date		new Date	
4	pstrxamt	DECIMAL(10,2)	YES	Transaction Amount			
5	pstrxsts	VARCHAR(10)	YES	Transaction Status	TRXSTS		N - New, CA - Cancelled, C - Completed
6	pstrxmtd	VARCHAR(10)		Transaction method	PYMTD		O - Cashless, C - Cash

Figure 4.15: Transaction Table

Reward:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psrwuid	VARCHAR(10)	YES	Reward Id			unique
2	psrwdnme	VARCHAR(10)	YES	Reward Name			
3	psrwdsc	VARCHAR(255)	YES	Reward Description			
4	psrwdlds	VARCHAR(255)		Reward Local Description			
5	psrwdfdt	DATE	YES	From Date			
6	psrwdtdt	DATE	YES	To Date			
7	psrwdtyp	VARCHAR(10)	YES	Discount Type	DISTYPE		P - Percentage Discount, V - Value Discount
8	psrwddva	DECIMAL(10,4)	YES	Discount Value			More than 0, if
9	psrwdism	VARCHAR(10)	YES	Min Indicator	YESORNO	N	Y - Yes, N - No
10	psrwadmin	DECIMAL(10,2)	YES	Min Amount		0	
11	psrwdcia	VARCHAR(10)	YES	Capped indicator	YESORNO	N	Y - Yes, N - No
12	psrwdcap	DECIMAL(10,2)	YES	Capped Amount		0	
13	psrwdam	VARCHAR(10)	YES	Applicable to all merchant	YESORNO	Y	Y - Yes, N - No
14	psrwdsts	VARCHAR(10)	YES	Reward Status	RWDSTS	I	I - Incoming, A - Available, P - Past, O - Out of Stock
15	psrwdqty	INTEGER	YES	Quantity			
16	crtuser	VARCHAR(255)		Create User			
17	mntuser	VARCHAR(255)		Maintenance User			

Figure 4.16: Reward Table

Reward Detail:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	psrwuid	VARCHAR(10)	YES	Reward Id			link to pswdpar
2	psmrcuid	VARCHAR(25)	YES	Merchant Id			link to psmrcpar

Figure 4.17: Reward Detail Table

Report:

No.	FieldName	fieldType	required?	fieldDesc	genType	Default	Remarks
1	prrphis	VARCHAR(50)	YES	Report Name			unique
2	prrptgdt	VARCHAR(255)	YES	Report Generated Date			
3	prrptsts	VARCHAR(255)	YES	Report Status	RPTSTS		P - Pending, E - Generated Error, C - Completed
4	prrptusr	VARCHAR(25)		User Generated			reportType is M only required, link to psmrcpar
5	prrptmch	VARCHAR(255)	YES	Merchant Generated			
6	prrpfcs	VARCHAR(255)	YES	Is Forecast Sales	YESORNO	N	Y - Yes, N - No
7	prrptfco	VARCHAR(255)	YES	Is Forecast Order	YESORNO	N	Y - Yes, N - No
8	prctusr	VARCHAR(255)		Creation User			
9	prmtusr	VARCHAR(255)		Maintenance User			

Figure 4.18: Report Table

4.2.3 ERD Diagram

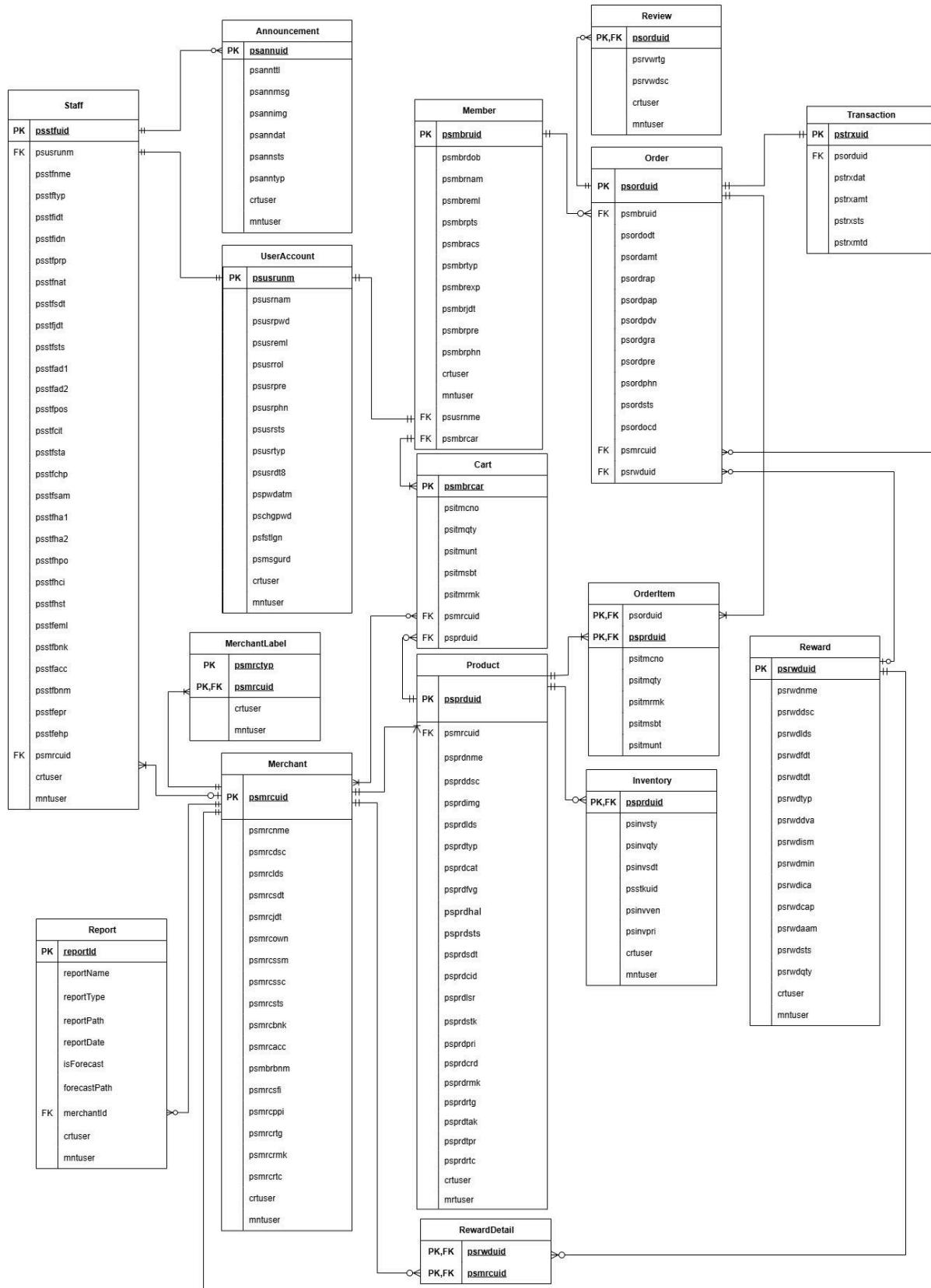


Figure 4.19: ERD Overview 2

4.2.4 Class Diagram

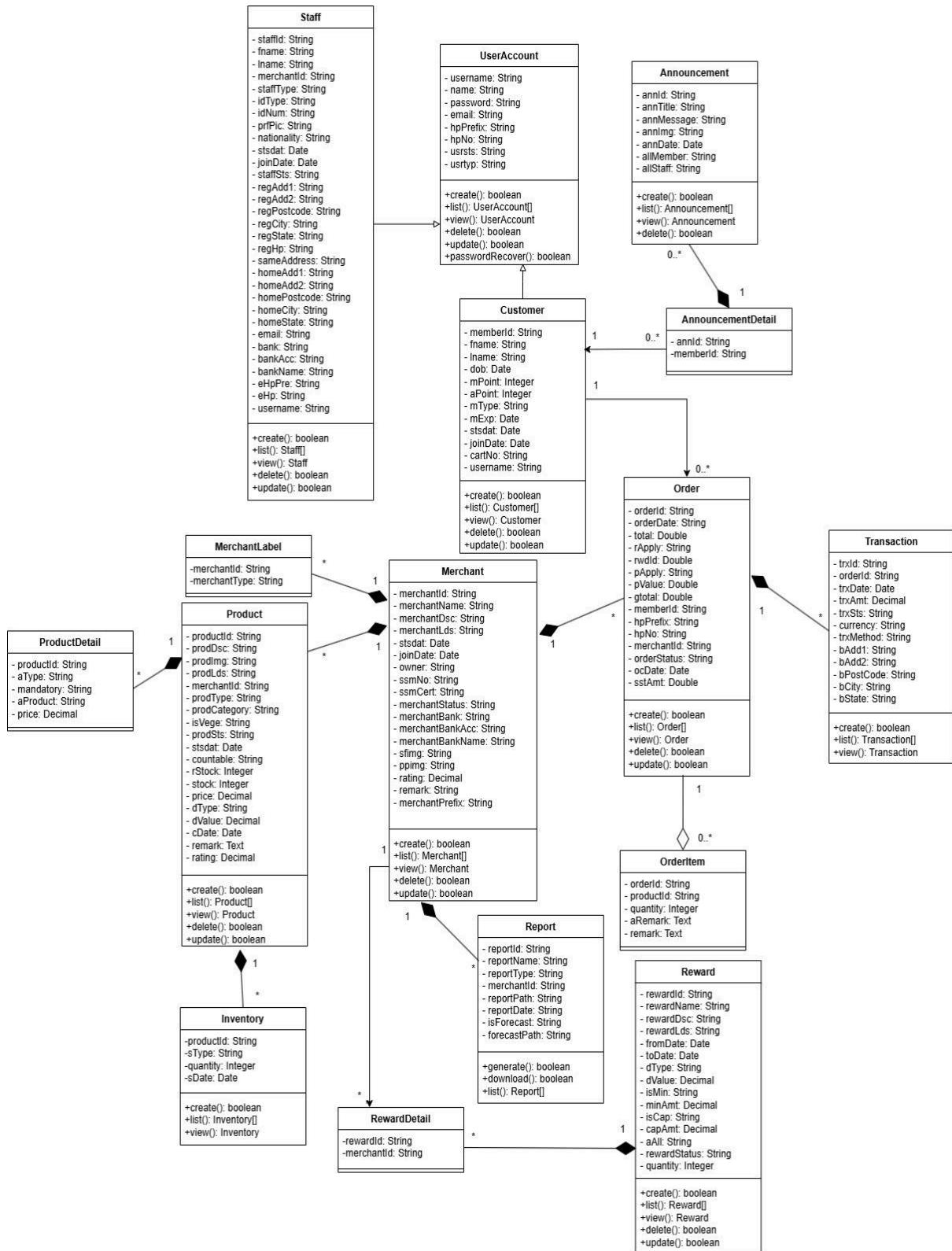


Figure 4.20: Class Diagram

4.3 Interaction Models (Main Use Case and Modules Use Cases)

4.3.1 Main Use Case

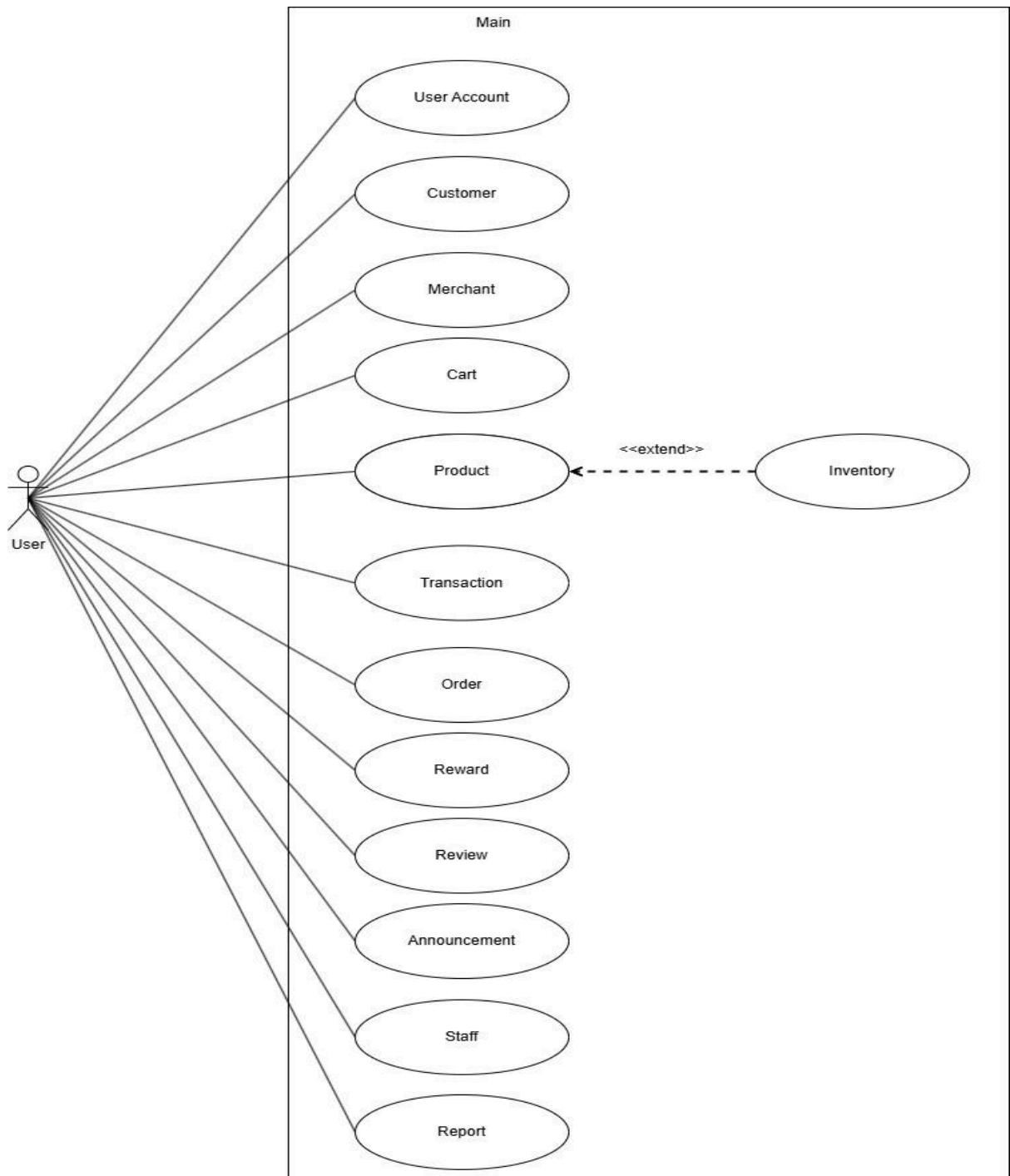


Figure 4.21: Main Use Cases

4.3.2 User Account Module

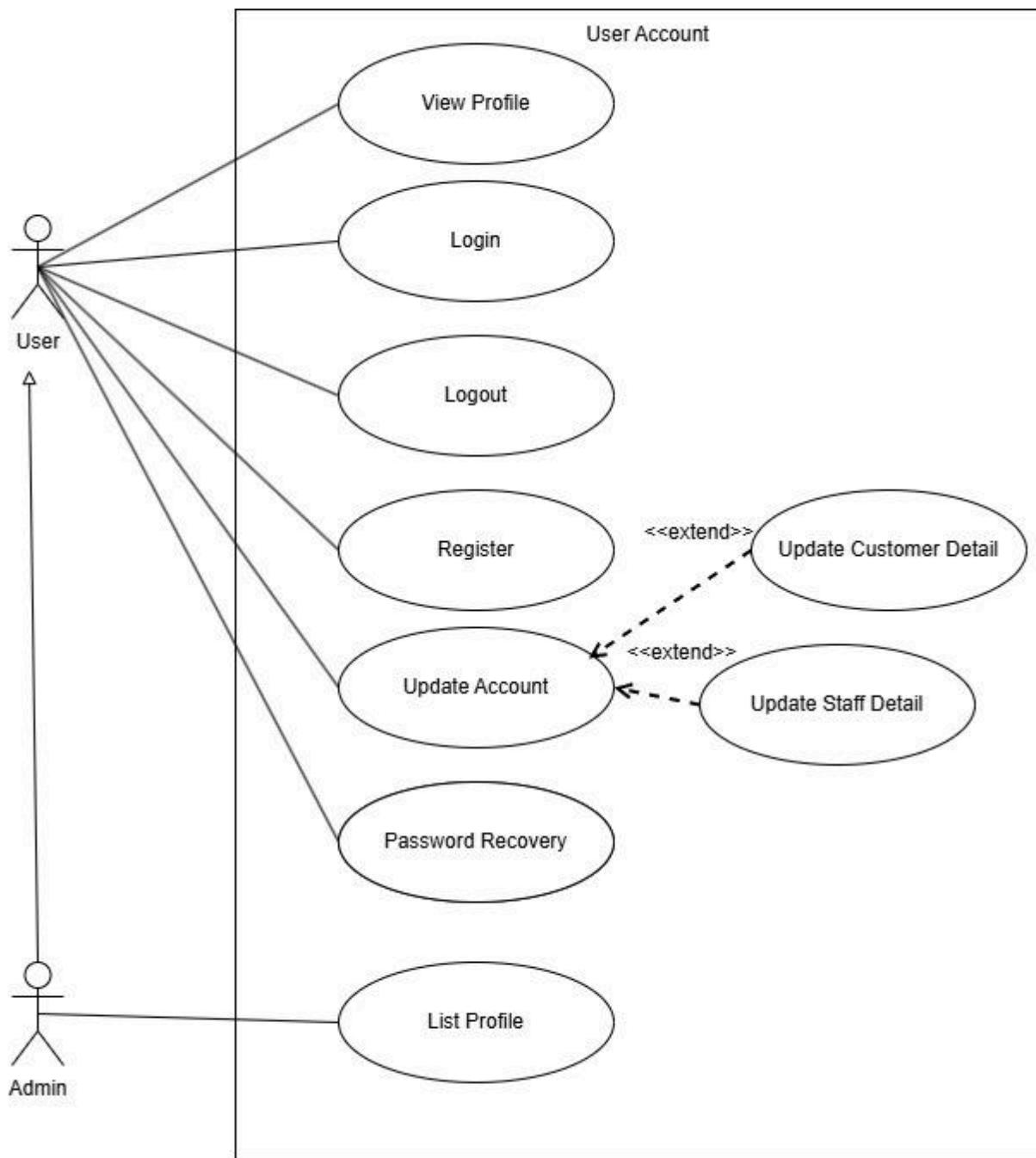


Figure 4.22: User Account Use Case

Table 4.1: User Login Use Case

Use case name: Login	
Pre-conditions: There is existing of user account	
Actor: All user	
Use case description: This use case is to describe the action for login into the system.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user direct to our system	2. The system validates the header session 3. If user is not logged in, display the user login page
4. User input the username and password accordingly and click the Login button	5. The system checks the validity of the login form. 6. If the payment info form is valid, direct user to the homepage of the system and update user login records
Alternative Flow Event	
A1: Step 2, The session header is returned with the condition of logged in. Return to step 6. A2. Step 5. Username is not in the system , Return error message “Username does not exist”. Return to step 4. A3. Step 5. Password not matching , Return error message “Password is wrong”. Return to step 4.	

Table 4.2: User Register Use Case

Use case name: Register user account	
Pre-conditions: N/A	
Actor: Customer	
Use case description: This use case is to describe the action for registering a user account by own	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. User click “Register account” in login page	
	2. Display the register form
3. User input the form	
	4. System Validate if the form is filled up correctly 5. Save the record in database and Direct user to login page
Alternative Flow Event	
A1: Step 4. Register form is invalid, return error messages and return step 3.	

Table 4.3: User Logout Use Case

Use case name: Logout	
Pre-conditions: The user is logged in	
Actor: All User	
Use case description: This use case is to describe the action to logout	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. User click the “Logout” button	
	2. System prompt the confirmation message
3. User select the popped up options	
	4. System validate the user selection as confirmed 5. Proceed to logout. Update the session records of the user.
Alternative Flow Event	
A1: Step 4. User selected “Cancel”, return to step 1	

Table 4.4: User Password Recovery Use Case

Use case name: Password Recovery	
Pre-conditions: Existing user account	
Actor: All User	
Use case description: This use case is to describe the action to recover the password	
Main Flow Event/ Basic flow	
Actor Action	System Response
6. User click the “forgot Password” in login page	
	7. System shows the password recovery form
8. User filled up accordingly	
	9. System validate the filled up form 10. System provide the new password form
11. User fill up the new password	
	12. System validate the inputted new password 13. New password is stored in the database and directs the user back to the login page.
Alternative Flow Event	
A1: Step 4. Invalid form, display error messages, return to step 3	
A2: Step 7. New password invalid, system display error message and direct user back to step 6	

Table 4.5: User Edit Profile Use Case

Use case name: Edit profile	
Pre-conditions: Existing user account	
Actor: All User	
Use case description: This use case is to describe the action to edit the profile	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. User click the “Profile” icon	
	2. System direct user to own account details
3. User change the ideal changes accordingly and click the “Save” button	
	4. Validate the changes is whether valid 5. If is valid, then update the records in the database and shows the success message
Alternative Flow Event	
A1: Step 4. Invalid changes, shows error messages and direct user back to step 2	

Table 4.6: List user account Use Case

Use case name: List user account	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action to list profiles	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “User account” from navigation	
	2. System identify whether the user have accessibility 3. Display the list of user
Alternative Flow Event	
A1: Step 2. Users don't have the accessibility to view the list. Return to homepage	

Table 4.7: View User Account Use Case

Use case name: View Profile	
Pre-conditions: Logged in user	
Actor: All user	
Use case description: This use case is to describe the action to view profile	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. User click the “Profile” icon	
	2. Direct user to the profile details and shows as form
Alternative Flow Event	
N/A	

4.3.3 Merchant

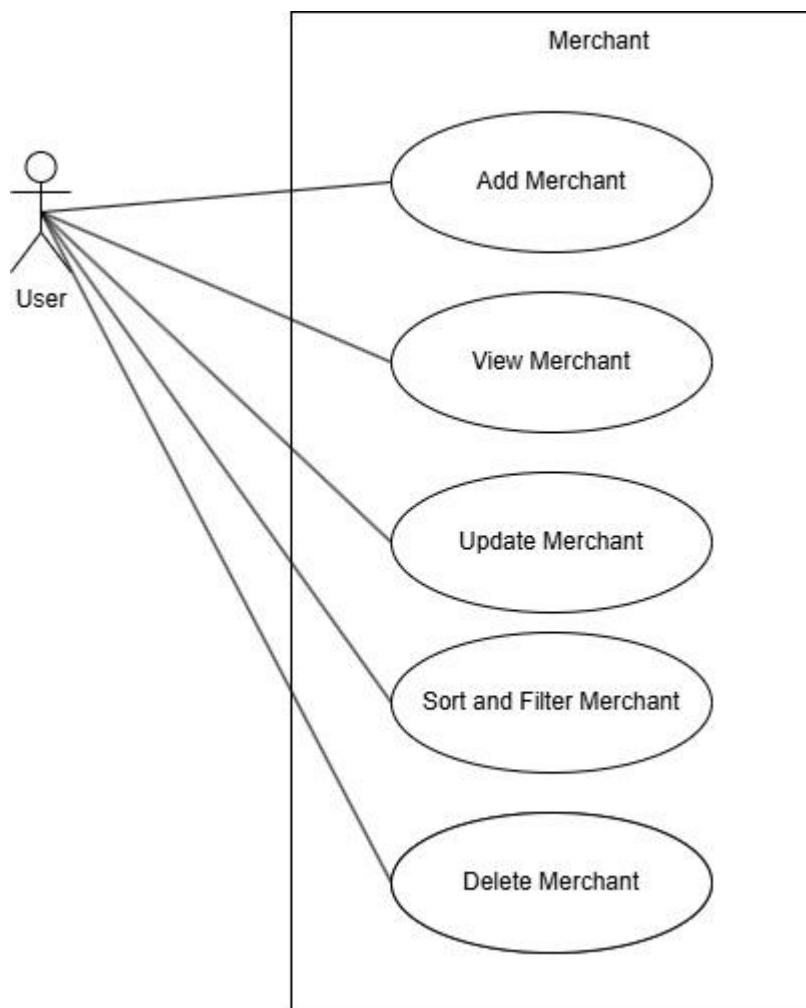


Figure 4.23: Merchant Use Case

Table 4.8: Create Merchant Use Case

Use case name: Create Merchant				
Pre-conditions: Logged in user				
Actor: Admin				
Use case description: This use case is to describe the action for Create the merchant				
Main Flow Event/ Basic flow				
<table border="1"> <thead> <tr> <th>Actor Action</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1. The user click the “Merchant” from the navigation bar</td> <td></td></tr> </tbody> </table>	Actor Action	System Response	1. The user click the “Merchant” from the navigation bar	
Actor Action	System Response			
1. The user click the “Merchant” from the navigation bar				

	2. System direct the user to the Merchant list page
3. User click “Add” Button	
	4. The system shows the merchant creation form
5. User fill up the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 8. If the form is invalid, it will show the error message. Return to step 5 A2: Step 11. If the user click the “Cancel” button, return to step 5	

Table 4.9: Edit Merchant Use Case

Use case name: Edit Merchant	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Editing the merchant.	
Actor Action	System Response
1. The user click the “Merchant” from the navigation bar	
	2. System direct the user to the Merchant list page
3. User click “Edit” Button beside the ideally edit record	
	4. The system shows the merchant edit form with previously added record
5. User do editing the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.

<p>Alternative Flow Event</p>
<p>A1: Step 8. If the form is invalid, it will show the error message. Return to step 5</p>
<p>A2: Step 11. If the user click the “Cancel” button, return to step 5</p>

Table 4.10: List Merchant Use Case

Use case name: List Merchant	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for List the merchant records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Merchant” from the navigation bar	
	2. System direct the user to the Merchant list page
Alternative Flow Event	
N/A	

Table 4.11: Sort and filter merchant Use Case

Use case name: Sort and filter Merchant	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for sort and filter listing the merchant records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
3. The user click the “Merchant” from the navigation bar	
	4. System direct the user to the Merchant list page
5. User input the sort and filter options	
	6. System based on the criteria to do the sorting and filtering 7. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.12: View Merchant Use Case

Use case name: View Merchant	
Pre-conditions: Logged in user	
Actor: Admin, Food Stall Owner, Food Stall Staff	
Use case description: This use case is to describe the action for View the merchant records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Merchant” from the navigation bar	
	2. System direct the user to the Merchant list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.13: Delete Merchant Use Case

Use case name: Delete Merchant	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for delete the merchant record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Merchant” from the navigation bar	
	2. System direct the user to the Merchant list page
3. User select the “Delete” button beside the ideally deleted record	
	4. System pop up the dialog for confirmation
5. The user select confirm	
	6. System determine whether the user select which button 7. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 7, if user select “Cancel”, there will be nothing happens then return to Step 2	

4.3.4 Product and Inventory

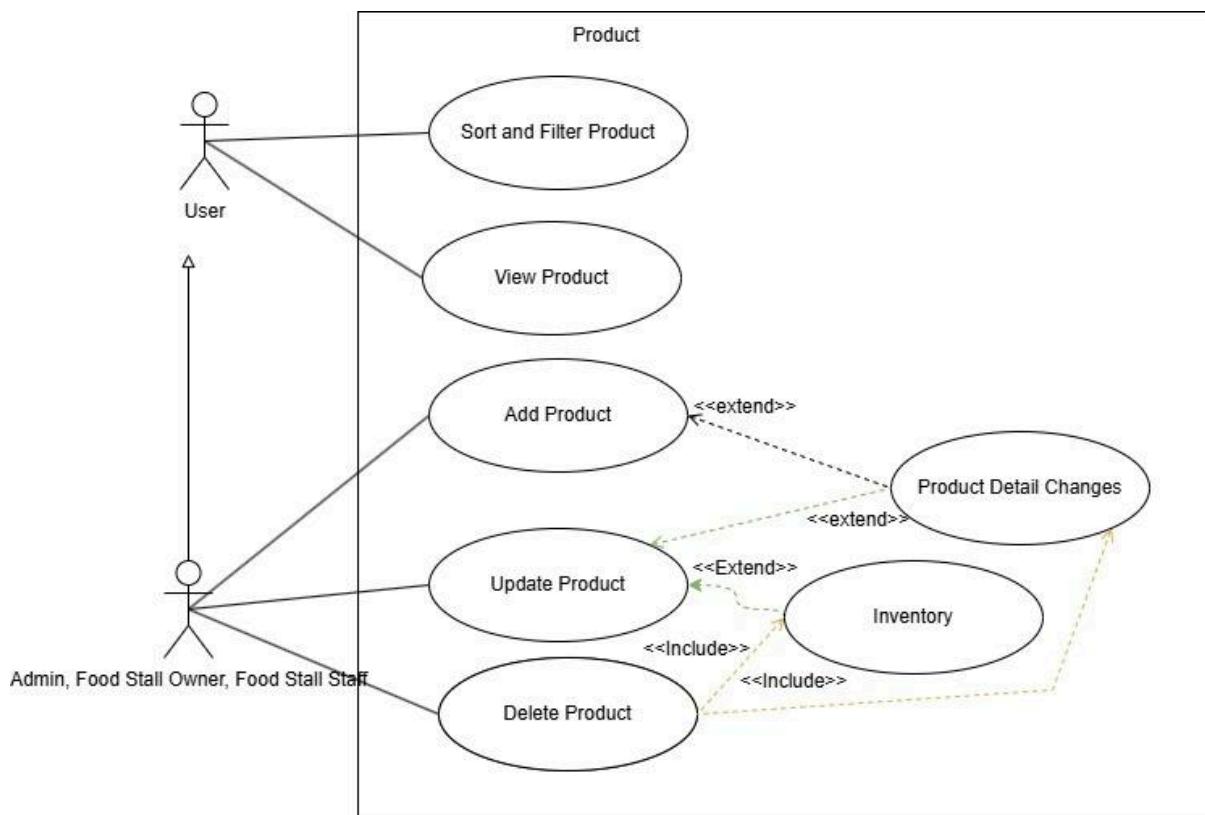


Figure 4.24:Product and Inventory Use Case

Table 4.14: Create Product Use Case

Use case name: Create Product	
Pre-conditions: Logged in user	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for Create the product	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User click “Add” Button	

	4. The system shows the product creation form
5. User fill up the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
	A1: Step 8. If the form is invalid, it will show the error message. Return to step 5 A2: Step 11. If the user click the “Cancel” button, return to step 5

Table 4.15: Edit Product Use Case

Use case name: Edit Product	
Pre-conditions: Logged in user	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for Editing the product.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User click “Edit” Button beside the ideally edit record	
	4. The system shows the product edit form with previously added record
5. User do editing the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.

Alternative Flow Event

A1: Step 8. If the form is invalid, it will show the error message. Return to step 5

A2: Step 11. If the user click the “Cancel” button, return to step 5

Table 4.16: List Product Use Case

Use case name: List Product	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for List the product records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
Alternative Flow Event	
N/A	

Table 4.17: Sort and filter Product Use Case

Use case name: Sort and filter Product	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the product records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User input the sort and filter options	
	4. System based on the criteria to do the sorting and filtering 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.18: View Product Use Case

Use case name: View Product	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for View the product records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.19: Delete Product Use Case

Use case name: Delete Product	
Pre-conditions: Logged in user	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for delete the product record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User select the “Delete” button beside the ideally deleted record	
	4. System pop up the dialog for confirmation
5. The user select confirm	
	6. System determine whether the user select which button 7. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 7, if user select “Cancel”, there will be nothing happens then return to Step 2	

Table 4.20: Add Inventory Use Case

Use case name: Add Inventory Record	
Pre-conditions: Logged in user	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for Create the inventory record for product	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User click “Edit” Button beside ideally stock in or stock out product	
	4. The system shows the product details form
5. User navigate by tapping the “Inventory” tab, if the product is countable	
	6. System shows the inventory list page
7. User click “Add” Button	
	8. System shows the Inventory field
9. User fill up the form accordingly	
10. User click the “Save” button	

	11. Validate the submitted form 12. If the form is valid, shows the confirmation dialog
13. User do selection from the confirmation	
	14. System validate the user confirmation selection 15. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 12. If the form is invalid, it will show the error message. Return to step 9 A2: Step 15. If the user click the “Cancel” button, return to step 9	

Table 4.21: Edit Inventory Use Case

Use case name: Edit Inventory	
Pre-conditions: Logged in user	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for Edit the inventory record for product	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page
3. User click “Edit” Button beside ideally stock in or stock out product	
	4. The system shows the product details form
5. User navigate by tapping the “Inventory” tab, if the product is countable	
	6. System shows the inventory list page
7. User click “Edit” Button beside the ideally edit record	
	8. System shows the Inventory form with the details previously create
9. User edit the form accordingly	

10. User click the “Save” button	
	11. Validate the submitted form 12. If the form is valid, shows the confirmation dialog
13. User do selection from the confirmation	
	14. System validate the user confirmation selection 15. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 12. If the form is invalid, it will show the error message. Return to step 9 A2: Step 15. If the user click the “Cancel” button, return to step 9	

Table 4.22: List Inventory History Use Case

Pre-conditions: List inventory history	
Actor: Food Stall owner, Admin, Food Stall Staff	
Use case description: This use case is to describe the action for List the product inventory records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Product” from the navigation bar	
	2. System direct the user to the Product list page

3. User click “Edit” Button beside ideally stock in or stock out product	
	4. The system shows the product details form
5. User navigate by tapping the “Inventory” tab, if the product is countable	
	6. System shows the inventory list page
Alternative Flow Event	
N/A	

4.3.5 Order

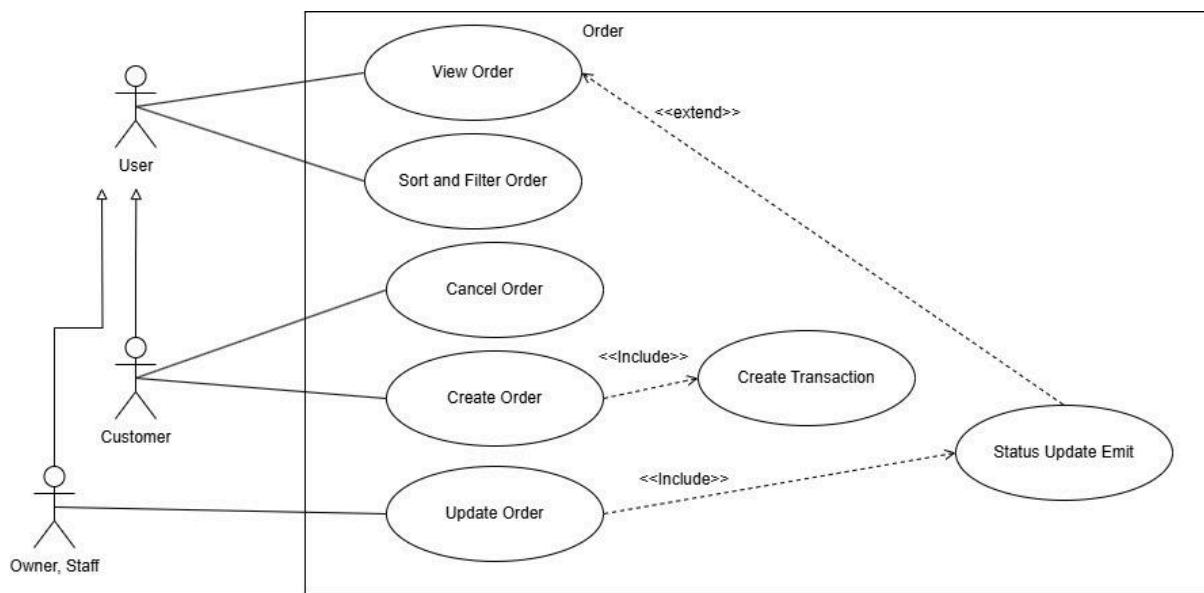


Figure 4.25: Order Use Case

Table 4.23: Create Order Use Case

Use case name: Create Order	
Pre-conditions: Logged in user	
Actor: Customer	
Use case description: This use case is to describe the action for Create the Order	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. User select all the ideal product 2. User proceed to Order	
	3. System validate the order 4. If the order is valid, then the order detail will be saved and proceed to payment gateway
5. user interact with the payment gateway for the payment	

	<ul style="list-style-type: none">6. The payment status and details will be returned from the payment gateway7. If the payment is successfully then will update the record in the order
8. User can select whether to proceed on order monitoring or back to main page	
	<ul style="list-style-type: none">9. If user select to proceed with order monitoring, the order status will be update in real time
<p>Alternative Flow Event</p>	
<p>A1: Step 4. If the order is invalid, it will show the error message. Return to step 1</p> <p>A2: Step 7. If the payment is not successfully conducted, it will return to the step 2</p> <p>A3: Step 9, if the user click to direct back to main page, System will direct user back to main page</p>	

Table 4.24: Update Order Use Case

Use case name: Update Order	
Pre-conditions: Logged in user	
Actor: food stall owner, food stall staff	
Use case description: This use case is to describe the action for updating the order status.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Order” from navigation bar	
	2. System direct the user to the Order list page
3. User click “Update” Button beside the ideally update record	
	4. The system will update the order into next phases. then emit the status for real time tracking.
Alternative Flow Event	
N/A	

Table 4.25: List order Use Case

Use case name: List Order	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for List the order records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Order” from the navigation bar	
	2. System direct the user to the Order list page
Alternative Flow Event	
N/A	

Table 4.26: Sort and Filter Order Use Case

Use case name: Sort and filter Order	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the order records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Order” from the navigation bar	
	2. System direct the user to the Order list page
3. User input the sort and filter options	
	4. System based on the criteria to do the sorting and filtering 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.27: View order Use Case

Use case name: View Order	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for View the order records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Order” from the navigation bar	
	2. System direct the user to the Order list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.28: Cancel order Use Case

Use case name: Cancel Order	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for canceling the order record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Order” from the navigation bar	
	2. System direct the user to the Order list page
3. User select the “Cancel” button beside the ideally deleted record	
	4. System first validate whether the order is available to proceed to cancel 5. if valid, the System pop up the dialog for confirmation
6. The user select confirm	
	7. System determine whether the user select which button 8. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 5, if order not valid, then return to step 2	

A1: Step 8, if user select “Cancel”, there will be nothing happens then return to Step 2

4.3.6 Reward

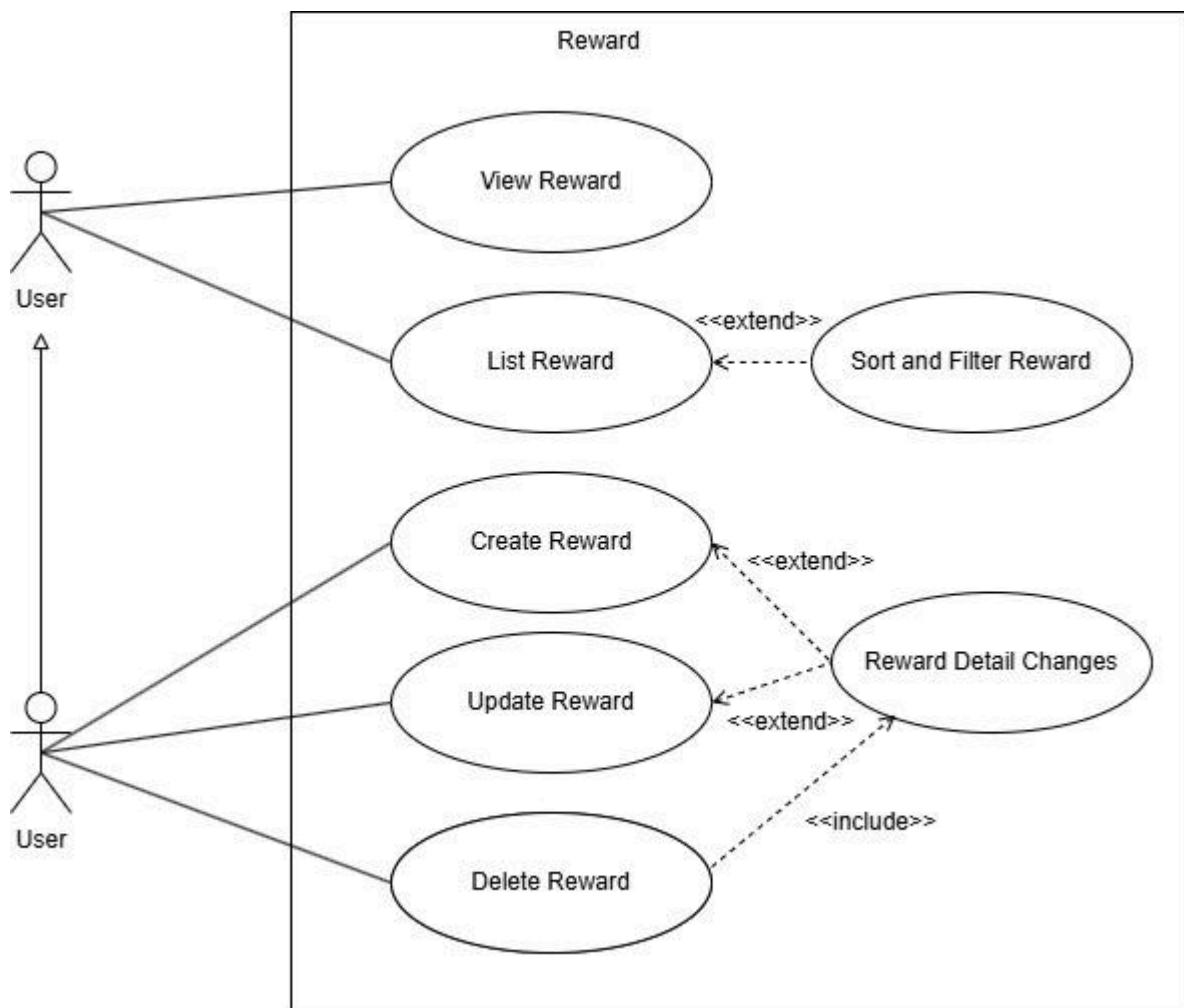


Figure 4.26: Reward Use Case

Table 4.29: Create reward Use Case

Use case name: Create Reward	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Create the reward	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	

	2. System direct the user to the Reward list page
3. User click “Add” Button	
	4. The system shows the reward creation form
5. User fill up the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 8. If the form is invalid, it will show the error message. Return to step 5	
A2: Step 11. If the user click the “Cancel” button, return to step 5	

Table 4.30:Edit Reward Use Case

Use case name: Edit Reward	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Editing the reward.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	
	2. System direct the user to the Reward list page
3. User click “Edit” Button beside the ideally edit record	
	4. The system shows the reward edit form with previously added record
5. User do editing the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.

Alternative Flow Event

A1: Step 8. If the form is invalid, it will show the error message. Return to step 5

A2: Step 11. If the user click the “Cancel” button, return to step 5

Table 4.31: List reward Use Case

Use case name: List Reward	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for List the reward records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	
	2. System direct the user to the Reward list page
Alternative Flow Event	
N/A	

Table 4.32: Sort and Filter Reward Use Case

Use case name: Sort and filter Reward	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the reward records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	
	2. System direct the user to the Reward list page
3. User input the sort and filter options	
	4. System based on the criteria to do the sorting and filtering 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.33: View Reward Use Case

Use case name: View Reward	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for View the reward records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	
	2. System direct the user to the Reward list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.34: Delete Reward Use Case

Use case name: Delete Reward	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for deleting the reward record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Reward” from the navigation bar	
	2. System direct the user to the Reward list page
3. User select the “Delete” button beside the ideally deleted record	
	4. System pop up the dialog for confirmation
5. The user select confirm	
	6. System determine whether the user select which button 7. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 7, if user select “Cancel”, there will be nothing happens then return to Step 2	

4.3.7 Announcement

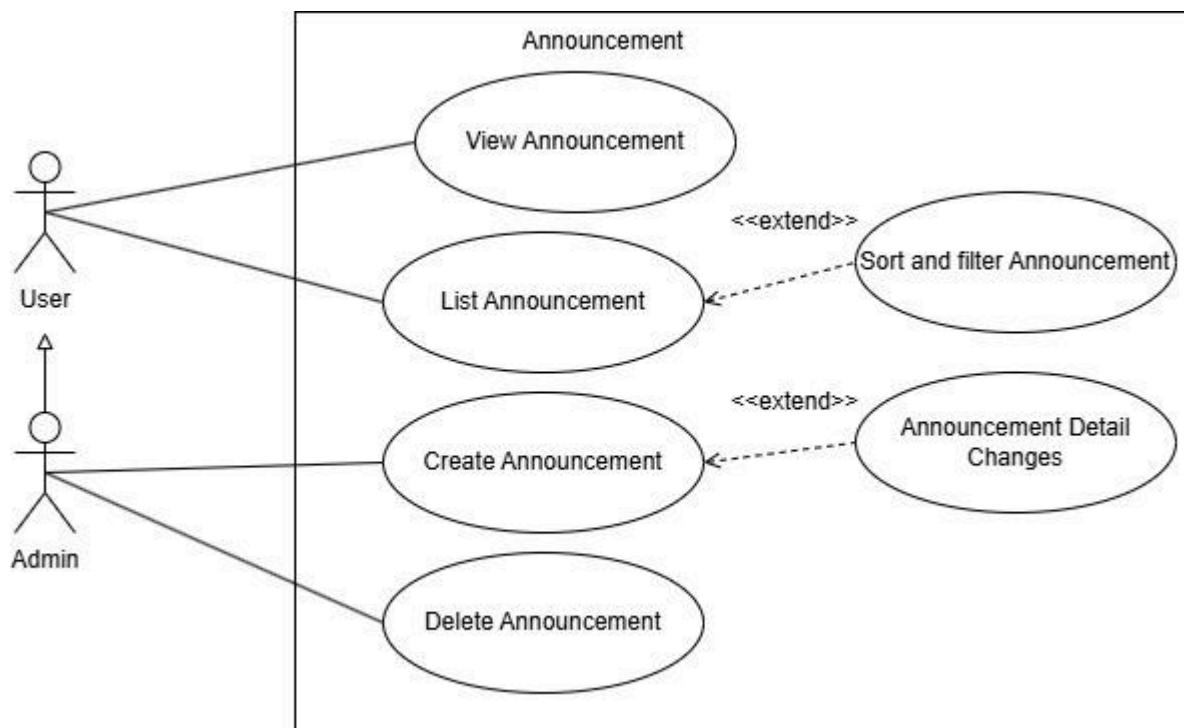


Figure 4.27: Announcement Use Case

Table 4.35: Create Announcement Use Case

Use case name: Create Announcement	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Create the Announcement	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Announcement” from the navigation bar	
	2. System direct the user to the Announcement list page
3. User click “Add” Button	

	4. The system shows the announcement creation form
5. User fill up the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 8. If the form is invalid, it will show the error message. Return to step 5 A2: Step 11. If the user click the “Cancel” button, return to step 5	

Table 4.36: List Announcement Use Case

Use case name: List Announcement	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for List the announcement records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Announcement” from the navigation bar	
	2. System direct the user to the Announcement list page
Alternative Flow Event	
N/A	

Table 4.37: Sort and Filter Announcement Use Case

Use case name: Sort and filter Announcement	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the announcement records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Announcement” from the navigation bar	
	2. System direct the user to the Announcement list page
3. User input the sort and filter options	
	4. System based on the criteria to do the sorting and filtering 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.38: View Announcement Use Case

Use case name: View announcement	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for View the announcement records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Announcement” from the navigation bar	
	2. System direct the user to the Announcement list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.39: Delete Announcement Use Case

Use case name: Delete Announcement	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for deleting the announcement record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Announcement” from the navigation bar	
	2. System direct the user to the Announcement list page
3. User select the “Delete” button beside the ideally deleted record	
	4. System pop up the dialog for confirmation
5. The user select confirm	
	6. System determine whether the user select which button 7. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 7, if user select “Cancel”, there will be nothing happens then return to Step 2	

4.3.9 Staff

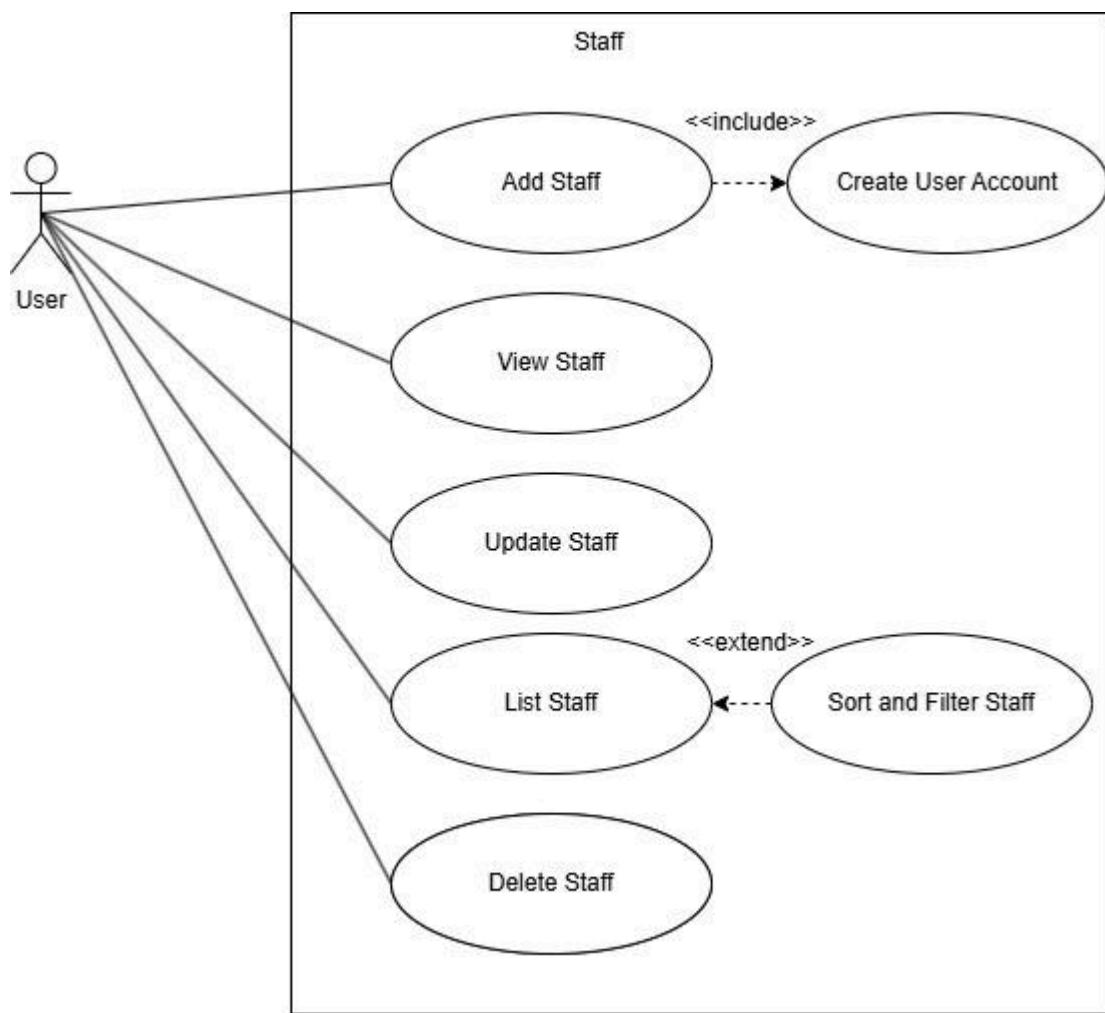


Figure 4.28: Staff Use Case

Table 4.40: Create Staff Use Case

Use case name: Create Staff	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Create the staff	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	

	2. System direct the user to the Staff list page
3. User click “Add” Button	
	4. The system shows the staff creation form
5. User fill up the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.
Alternative Flow Event	
A1: Step 8. If the form is invalid, it will show the error message. Return to step 5	
A2: Step 11. If the user click the “Cancel” button, return to step 5	

Table 4.41: Edit Staff Use Case

Use case name: Edit Staff	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for Editing the staff.	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	
	2. System direct the user to the Staff list page
3. User click “Edit” Button beside the ideally edit record	
	4. The system shows the staff edit form with previously added record
5. User do editing the form accordingly 6. User click the “Save” button	
	7. Validate the submitted form 8. If the form is valid, shows the confirmation dialog
9. User do selection from the confirmation	
	10. System validate the user confirmation selection 11. If the user clicks confirm, then the record will be saved into the database. Then shows the success message.

Alternative Flow Event

A1: Step 8. If the form is invalid, it will show the error message. Return to step 5

A2: Step 11. If the user click the “Cancel” button, return to step 5

Table 4.42: List Staff Use Case

Use case name: List Staff	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for List the staff records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	
	2. System direct the user to the Staff list page
Alternative Flow Event	
N/A	

Table 4.43: Sort and filter staff Use Case

Use case name: Sort and filter Staff	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the staff records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	
	2. System direct the user to the Staff list page
3. User input the sort and filter options	
	4. System based on the criteria to do the sorting and filtering 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.44: View Staff Use Case

Use case name: View Staff	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for View the staff records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	
	2. System direct the user to the Staff list page
3. User select the ‘View’ icon beside the ideally view details record	
	4. System based on the the record id to find the record 5. The record details will be showed in form format
Alternative Flow Event	
N/A	

Table 4.45: Delete Staff Use Case

Use case name: Delete Staff	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for deleting the staff record.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Staff” from the navigation bar	
	2. System direct the user to the Staff list page
3. User select the “Delete” button beside the ideally deleted record	
	4. System pop up the dialog for confirmation
5. The user select confirm	
	6. System determine whether the user select which button 7. If user select “Confirm”, then proceed to delete the record in database and provide the successful message
Alternative Flow Event	
A1: Step 7, if user select “Cancel”, there will be nothing happens then return to Step 2	

4.3.10 Report

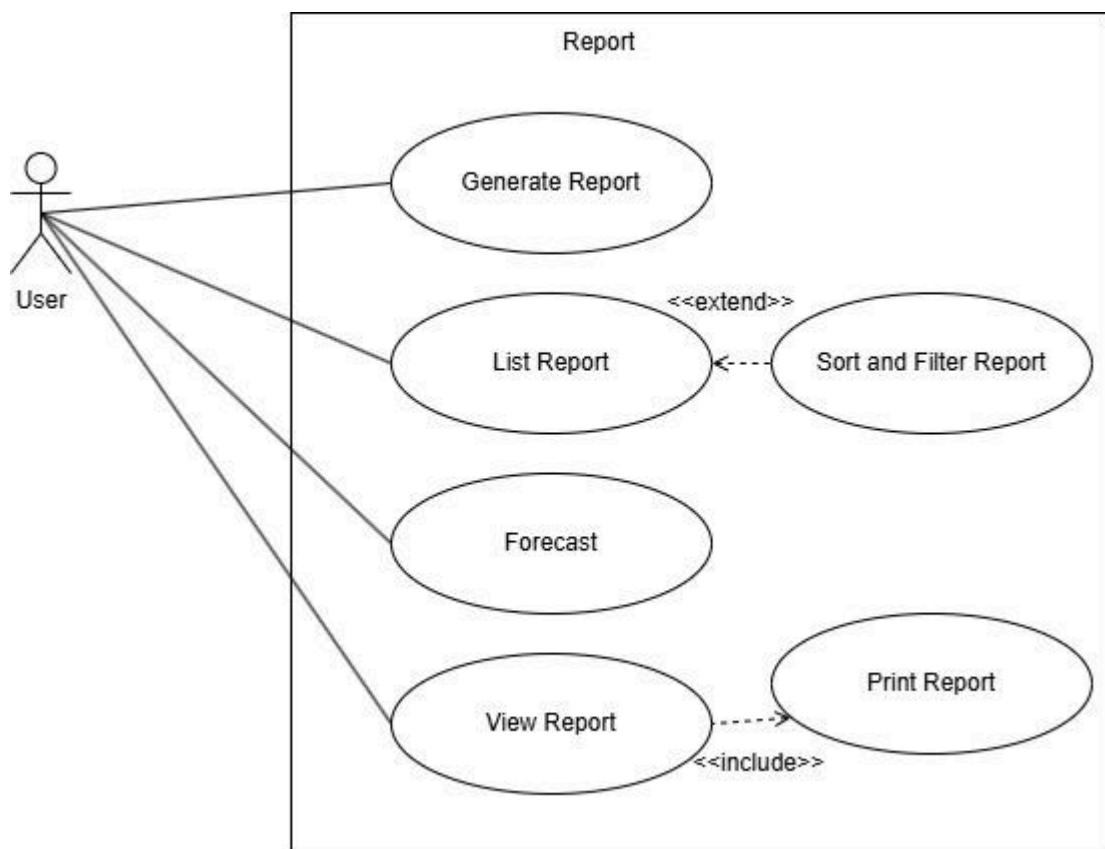


Figure 4.29: Report Use Case

Table 4.46: Generate Report Use Case

Use case name: Generate Report	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for generating the report.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Report” from the navigation bar	
	2. System display the report list page
3. Fill up the criteria accordingly	

	4. System will validate the criteria 5. if valid, System will based on the criteria to do the listing
6. User click "Generate"	
	7. System based on the criteria input to generate a new report
Alternative Flow Event	
A1: Step 5, if the criteria is invalid, display the error message. Return Step 3	

Table 4.47: View Report Use Case

Use case name: View Report	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for viewing the report.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Report” from the navigation bar	
	2. System display the report list page
3. User click the “View” button beside the ideally view report	
	4. The system will search for the document in system path 5. If searched, the document will be downloaded to the local machine then show the success message
Alternative Flow Event	
A1: Step 5, if not found, display the error message. Return Step 2	

Table 4.48: List Report Use Case

Use case name: List Report	
Pre-conditions: Logged in user	
Actor: Admin	
Use case description: This use case is to describe the action for generating the report.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Report” from the navigation bar	
	2. System display the report list page
Alternative Flow Event	
N/A	

Table 4.49: Sort and Filter report Use Case

Use case name: Sort and filter Report	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for sort and filter listing the report records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Report” from the navigation bar	
	2. System direct the user to the Report list page
3. User input the the criteria of the report accordingly	
	4. System based on the criteria to do the sorting and filtering on report records 5. If record(s) exist, System display the result to user
Alternative Flow Event	
A1: Step 5. If no record exists, return the message “No record found” on the list page.	

Table 4.50: Forecast Report Use Case

Use case name: Forecast Report	
Pre-conditions: Logged in user	
Actor: User	
Use case description: This use case is to describe the action for forecast the report records.	
Main Flow Event/ Basic flow	
Actor Action	System Response
1. The user click the “Report” from the navigation bar	
	2. System direct the user to the Report list page
3. User click the ‘Forecast’ button beside the ideally being forecast report	
	4. The system will determine the report can be forecasted 5. If can, the system will forecast the result in the document then the result will be downloaded to the local machine
Alternative Flow Event	
A1: Step 5. If the report can't be forecasted, return error message, return step 2	

4.4 Behavioral Models(State Diagrams)

4.4.1 User Account

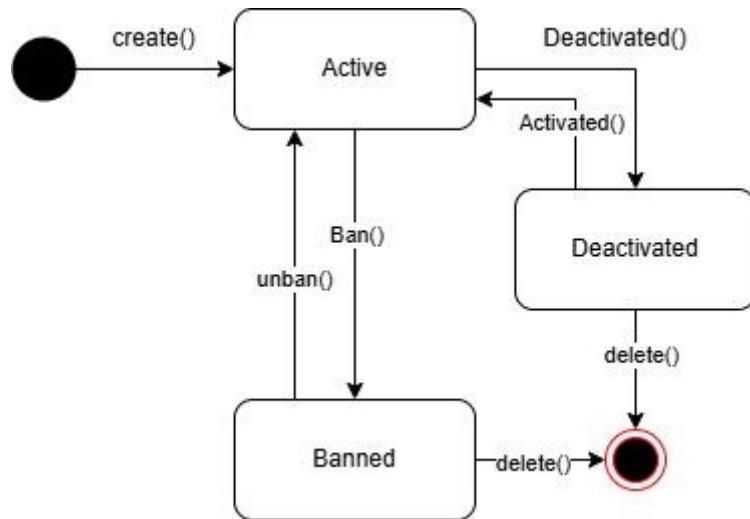


Figure 4.30: User Account State Diagram

4.4.2 Product

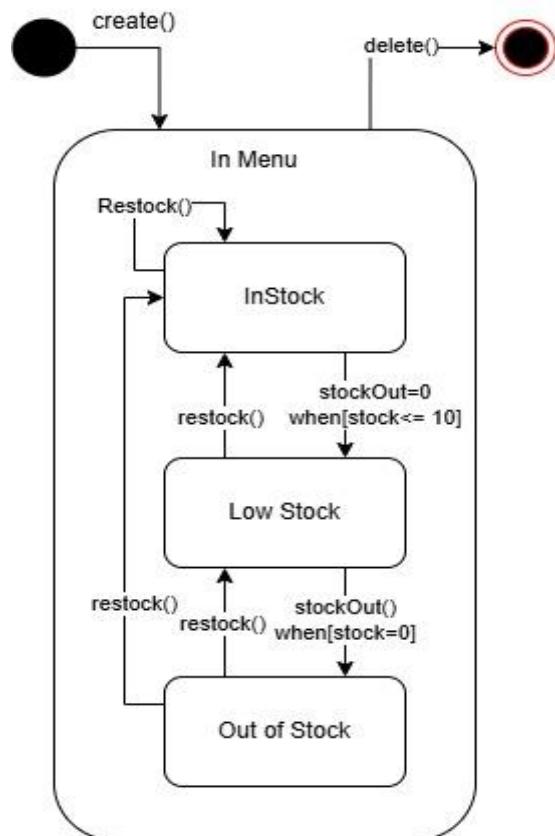


Figure 4.31: Product State Diagram

4.4.3 Order

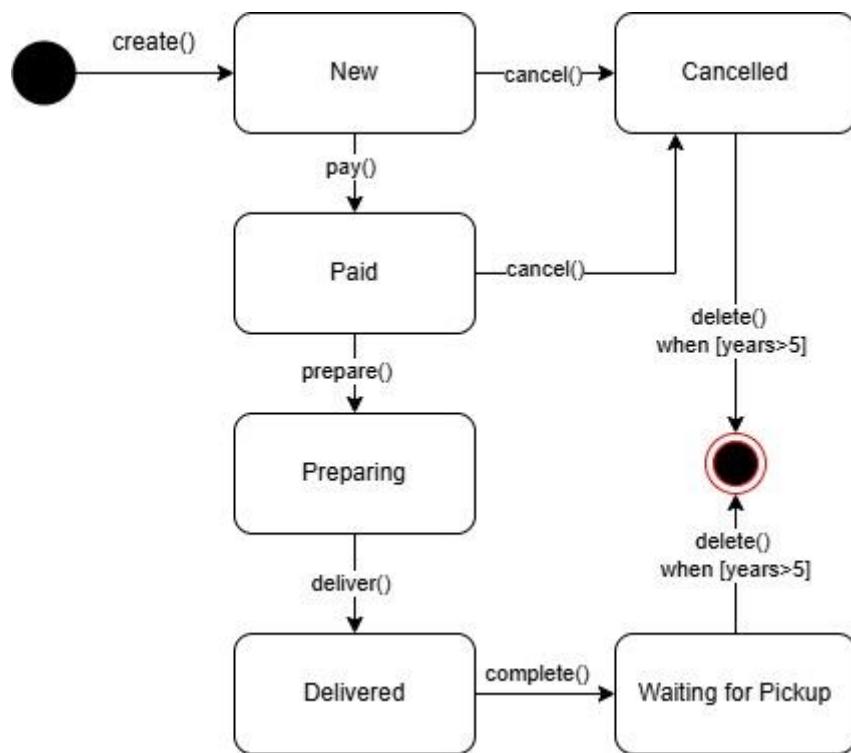


Figure 4.32: Order State Diagram

4.4.4 Reward

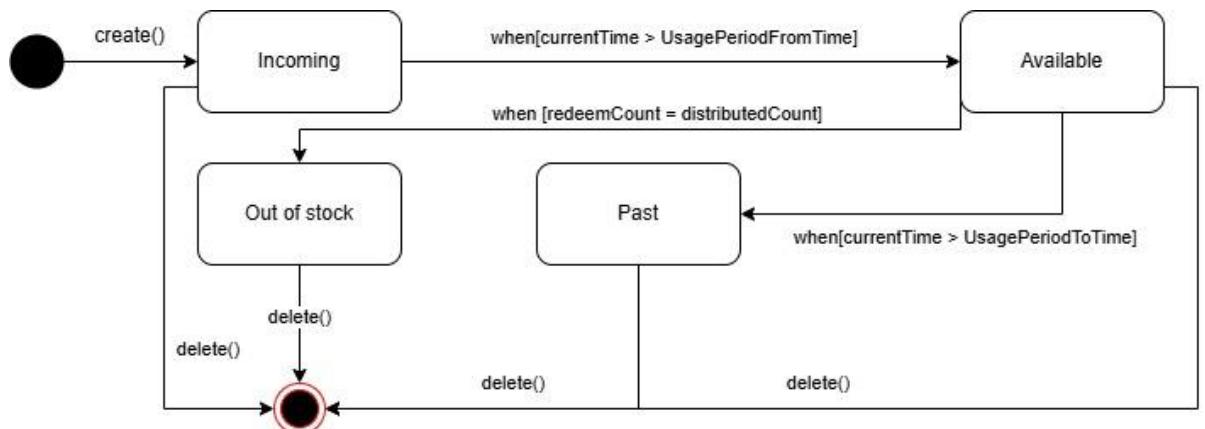


Figure 4.33: Reward State Diagram

4.5 Other designs (Architectural design & UI design)

4.5.1 Architecture Design

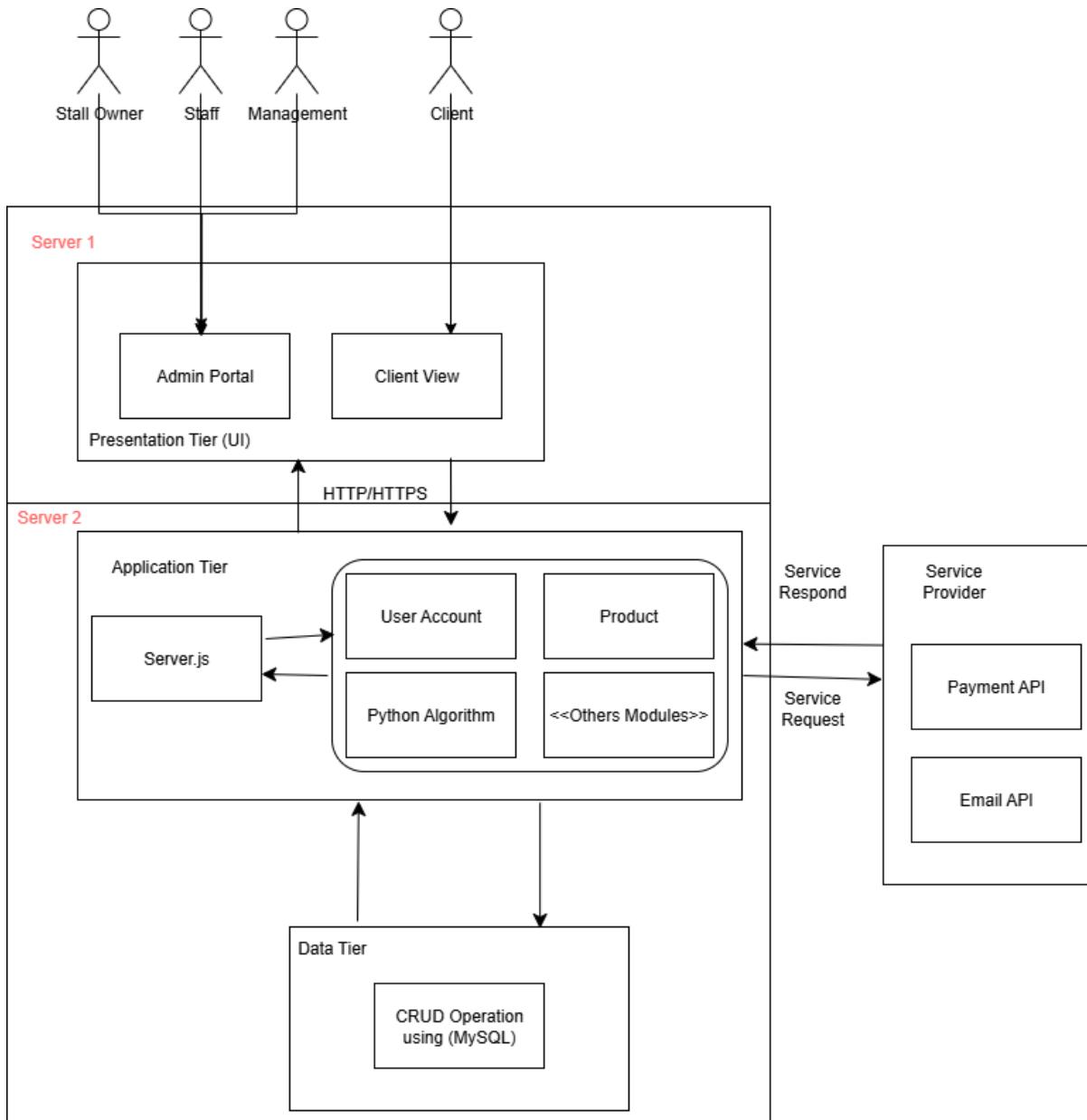


Figure 4.34: System Architecture with communication protocol

Based on Chapter 3 already considered for the system architectural design, this section will provide more information about the communication protocol used between each tier. The communication protocol for this project is HTTP/HTTPS.

The communication protocol between the Client (users like Stall Owner, Staff, Management, and Client) and the UI (Presentation Tier) is the clients interact with the system through a web browser or application, sending HTTP/HTTPS requests to the Presentation Tier (React frontend). These requests retrieve data and update the UI dynamically based on user actions.

Using HTTPS ensures secure communication by encrypting data, protecting it from interception and tampering.

This protocol also is used for communication between the Presentation Tier (React frontend) and the Application Tier (Express.js backend). When a user interacts with the UI, HTTP requests such as GET, POST, PUT, and DELETE are sent to the server (Server.js). The backend processes these requests and responds with the necessary data, often fetched from the Data Tier (PostgreSQL database). Additionally, HTTP/HTTPS is also used to interact with external services like the Payment API in your system. Figure above will be the final showing of the system architectural design. Using HTTPS ensures data encryption, protecting user credentials and sensitive information from interception. Additionally, WebSockets may be used for real-time updates if the system requires features like live notifications or instant data synchronization.

4.5.2 Preliminary Interface Design

This section presents the preliminary sketches for the User Interface design, which will serve as a reference for the more detailed design to be discussed in the following chapter.

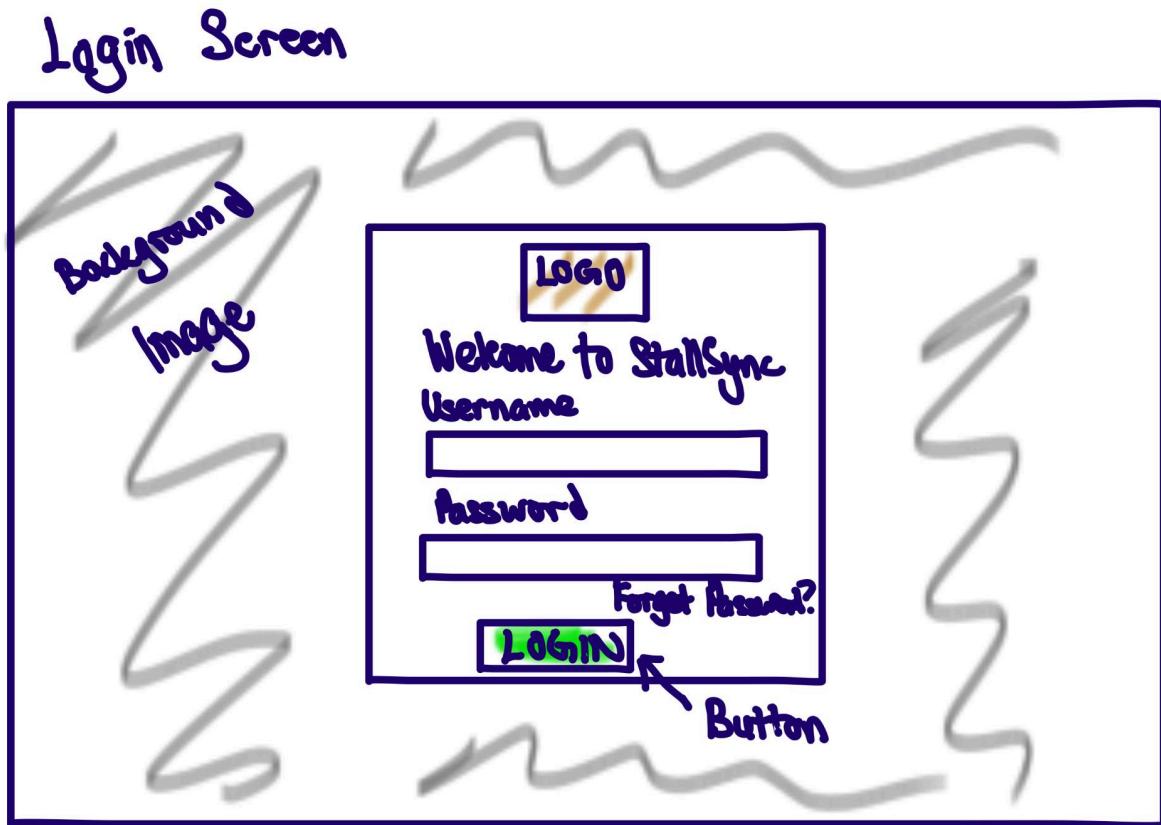


Figure 4.35: Login Screen Preliminary Interface Design

According to Figure 4.5.1, there will be the background image and the logo will be placed on the screen. Users can only use the username and password to login into the admin system. The “forgot password?” is served as the password recovery function.

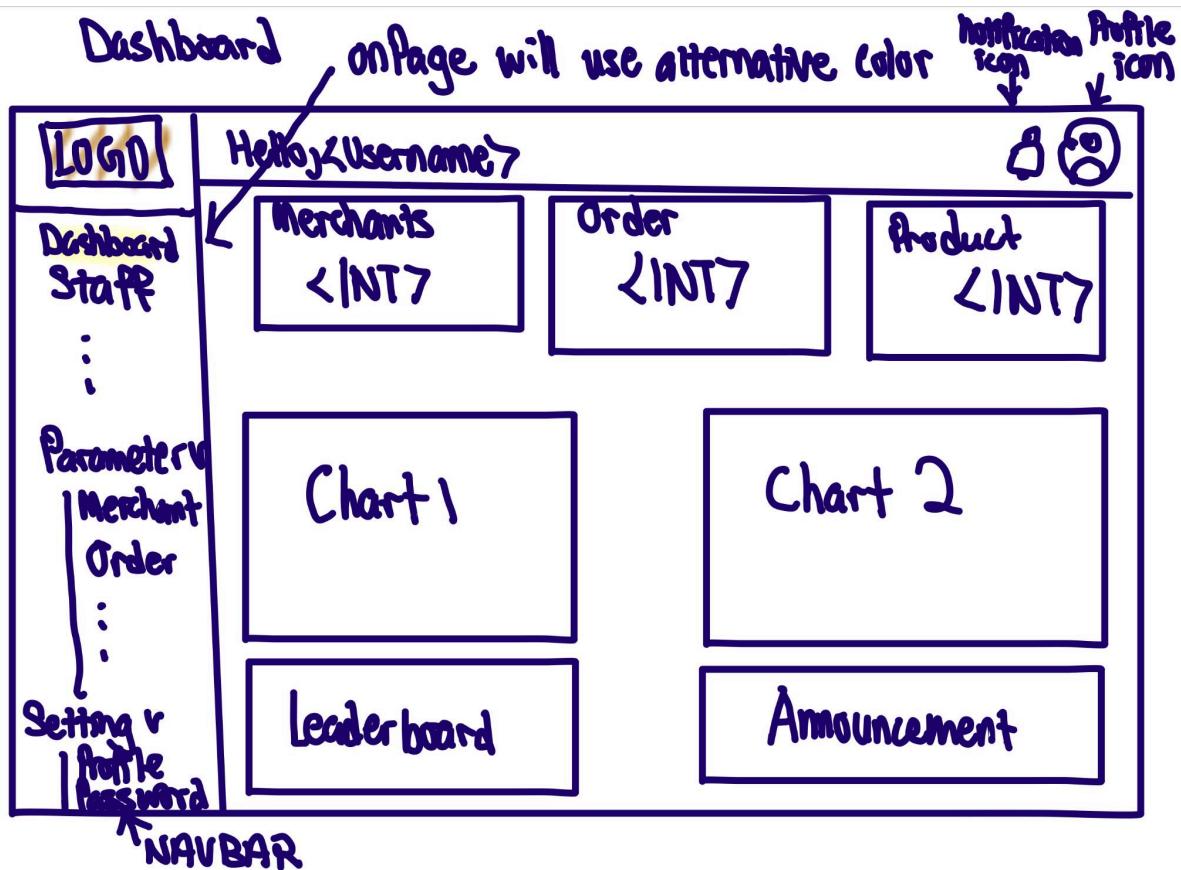


Figure 4.36: Dashboard Preliminary Interface Design

According to figure 4.5.2, this is the Stall Sync admin portal dashboard. It will include the summarized info for example the statistic chart, the counts and more. It is used to provide the overview to the system although there is a Report module. When the user is on “Dashboard” screen, the navigation bar “Dashboard” will use the alternative color to identify where the user is at.

List Page Example

The hand-drawn sketch illustrates a preliminary interface design for a 'Staff' listing page. The layout is as follows:

- Header:** A logo placeholder, a greeting "Hello <Username>" with a user icon, and a green "ADD" button.
- Breadcrumbs:** "Staff" and "All Staff".
- Table:** A grid showing staff details with columns: STAFFID, NAME, Gmail, and ACTIONS. The data is as follows:

STAFFID	NAME	Gmail	ACTIONS
Staff1	Kenny	k123@g.com	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Staff2	Benny	b456@g.com	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
Staff3	Jenny	j789@g.com	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
- Actions:** Below the table, there are three buttons labeled "VIEW", "EDIT", and "DEL" with upward arrows pointing to their respective columns in the table.
- Pagination:** Navigation controls including '<' and '>', a current page indicator "1/10", and "Pages" text.
- Left Sidebar:** A vertical sidebar with handwritten notes: "Dashboard", "Staff", ":", "Parameter 1", and "Setting 1".

Figure 4.37: List Page Preliminary Interface Design

According to figure 4.5.3, this is a listing page example from “Staff”, the first we can see is the title of the screen and followed by the breadcrumbs which are known as the routes. Second, there will be an “Add” button to link the user for creation of a new record. In the “ACTIONS” column, there will be the “View”, “Edit”, and “Delete” button in sequence. The “Add”, “View” and “Update” will use the same form page but with different validation and disable may be applied.

Form Page Example

The form page example is a hand-drawn sketch of a mobile application interface. It features a header with a logo and a greeting "Hello, {Username}!". Below the header is a sidebar with links like "Dashboard", "Staff", and "Parameter Setting". The main content area is titled "Staff" and "Staff / Staff (Add)". It contains several input fields: "StaffID" (with placeholder "Enter StaffID"), "StaffName" (placeholder "Enter name"), "Gmail" (placeholder "Enter gmail"), "dob" (placeholder "dd/mm/yyyy" with a note "Date Picker" pointing to it), "Type" (placeholder "Select Type" with a dropdown arrow), and "Status" (a switch button set to "YES"). At the top right of the content area are "BACK" and "SAVE" buttons.

Figure 4.38: Form Page Preliminary Interface Design

According to Figure 4.5.4, this is the example of the form page from Staff. There are different field styles that will be showing which are very common that will be used. The “BACK” button is used to navigate to the previous screen, whereas the “SAVE” button will be used to save records that are being created or updated.

4.5.3 Detailed Interface Design

In this section, it will base on the previous section to create a better interface design which can be used as the reference on real development.

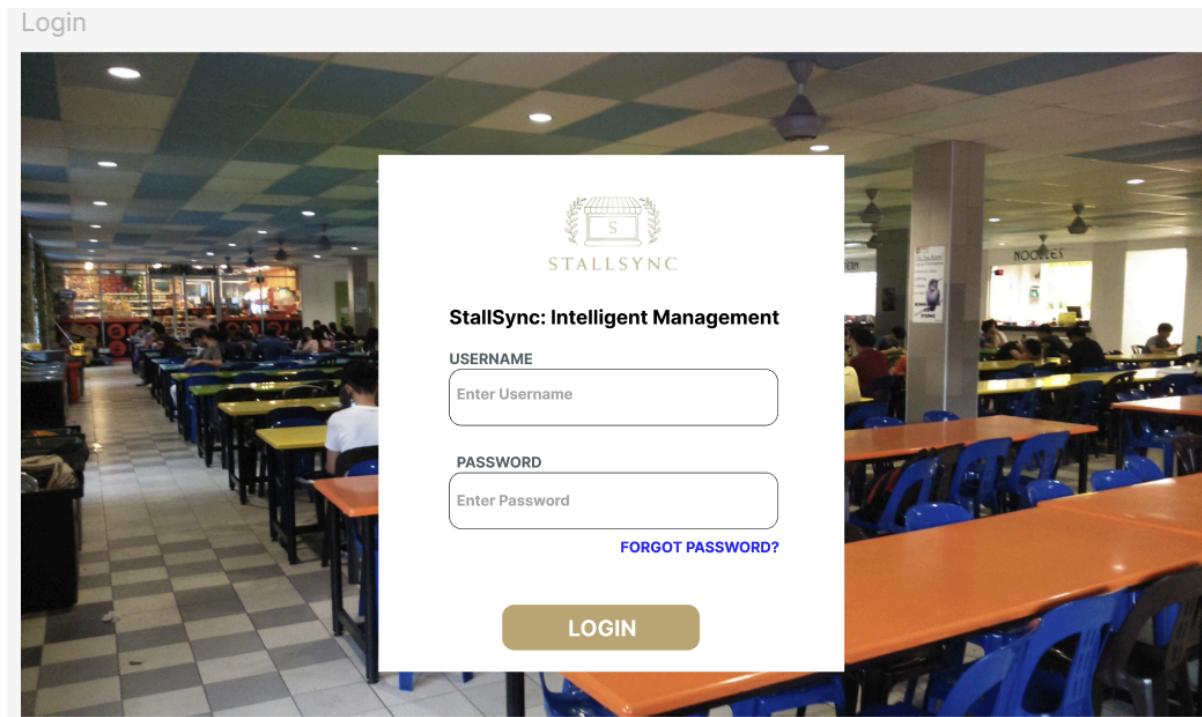


Figure 4.39: Login Page

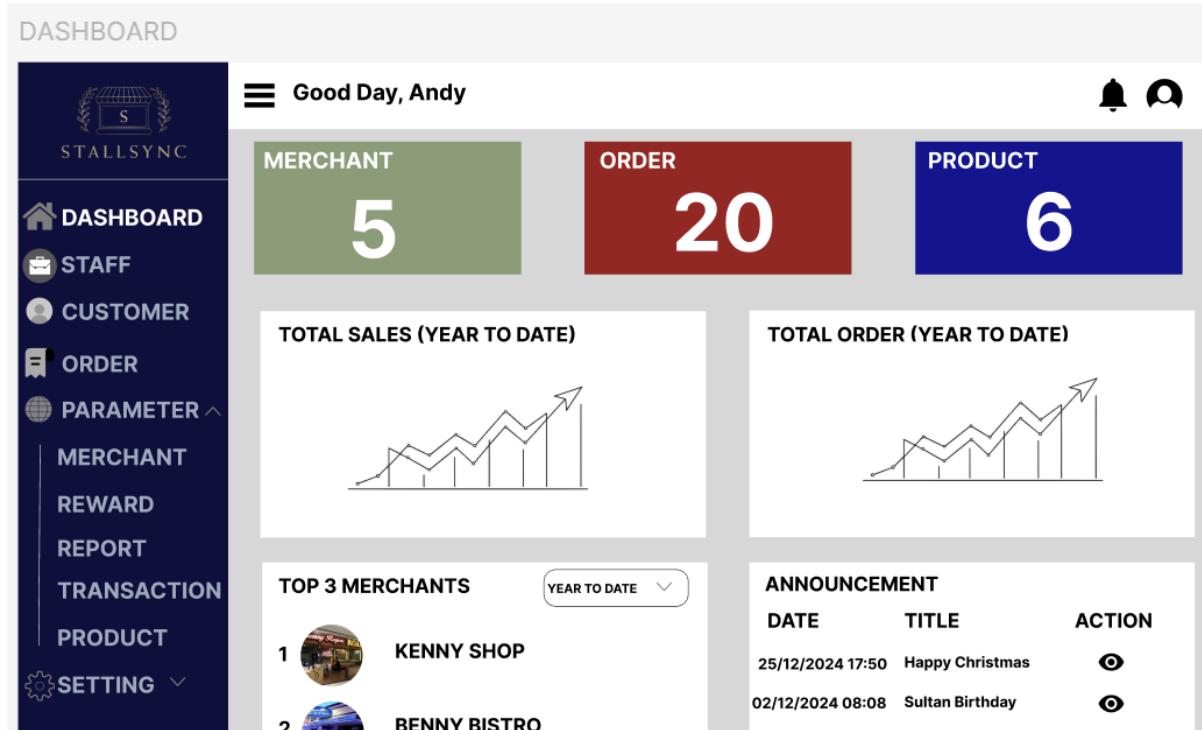


Figure 4.40: Dashboard

STAFF

STAFF ID	STAFF NAME	STAFF EMAIL	HP NO.	USER TYPE	STATUS	ACTIONS
S001	Andy Enderson	andy@gmail.com	+60123456789	Admin	Yes	
S002	Alysson Keth	Alyketh@gmail.com	+86788733333	Owner	No	
S003	Jessie Parth	Jandparth@gmail.com	+60198765432	Admin	Yes	

Show total 1-3 entries of 3 < 1 > 10/Page

Figure 4.41: Staff Page

PRODUCT

PRODUCT ID	PRODUCT NAME	DESCRIPTION	MERCHANT ID	PRICE (RM)	STATUS	ACTIONS
BB001	Nasi Lemak	Serve with Sambal	BNYBST	5.00	Low Stock	
KS001	Mac & Cheese	Serve with Cheese	KNYSHP	10.00	Available	
BB002	Tosai	Best in town	BNYBST	3.20	Out of Stock	

Show total 1-3 entries of 3 < 1 > 10/Page

Figure 4.42: Product Page

Order

Good Day, Andy

Order H/Order

DATE	CUSTOMER NAME	CONTACT	ORDER AMOUNT	STATUS	ACTIONS
25/12/2024 17:50	KENNY J	+60123456789	RM 89.90	CANCELLED	
25/12/2024 17:49	KUMAR NATHAN	+86788733333	RM 20.70	PAID	
25/12/2024 17:30	FAIZUL MAZLAN	+60198765432	RM 15.90	COMPLETED	

Show total 1-3 entries of 3 < 1 > 10/Page

DASHBOARD STAFF CUSTOMER ORDER PARAMETER MERCHANT REWARD REPORT TRANSACTION PRODUCT SETTING

Figure 4.43: Order Page

Merchant

Good Day, Andy

Merchant H/Merchant

ADD

MERCHANT ID	MERCHANT NAME	CONTACT	JOIN DATE	OWNER	STATUS	ACTIONS
KNYSHP	Kenny Shop	+60123456789	25/12/2014	S002	Yes	
BNYBST	Benny Bistro	+86788733333	25/12/2015	S005	No	
SSMZ	Sushi MingZ	+60198765432	25/12/2016	S008	Yes	

Show total 1-3 entries of 3 < 1 > 10/Page

DASHBOARD STAFF CUSTOMER ORDER PARAMETER MERCHANT REWARD REPORT TRANSACTION PRODUCT SETTING

Figure 4.44: Merchant Page

REWARD ID	REWARD NAME	USAGE PERIOD FROM	TO	QUANTITY	STATUS	ACTIONS
FRIDAY	TGIF Voucher	25/12/2024 17:50	25/01/2024 17:50	10	Incoming	
GAZ	Get Urs	25/12/2024 17:50	25/12/2024 18:50	0	Out Of Stock	
RAYA	Hari Raya	01/04/2024 17:50	31/04/2024 17:50	100	Past	

Show total 1-3 entries of 3 < 1 > [10/Page]

Figure 4.45: Reward Page

DATE	REPORT NAME	STATUS	ACTIONS
25/12/2024 17:50	SalesReport-Jan2024_May2024.xlsx	Yes	
25/12/2024 17:50	SalesReport-Jan2024_May2024.xlsx	Yes	
01/04/2024 17:50	OrderReport-Jan2024_May2024.xlsx	Yes	

Show total 1-3 entries of 3 < 1 > [10/Page]

Figure 4.46: Report Page

The screenshot shows the 'User Account' form page from the StallSync application. At the top right, there is a user greeting 'Good Day, Andy' and icons for notifications and messages. On the far right, there is a green 'SAVE' button. The main content area is titled 'User Account' and includes the subtitle 'H/User Account'. It contains four input fields: 'USERNAME' (placeholder 'Enter Username'), 'NAME' (placeholder 'Enter Name'), 'CONTACT NO.' (placeholder 'Enter Contact No.'), and 'EMAIL' (placeholder 'Enter Email'). Below these fields is a dropdown menu labeled 'USER TYPE' with the option 'Select User Type'. On the left side of the screen, there is a vertical sidebar with the 'STALLSYNC' logo at the top. Below the logo, the sidebar lists several menu items with corresponding icons: 'DASHBOARD' (house icon), 'STAFF' (person icon), 'CUSTOMER' (person icon), 'ORDER' (receipt icon), 'PARAMETER' (globe icon), 'SETTING' (gear icon), and 'MY PROFILE' (person icon). Under 'MY PROFILE', there is a link 'CHANGE PASSWORD'. The background of the main content area is light gray.

Figure 4.47: Example Form Page

4.6 Chapter Summary and Evaluation

This chapter uses several diagrams to explain the StallSync: Intelligent Management for Food Stalls from different angles. The system design follows object-oriented principles and adopts a 3-tier architecture. A UML class diagram shows how users interact with the system. Activity diagram outlines the steps involved in how to use the AI (time-series forecasting). A use case diagram and use case descriptions provide detailed flow of the system's processes, including alternative flows when specific conditions occur. These functions are organized into separate modules. An Entity-Relationship Diagram (ERD) and a data dictionary describe how data is structured and outline the system's data requirements. A basic GUI design is also included. It offers a preview of how users will interact with the system in practice. Together, these diagrams provide a clear view of the system's design and behavior. This understanding will help ensure a smooth and successful implementation in the next phase.

Chapter 5

Implementation and Testing

5 Implementation and Testing

The StallSync system is designed for merchants to manage food orders online. It uses modern web technologies. The core stack includes React.js, Next.js, Node.js, and MySQL. React.js helps build the user interface. It offers reusable components and keeps the codebase organized. Next.js improves performance. It uses server-side rendering and built-in routing. This also helps with search engine visibility. The backend runs on Node.js. It handles asynchronous tasks and input/output operations. Express.js is used for routing and middleware. The system uses MySQL as the main database. Developers manage it using XAMPP during development. The frontend and backend connect through HTTP requests. These follow RESTful API standards. The system includes basic security features. These protect data and support safe communication between users and the server. BREVO handles email communication. It uses an API key for secure access. This feature is mainly used for password reset emails and system alerts. Testing plays a key role in development. The team uses black-box testing. This method checks what the system does, not how it's coded. Tests focus on key features used by merchants. The testing follows an incremental and bottom-up method. Each module is tested on its own. Then, modules are tested together as they are combined. This helps catch issues early. Test cases cover many situations. These include normal, incorrect, edge, and special input values. The testing environment includes specific requirements. It uses Windows 11, XAMPP, and a device with enough memory and processing power. Before testing starts, certain conditions must be met. These are called entry criteria. After testing ends, the team checks exit criteria. This ensures testing is complete, major bugs are fixed, and the timeline and budget are met. The whole process helps make sure the system works well. It confirms each feature meets the project's goals and user needs.

5.1 Implementation

5.1.1 User Authentication (Security)

The User Authentication section of StallSync is designed to ensure secure login for system users through a carefully structured process involving validation, verification, and token management. When a login request is received, the system first validates the input credentials using a custom validation function to ensure that both the username and password are provided and meet the expected format. If validation fails, the system returns an error response to the client. Next, the backend checks whether the API key provided in the request header matches the key stored in the environment variables. This acts as a security gateway to block unauthorized access from external sources.

Once the API key is verified, the system searches for the user in the psusrprf table by matching the provided username. If a matching user is found, the system checks the user's account status. If the account is locked (L), closed (C), or expired (E), the system immediately denies access and responds with the appropriate error message. If the account is active, the system uses the bcrypt library to compare the input password with the hashed password stored in the database. If the passwords match, the system proceeds to generate a JSON Web Token (JWT) that encapsulates the user ID, username, and login timestamp as the payload.

Before issuing the token, the system initiates a database transaction to ensure atomicity for subsequent operations. It then checks the usrlgnpf table to see if a login record for the user already exists. If it does, the record is updated with the new login status, timestamp, and token. If not, a new record is created. The system also updates the user's psusrprf record to indicate that the first login has occurred and optionally stores a push notification token if provided. Once all updates are successfully completed, the transaction is committed to the database. Finally, the JWT is returned in a "Bearer" format, allowing the user to authenticate future requests. If any error occurs during this process, appropriate error handling and transaction rollback ensure system stability and data integrity. This implementation demonstrates robust security practices through input validation, hashed password comparison, token-based authentication, and transactional consistency.

Example Code Snippet:

```

// User Login
exports.login = async (req,res) =>{
    const { errors, isValid } = validateLoginInput(req.body);
    if (!isValid) {
        return returnError(req, 400, errors, res);
    }

    if (req.headers["api-key"] != process.env.API_KEY) return returnError(req, 500, "INVALIDKEY", res);

    const username = req.body.username;
    const password = req.body.password;
    await psusrprf.findOne({
        where: {
            psusrunm: username
        }, raw: true
    }).then(async data => {
        if (data) {
            if (data.psusrsts == 'L') return returnError(req, 400, { username: 'ACCOUNTLOCKED' }, res);
            else if (data.psusrsts == 'C') return returnError(req, 400, { username: 'ACCOUNTCLOSED' }, res);
            else if (data.psusrsts == 'E') return returnError(req, 400, { username: 'MEMBEREXPIRED' }, res);

            bcrypt.compare(password, data.psusrpwd).then(async isMatch => {
                if (isMatch) {
                    const t = await connection.sequelize.transaction();

                    let payload = { id: data.id, psusrunm: data.psusrunm, iat: Date.now() };

                    jwt.sign(
                        payload,
                        "secret",
                        async (err, token) => {
                            // Write / Update into Login Table - SSO Purpose
                            await usrlgnpf.findOne({
                                where: {
                                    psusrunm: data.psusrunm
                                }, raw: true
                            }).then(lgnpf => {
                                if (lgnpf) {
                                    let options = {
                                        pslgnsts: true,
                                        psigidat: new Date(),
                                        pslgntkn: "Bearer " + token
                                    };
                                    if (!lgnpf.pslgnsts) {
                                        options.psigidat = new Date();
                                    }
                                    usrlgnpf.update(options, {
                                        transaction: t,
                                        where: {
                                            id: lgnpf.id
                                        }
                                    }).catch(err => {
                                        console.log(err);
                                        t.rollback();
                                        return returnError(req, 500, "UNEXPECTEDERROR", res);
                                    });
                                } else {
                                    usrlgnpf.create({
                                        psusrunm: data.psusrunm,
                                        pslgnsts: true,
                                        psigidat: new Date(),
                                        pslgntkn: "Bearer " + token
                                    }, {
                                        transaction: t,
                                    }).catch(err => {
                                        console.log(err);
                                        t.rollback();
                                        return returnError(req, 500, "UNEXPECTEDERROR", res);
                                    });
                                }
                            });
                        }
                    );
                    // Update First Login Flag
                    await psusrprf.update({
                        psfstlg: false,
                        psapptkn: req.body.push_token ? req.body.push_token : data.psapptkn
                    }, {
                        transaction: t,
                        where: {
                            id: data.id
                        }
                    }).catch(err => {
                        console.log(err);
                        t.rollback();
                        return returnError(req, 500, "UNEXPECTEDERROR", res);
                    });

                    t.commit();
                    return returnSuccess(200, { token: "Bearer " + token }, res);
                }
            });
        } else return returnError(req, 400, { password: "INCORRECTPASS" }, res);
    });
}

else return returnError(req, 400, { username: "USERNOTFOUND" }, res);
}).catch(err => {
    console.log(err);
    return returnError(req, 500, "UNEXPECTEDERROR", res);
});
}

```

Figure 5.1: Code Snippet for User Authentication

5.1.2: Success/Error Response

The Success/Error Response Consistency module in StallSync is designed to standardize how the backend handles and returns both success and error responses. The returnError function dynamically constructs error messages based on HTTP status, localization preferences (supporting English and Chinese), and specific error codes. It processes input codes using logic that replaces placeholders (e.g., &length or #field) with actual values or field names defined in external constants. When a validation error occurs (e.g., status 400), it logs the error into the errlogpf table, capturing the API path, error details, and request body for audit and debugging. If a specialCode and specialMessage are provided, it directly returns a custom error structure. Additionally, for generic errors, it matches the error code against localized message dictionaries and logs them accordingly. The function also supports writing failure logs to a history system if a reference ID is provided. On the other hand, the returnSuccess function ensures all successful responses follow a consistent format by wrapping the returned data in a JSON object with a result: 'success' flag and a message payload. This design ensures frontend developers can reliably parse all responses, improving maintainability, debugging, and user feedback consistency.

Example Code Snippet for Success Response:

```
module.exports = function returnSuccess(status,data,res){
    const success = {};
    success.result = 'success';
    success.message = data;
    return res.status(status).json(success); // send the error response to client
};
```

Figure 5.2: Code Snippet for Success Response

Example Code Snippet for Error Response:

```

module.exports = function returnError(req, status, code, res, specialCode, specialMessage, extra, ref, api) {
    const errors = {};
    if (req.headers['locale'] !== '' && req.headers['locale'] !== undefined && req.headers['locale'] !== null) {
        var locale = req.headers['locale'].toLowerCase();
    } else {
        var locale = 'en';
    }
    if (specialCode && specialCode != '' && specialMessage && specialMessage != '') {
        errors.result = 'error';
        errors.code = specialCode;
        errors.message = [
            error: specialMessage
        ];
        errors.extra = extra;
        return res.status(status).json(errors); // send the error response to client
    } else {
        if (status == '400' || status == 400) {
            let errorpf = {
                api: req.originalUrl,
                error_code: JSON.stringify(code),
                incoming: JSON.stringify(req.body),
                caller: req.user ? req.user.psusrnum : ''
            };
            Object.keys(code).forEach((key) => {
                let tempCode = code[key];
                let par = tempCode.indexOf('&') > 0 ? tempCode.split('&') : tempCode;
                let field = tempCode.indexOf('#') > 0 ? tempCode.split('#') : tempCode;
                if (code[key].indexOf('&') > 0) {
                    if (locale == 'en') {
                        if (errorMessage[par[0]]) code[key] = errorMessage[par[0]].replace('&length', par[1]);
                        else code[key];
                    } else {
                        if (errorMessageCN[par[0]]) code[key] = errorMessageCN[par[0]].replace('&length', par[1]);
                        else code[key];
                    }
                } else if (code[key].indexOf('#') > 0) {
                    if (locale == 'en') {
                        if (errorMessage[field[0]]) code[key] = errorMessage[field[0]].replace('#field', fieldNames[field[1]]);
                        else code[key];
                    } else {
                        if (errorMessageCN[field[0]]) code[key] = errorMessageCN[field[0]].replace('#field', fieldNames[field[1]]);
                        else code[key];
                    }
                } else {
                    if (locale == 'en') {
                        if (errorMessage[code[key]]) code[key] = errorMessage[code[key]];
                        else code[key];
                    } else {
                        if (errorMessageCN[code[key]]) code[key] = errorMessageCN[code[key]];
                        else code[key];
                    }
                }
            });
            errors.message = code;
            errorpf.error_desc = JSON.stringify(code);
            errlogpf.create(errorpf);
        } else {
            if (locale == 'en') {
                errors.code = code;
                errors.message = [
                    error: errorMessage[code]
                ];
            } else {
                errors.code = code;
                errors.message = [
                    error: errorMessageCN[code]
                ];
            }
            errlogpf.create({
                api: req.originalUrl,
                error_code: errors.code,
                error_desc: errors.message.error,
                incoming: JSON.stringify(req.body),
                caller: req.user ? req.user.psusrnum : ''
            });
        }
        errors.result = 'error';
        errors.extra = extra;
        if (ref && !_.isEmpty(ref)) {
            // Write History Log
            common.writeLog(api, JSON.stringify(errors), "OUT", ref, req.user.psusrnum, "FAIL");
        }
    }
    return res.status(status).json(errors); // send the error response to client
};

```

Figure 5.3: Code Snippet for Error Response

5.1.3: Header Setup

The Header Setup section implements user authentication via JSON Web Tokens (JWT) using Passport.js, specifically the JWT strategy. This setup ensures that all API routes requiring authentication can securely extract and verify user credentials from request headers. When a request with a JWT token is received, the strategy decodes the payload and uses the user ID (`jwt_payload.id`) to search for the corresponding user in the `psusrprf` table. If a match is found, a user object is constructed, containing essential attributes like `username`, `email`, `user type`, `status`, `role`, and `first login flag`. Depending on the user's type (`psusrtyp`), additional data is attached: for members (MBR), the system retrieves the member UID and points from the `psmbrprf` table; for merchants (MCH), it fetches the merchant UID from the `psmrcpar` table. If no related data is found, the respective fields are initialized as empty strings. This dynamic user object is then passed to the `done()` callback, which attaches the authenticated user to the request object for use in protected routes. If no user is found, authentication fails. This centralized header setup ensures secure, role-aware access control across the application and simplifies downstream logic by consistently attaching validated user details to each request.

Example Code Snippet:

```
module.exports = passport => {
  passport.use(
    new JwtStrategy(opts, (jwt_payload, done) => {
      psusrprf.findByPk(jwt_payload.id, { raw: true }).then(async result => {
        if (result) {
          let user = {
            id: result.id,
            psusrnum: result.psusrnum,
            psusrnam: result.psusrnam,
            psusreml: result.psusreml,
            psusrtyp: result.psusrtyp,
            psfstlgn: result.psfstlgn,
            pschgpwd: result.pschgpwd,
            psusrsts: result.psusrsts,
            psusrphn: result.psusrphn,
            psusrpre: result.psusrpre,
            psusrrol: result.psusrrol
          }

          if (user.psusrtyp == "MBR") {
            let mbr = await psmbrprf.findOne({ where: { psmbphn: user.psusrnum }, raw: true, attributes: ['psmbruid'] })

            if (mbr) {
              user.psmbruid = mbr.psmbruid
              user.psmbrpts = mbr.psmbrpts
            } else {
              user.psmbruid = ''
              user.psmbrpts = ''
            }
          } else if (user.psusrtyp == "MCH") {
            let merchant = await psmrcpar.findOne({ where: { psmrcown: user.psusrnum }, raw: true, attributes: ['psmrcuid'] })

            if (merchant) {
              user.psmrcuid = merchant.psmrcuid
            } else {
              user.psmrcuid = ''
            }
          }
        }

        return done(null, user);
      } else return done(null, false);
    }).catch(err => console.log(err));
  });
};
```

Figure 5.4: Code Snippet for Header response setup

5.1.4: Standardized List Function Structure

The Standardized List Function Structure is a reusable pattern in StallSync for retrieving and presenting paginated data from the database. The function begins by determining the pagination settings: it sets a default limit of 10 records per page but allows the client to specify a custom limit and page number through query parameters. Next, it handles filtering and sorting options dynamically based on input query parameters such as type, type2, and search. If a search term is provided, the query will match records using either exact or partial matches on the primaryKey and description fields using Sequelize's Op.eq and Op.like operators.

The function then performs the database query using findAndCountAll() to retrieve both the data (rows) and total count of matching records (count). Specific fields are selected using aliases and ordering is applied. After fetching, the system enriches the results by mapping certain type fields (like type1 and type5) to their human-readable descriptions via a helper function retrieveSpecificGenCodes(). This helps in making the response more understandable for the end-user.

Finally, the function wraps the results in a consistent response format using the returnSuccess() utility. If records are found, it returns the data along with total count and additional metadata; if no records match, it returns an empty dataset with a total of 0. This modular, standardized approach ensures maintainability and consistency across all list-based endpoints in the system.

Example Code Snippet:

```

exports.list = async (req, res) => {
    //1. Record Fetching Setting
    let limit = 10;
    if (req.query.limit) limit = req.query.limit;

    let from = 0;
    if (!req.query.page) from = 0;
    else from = parseInt(req.query.page) * parseInt(limit);

    //2. Sort and Filter (If Any)
    let option = {};

    req.query.primaryKey = req.query.primaryKey ? req.query.primaryKey : "";
    req.query.description = req.query.description ? req.query.description : "";

    if (req.query.type && !_.isEmpty(req.query.type)) {
        option.type1 = req.query.type;
    }

    if (req.query.type2 && !_.isEmpty(req.query.type2)) {
        option.type2 = req.query.type2;
    }

    if (req.query.search && !_.isEmpty(req.query.search)) {
        option = {
            [Op.or]: [
                { primaryKey: { [Op.eq]: req.query.search } },
                { primaryKey: { [Op.like]: "%" + req.query.search + "%" } },
                { description: { [Op.eq]: req.query.search } },
                { description: { [Op.like]: "%" + req.query.search + "%" } },
            ],
        };
    }

    //3. Start Fetching
    const { count, rows } = await filename.findAndCountAll({
        limit: parseInt(limit),
        offset: from,
        where: option,
        raw: true,
        attributes: [
            "primaryKey", "id",
            "primaryKey",
            "description",
            "localDescription",
            "type1",
            "type2",
            "type3",
            "type4",
            "type5",
            "type6",
        ],
        order: [["id", "asc"]],
    });

    //4. Apply the types description (For example A - Active, E - Expired...) && Apply Format
    let newRows = [];
    for (var i = 0; i < rows.length; i++) {
        let obj = rows[i];

        if (!_.isEmpty(obj.type1)) {
            let description = await common.retrieveSpecificGenCodes(
                req,
                "CODE1",
                obj.type1
            );
            obj.type1desc =
                description.prgedesc && !_.isEmpty(description.prgedesc)
                ? description.prgedesc
                : "";
        }

        if (!_.isEmpty(obj.type5)) {
            let description = await common.retrieveSpecificGenCodes(
                req,
                "YESORNO",
                obj.type5
            );
            obj.type5desc =
                description.prgedesc && !_.isEmpty(description.prgedesc)
                ? description.prgedesc
                : "";
        }

        newRows.push(obj);
    }

    // 5. Return Result
    if (count > 0)
        return returnSuccess(
            200,
            {
                total: count,
                data: newRows,
                extra: { file: "filename", key: ["filename"] },
            },
            res
        );
    else return returnSuccess(200, { total: 0, data: [] }, res);
};

```

Figure 5.5: Code Snippet for Standardized List Function

5.1.5: Standardized FindOne/View Function Structure

The Standardized FindOne/View Function Structure in StallSync serves as a standardized method for retrieving and displaying a single record based on a unique identifier. The function begins by validating the presence of the id parameter in the request query. If the ID is missing or empty, the system immediately returns an error using the returnError() utility, indicating that the record ID is required. Once validated, the system uses Sequelize's findOne() method to query the database and retrieve the corresponding record based on the primaryKey value.

If a matching record is found, the function proceeds to enrich the data by formatting specific fields for better readability. For example, if the type1 or type5 fields are not empty, it calls retrieveSpecificGenCodes() to fetch human-readable descriptions from the general codes table, and appends these descriptions as type1dsc and type5dsc respectively. This improves the clarity of the response for frontend use. After formatting, the fully prepared object is returned using the returnSuccess() utility with a 200 status code. If no matching record is found, an appropriate error response is returned. Additionally, any unexpected database errors are caught and logged, and a generic error message is returned to the client. This function ensures a clean and consistent way to view individual records across different modules in the system.

Example Code Snippet:

```

exports.findOne = async (req, res) => {
    // 1. Validate Id
    const id = req.query.id ? req.query.id : "";
    if (id == "") return returnError(req, 500, "RECORDIDISREQUIRED", res);
    // 2. Fetch Data
    filename
        .findOne({ where: { primaryKey: id }, raw: true })
        .then(async (obj) => {
            if (obj) {

                // 3. Formatting
                if (!_.isEmpty(obj.type1)) {
                    let description = await common.retrieveSpecificGenCodes(
                        req,
                        "CODE1",
                        obj.type1
                    );
                    obj.type1desc =
                        description.prgedesc && !_.isEmpty(description.prgedesc)
                            ? description.prgedesc
                            : "";
                }
                if (!_.isEmpty(obj.type5)) {
                    let description = await common.retrieveSpecificGenCodes(
                        req,
                        "YESORNO",
                        obj.type5
                    );
                    obj.type5desc =
                        description.prgedesc && !_.isEmpty(description.prgedesc)
                            ? description.prgedesc
                            : "";
                }
            }

            // 4. Return Result
            return returnSuccess(200, obj, res);
        } else return returnError(req, 500, "NORECORDFOUND", res);
    })
    .catch((err) => {
        console.log(err);
        return returnError(req, 500, "UNEXPECTEDERROR", res);
    });
}

```

Figure 5.6: Code Snippet for Standardized Detail/FindOne/View Function

5.1.6: Standardized Create Function Structure

The Standardized Create Function Structure in StallSync defines the standard process for inserting new records into the database while ensuring data integrity and consistency. The function begins with input validation using a dedicated validatefilenameInput() function. If any validation fails, it immediately responds with detailed error messages through returnError(). After validation, the system generates a unique reference code using a combination of a prefix ("TRN-") and an auto-incremented running number obtained from getNextRunning(). This ensures that every record has a traceable and standardized identifier. Before creating a new entry, the function checks for duplication by querying the database for an existing record with the same filename. If a duplicate exists, it halts the operation and returns an error. If no duplicate is found, the function performs dropdown code validation to verify that user-selected values for fields like type1 and type5 are valid entries in the general code list (e.g., CODE1, YESORNO). If any of these validations fail, an error is returned indicating invalid data values.

If all checks pass, the record is created using Sequelize's create() method, storing all relevant fields along with metadata such as the creator and maintainer usernames (crtuser, mntuser). After successful creation, the function logs the action using writeMntLog() for audit purposes, and finally sends a standardized success message (RECORDCREATED) using returnSuccessMessage(). If any error occurs during creation or database access, it is caught and reported using a generic UNEXPECTEDERROR handler. This structured approach promotes robust error handling, validation, and audit logging in the system's data creation process.

Example Code Snippet:

```

exports.create = async (req, res) => {
    //1. Validation
    const { errors, isValid } = validatefilenameInput(req.body, "A");
    if (!isValid) return returnError(req, 400, errors, res);

    // 2. Generate Code(uiv, running code), if any:
    let code = await common.getNextRunning("TRN");
    let initial = "TRN-";
    let reference = initial;
    reference += _.padStart(code, 6, '0');

    //3. Duplicate Check
    filename
        .findOne({
            where: {
                filename: req.body.filename,
            },
            raw: true,
        })
        .then(async (trnsd) => {
            if (trnsd)
                return returnError(req, 400, { filename: "RECORDEXISTS" }, res);
            else {
                //4. Code Checking
                let ddlErrors = {};
                let err_ind = false;
                let CODE1 = await common.retrieveSpecificGenCodes(
                    req,
                    "CODE1",
                    req.body.type1
                );
                if (!CODE1 || !CODE1.prgedesc) {
                    ddlErrors.type1 = "INVALIDDATAVALUE";
                    err_ind = true;
                }
                if (!_.isEmpty(req.body.type5)) {
                    let yesorno = await common.retrieveSpecificGenCodes(
                        req,
                        "YESORNO",
                        req.body.type5
                    );
                    if (!yesorno || !yesorno.prgedesc) {
                        ddlErrors.type5 = "INVALIDDATAVALUE";
                        err_ind = true;
                    }
                }
                if (err_ind) return returnError(req, 400, ddlErrors, res);
                else {

                    //5. Creation of record
                    filename
                        .create({
                            filename: req.body.filename,
                            description: req.body.description,
                            localDescription: req.body.localDescription,
                            type1: req.body.type1,
                            type2: req.body.type2,
                            type3: req.body.type3,
                            type4: req.body.type4,
                            type5: req.body.type5,
                            type6: req.body.type6,
                            crtuser: req.user.psusrnum,
                            mntuser: req.user.psusrnum,
                        })
                        .then(async (data) => {
                            let created = data.get({ plain: true });

                            // 6. Logging
                            common.writeMntLog(
                                "filename",
                                null,
                                null,
                                created.filename,
                                "A",
                                req.user.psusrnum,
                                "", created.filename);

                            //7. Return Success
                            return returnSuccessMessage(req, 200, "RECORDCREATED", res);
                        })
                        .catch((err) => {
                            console.log(err);
                            return returnError(req, 500, "UNEXPECTEDERROR", res);
                        });
                }
            }
        })
        .catch((err) => {
            console.log(err);
            return returnError(req, 500, "UNEXPECTEDERROR", res);
        });
}

```

Figure 5.7: Code Snippet for Standardized Create Function

5.1.7: Standardized Update Function Structure

The Standardized Update Function Structure in StallSync defines a secure and consistent process for modifying existing records in the database. It begins by retrieving the record's primary identifier (id) from the request body. If the ID is missing, the system immediately returns an error. The function then validates the incoming data using a context-aware validation method (validatefilenameInput) where "C" indicates update mode. If any validation errors are detected, the update is halted and the client is notified accordingly. Next, the function attempts to fetch the existing record from the database using Sequelize's findOne() method. It excludes certain audit fields (createdAt, crtuser, mntuser) from the result. Upon finding the record, the system performs a timestamp consistency check using the updatedAt field to ensure the data hasn't changed since the client retrieved it, which is a technique known as optimistic concurrency control. If the timestamps don't match, it returns a "RECORDOUTOFSYNC" error, prompting the user to reload the latest data.

Before proceeding with the update, the function performs dropdown code validation for fields such as type1 and type5, using helper functions to ensure the values provided are valid entries from general code tables like CODE1 and YESORNO. If any invalid codes are detected, the system stops the update and returns a descriptive error.

Once all checks pass, the update is executed using Sequelize's update() method, modifying the necessary fields and updating the mntuser (modified by) metadata. After the update, the system logs the before-and-after states using writeMntLog() to maintain audit trails for accountability. Finally, the function returns a standardized success message

(RECORDUPDATED) through returnSuccessMessage(). Any unexpected errors during the process are caught and returned as internal errors. This comprehensive structure ensures data accuracy, prevents conflicts, and maintains consistent audit logging.

Example Code Snippet:

```

exports.update = async (req, res) => {
    // 1. Get To-be-updated Primary Key (Known as ID)
    const id = req.body.id ? req.body.id : "";
    if (!id == "") return returnError(req, 500, "RECORDIDISREQUIRED", res);

    //2. Validation Checking
    const { errors, isValid } = validatefilenameInput(req.body, "C");
    if (!isValid) return returnError(req, 400, errors, res);

    //3. Fetch Record
    await filename
        .findOne({
            where: {
                filename: id,
            },
            raw: true,
            attributes: {
                exclude: ["createdAt", "crtuser", "mntuser"],
            },
        })
        .then(async (data) => {
            if (data) {
                // 4. Check If date is updated
                if (isNaN(new Date(req.body.updatedAt)) || (new Date(data.updatedAt).getTime() != new Date(req.body.updatedAt).getTime()))
                    | return returnError(req, 500, "RECORDOUTOFSYNC", res)

                // 5. Check Formatting / Code
                let ddlErrors = {};
                let err_ind = false;
                let CODE1 = await common.retrieveSpecificGenCodes(
                    req,
                    "CODE1",
                    req.body.type1
                );
                if (!CODE1 || !CODE1.prgedesc) {
                    ddlErrors.type1 = "INVALIDDATAVALEUE";
                    err_ind = true;
                }
                if (!_.isEmpty(req.body.type5)) {
                    let yesorno = await common.retrieveSpecificGenCodes(
                        req,
                        "YESORNO",
                        req.body.type5
                    );
                    if (!yesorno || !yesorno.prgedesc) {
                        ddlErrors.type5 = "INVALIDDATAVALEUE";
                        err_ind = true;
                    }
                }
                if (err_ind) return returnError(req, 400, ddlErrors, res);

                //6. Start Update
                filename
                    .update(
                        {
                            description: req.body.description,
                            localDescription: req.body.localDescription,
                            type1: req.body.type1,
                            type2: req.body.type2,
                            type3: req.body.type3,
                            type4: req.body.type4,
                            type5: req.body.type5,
                            type6: req.body.type6,
                            mntuser: req.user.psusrnum,
                        },
                        {
                            where: {
                                id: data.id,
                            },
                        }
                    )
                    .then(async () => {
                        //7. Logging
                        common.writeMntLog(
                            "filename",
                            data,
                            await filename.findById(data.id, { raw: true }),
                            data.filename,
                            "C",
                            req.user.psusrnum
                        );
                        //8. Return Result
                        return returnSuccessMessage(req, 200, "RECORDUPDATED", res);
                    });
                [] else return returnError(req, 500, "NORECORDFOUND", res);
            }
        })
        .catch((err) => {
            return returnError(req, 500, "UNEXPECTEDERROR", res);
        });
};
}

```

Figure 5.8: Code Snippet for Standardized Update Function

5.1.8: Standardized Delete Function Structure

The Standardized Delete Function Structure in StallSync provides a systematic approach for safely removing records from the database while maintaining traceability. The function starts by retrieving the primary identifier (id) of the record to be deleted from the request body. If the id is not provided, it immediately responds with an error indicating that the record ID is required. It then attempts to locate the target record using Sequelize's `findOne()` method. If the record is found, the function proceeds to delete it using the `destroy()` method, which removes the record based on its internal id.

To ensure accountability, a deletion log is created using `writeMntLog()`, recording the action type ("D" for delete), the username of the person who performed the deletion, and the identifier of the deleted record. This logging mechanism supports audit trail requirements and makes the system more transparent. After successful deletion and logging, the function returns a standardized success response (RECORDDELETED) to inform the client. If the record is not found or if any unexpected error occurs during the process, the function captures the issue and returns a relevant error message. This structured pattern guarantees secure and traceable deletions within the application.

Example Code Snippet:

```
exports.delete = async (req, res) => {
    //1. Get to-be-deleted Primary Key(Known as id)
    const id = req.body.id ? req.body.id : "";
    if (id == "") return returnError(req, 500, "RECORDIDISREQUIRED", res);

    //2. fetch record
    await filename
        .findOne({
            where: {
                filename: id,
            },
            raw: true,
        })
        .then((trnscd) => {
            if (trnscd) {
                //3. Start to delete
                filename
                    .destroy({
                        where: { id: trnscd.id },
                    })
                    .then(() => {
                        //4. Logging
                        common.writeMntLog(
                            "filename",
                            null,
                            null,
                            trnscd.filename,
                            "D",
                            req.user.psusrnum,
                            "",
                            trnscd.filename
                        );
                    });

                //5. Return result
                return returnSuccessMessage(req, 200, "RECORDDELETED", res);
            }
        })
        .catch((err) => {
            console.log(err);
            return returnError(req, 500, "UNEXPECTEDERROR", res);
        });
    } else return returnError(req, 500, "NORECORDFOUND", res);
}
.catch((err) => {
    console.log(err);
    return returnError(req, 500, "UNEXPECTEDERROR", res);
});
};

};
```

Figure 5.9: Code Snippet for Standardized Delete Function

5.1.9: Setting for Server

The server setup for the StallSync system is configured using Node.js with the Express.js framework to handle backend API requests. The server environment is initialized in a dedicated entry point file (e.g., server.js or index.js), where middleware such as body-parser, cors, and dotenv are applied to support JSON parsing, cross-origin requests, and environment variable management respectively. The backend server listens on a defined port (commonly 3001 or any custom value defined in the .env file). Additionally, modular route files are imported to keep the API structure organized. Error handling middleware is also included to manage unexpected behaviors consistently across all endpoints. This setup ensures a clean, maintainable, and scalable backend environment to serve API operations securely and efficiently.

The backend server for the system is built using Express.js on top of Node.js. It integrates important middleware for JSON parsing, security, logging, authentication, and routing. The server also uses Sequelize to connect to a MySQL database and supports automatic synchronization of models. passport is used for authentication, and rate-limit is added to prevent abuse by limiting the number of requests per IP. Logs for API access and system errors are handled using morgan and written to log files with timestamps via the moment library. Multiple route modules are organized under api/ endpoints. The server listens on a specified port (default 3000) and includes retry logic in case the database connection fails.

Example Code Snippet:

```

const express = require('express');
const bodyParser = require('body-parser');
const passport = require('passport');
const cors = require('cors');
const morgan = require('morgan');
const fs = require('fs');
const path = require('path');
const moment = require('moment');
const rateLimit = require("express-rate-limit");
const db = require('./models');
const common = require('../common/common');

const app = express();

// Middleware
app.use(cors({ origin: ['http://localhost:3000'], credentials: true }));
app.use(bodyParser.json());
app.use(passport.initialize());
require('../config/passport')(passport);

// Logging
const accessLog = fs.createWriteStream(path.join(__dirname, 'logs/access.log'), { flags: 'a' });
morgan.token('date', () => moment().format('DD-MM-YYYY, h:mm:ss a'));
app.use(morgan('[:date] :method :url :status', { stream: accessLog }));

// Rate limiting
app.use(rateLimit({ windowMs: 15 * 60 * 1000, max: 100 }));

// Routes
app.use('/api/psusrprf', require('../routes/psusrprf'));
// (Add other routes similarly)

// Fallback for unknown APIs
app.use((req, res) => res.status(404).send('APINOTFOUND'));

// Error logging
process.on('uncaughtException', err => {
  common.logging("ERROR", moment().format() + " " + JSON.stringify(err));
  process.exit(1);
});

process.on('unhandledRejection', err => {
  common.logging("ERROR", moment().format() + " " + JSON.stringify(err));
  process.exit(1);
});

// Start server
async function start() {
  try {
    await db.sequelize.authenticate();
    await db.sequelize.sync();
    const PORT = process.env.PORT || 3000;
    app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
  } catch (err) {
    console.error('DB connection failed. Retrying...', err.message);
    setTimeout(start, 5000);
  }
}

start();

```

Figure 5.10: Code Snippet for Server.js

5.1.10: Network Protocol

The system communicates primarily over the HTTP/HTTPS protocol, which is the foundation of web communication between the client and server. In development, HTTP is typically used for speed and simplicity, while in production, HTTPS is enforced to ensure encrypted and secure data transmission between the frontend (Next.js application) and the backend (Node.js API). Each request made from the frontend, such as placing an order or fetching a user profile, is routed through HTTP methods like GET, POST, PUT, or DELETE. Proper header configuration is applied to include tokens (e.g., JWT) for authenticated endpoints. This secure protocol design helps prevent man-in-the-middle attacks, ensures data confidentiality, and supports modern browser standards.

Example Code Snippet:

```
const express = require('express');
const router = express.Router();

// -- Load Common Authentication --
const authenticateRoute = require('../common/authenticate');
// -- Load Controller --
const psprdpar = require("../controllers/psprdpar-controller");

// @route   GET api/psprdpar/find-one
// @desc    Find OTP Parameter
// @access  Private
router.get("/detail", authenticateRoute, psprdpar.findOne);

// @route   GET api/psprdpar/list
// @desc    List OTP Parameter
// @access  Private
router.get("/list", authenticateRoute, psprdpar.list);

// @route   POST api/psprdpar/create
// @desc    Create OTP Parameter
// @access  Private
router.post("/create", authenticateRoute, psprdpar.create);

// @route   POST api/psprdpar/delete
// @desc    Delete OTP Parameter
// @access  Private
router.post("/delete", authenticateRoute, psprdpar.delete);

// @route   POST api/psprdpar/update
// @desc    Update OTP Parameter
// @access  Private
router.post("/update", authenticateRoute, psprdpar.update);

module.exports = router;
```

Figure 5.11: Code Snippet for Modules Routing

5.1.11: Database



Figure 5.12: XAMPP Logo

source:https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSQLWaVXSNCqNB3i97_nQAnK_gES0WBCfnvVQ&s

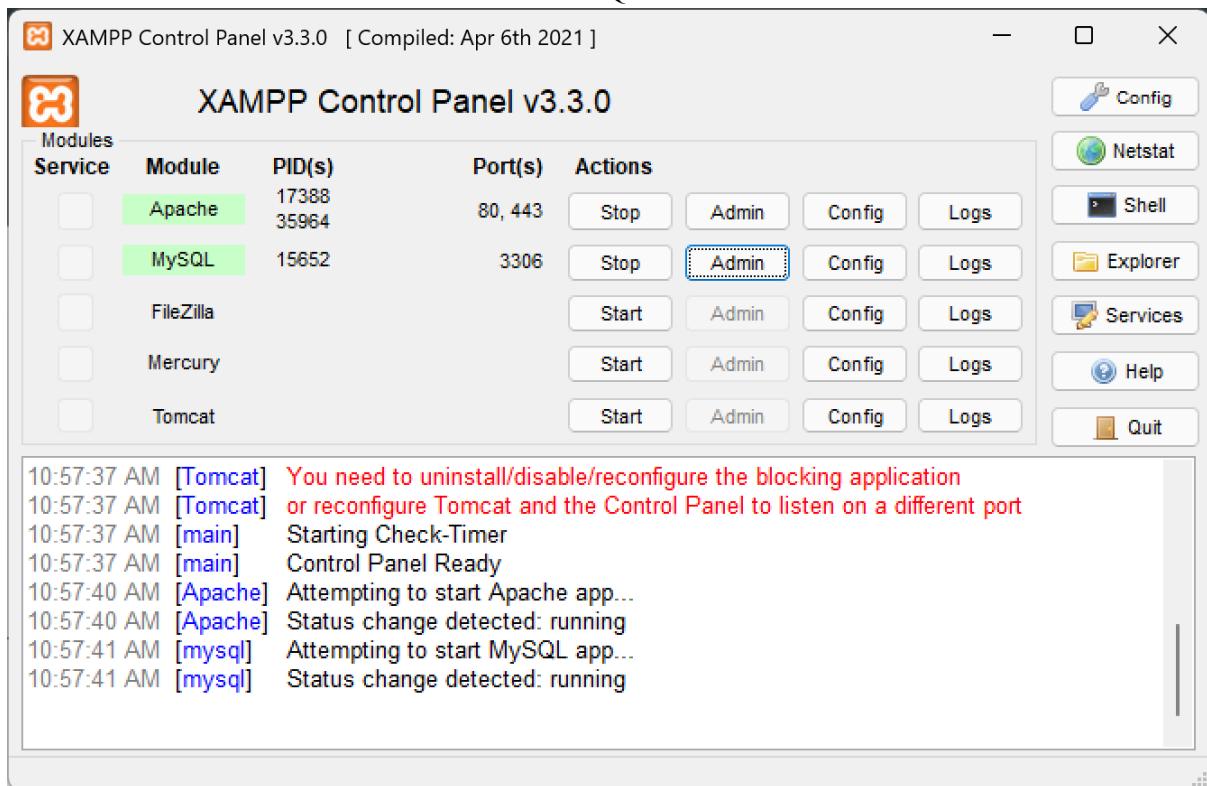


Figure 5.13: XAMPP UI

For database management, MySQL is used as the relational database system, managed through phpMyAdmin which is a user-friendly web-based interface provided by XAMPP. XAMPP is a free, local development environment that bundles Apache, MySQL, and PHP, enabling developers to run the backend server and database locally. The database schema is designed using MySQL's table structure, which stores data related to users, merchants, orders, inventory, and reports. PhpMyAdmin allows for easy database administration tasks such as importing/exporting tables, managing indexes, setting up foreign key relationships, and executing raw SQL queries. It is also useful for real-time data inspection during development and testing. MySQL's reliability and compatibility with Sequelize ORM ensure smooth integration with the Node.js backend.

5.1.12: Forecasting (Machine Learning)

The forecasting system integrates a Node.js backend with a Python-based machine learning module (`app.py`) to provide automated sales and order count predictions. The core functionality begins with the `forecast` function in `Node.js`, which is triggered through an API call. This function first performs input validation to ensure the required parameters — `prrptnme` (report name) and `forecast_type` (either "S" for sales or "O" for orders) — are provided. If either is missing, an error response is returned immediately.

Once validated, the backend spawns a Python process using `child_process.spawn` to run `ML/app.py`. The report name and forecast type are passed to the script via standard input in JSON format. The backend then listens for output from the Python script on `stdout`, which is expected to return a JSON string containing the filename of the generated forecast image. If there are any errors during execution, such as issues with the Python logic or data parsing, they are captured and logged through `stderr`.

When the Python process completes, the backend parses the returned JSON to extract the forecast image filename. It then updates the appropriate forecast status field (`prrpfc`s or `prrpfc`o) in the `prrpthis` table, records the image metadata in the `psdocmas` table, and moves the image from a temporary storage location to the final storage path using the `common.writeImage` utility. If any step fails — for example, if the image writing fails or JSON parsing fails — the backend ensures error handling is in place by rolling back the database transaction to maintain data integrity.

To retrieve forecast results, the `findForecast` function is used. This function first validates the query parameters (`prrptnme`, `prrpfc`o, and `prrpfc`s) and then checks for the existence of the report in the database. Based on the forecast status, it proceeds to read the relevant forecast CSV files — either `forecastOrder-<basename>.csv` or `forecastSales-<basename>.csv` — and formats the forecasted values into a readable structure. It also constructs the corresponding forecast image filenames (`order-<basename>.png` or `sales-<basename>.png`). The processed data and image references are then returned to the frontend for visualization.

On the Python side, the `app.py` script performs the machine learning logic. It reads JSON input from `stdin`, which includes the report name and forecast type, and then decides whether to perform sales forecasting (`sales_predict`) or order forecasting (`order_predict`). Both functions follow a similar process: they load the CSV report from `./documents/reports/`, standardize the `Transaction_Date` column, fill in any missing dates, and aggregate the data (either transaction amounts or order counts per day). The ARIMA model with fixed parameters ($p=5$, $d=0$, $q=5$) is trained on this time-series data to forecast the next 30 days. The script then generates a line plot that overlays actual data (solid blue) and forecasted data (dashed red) using Matplotlib and Seaborn. The forecast graph is saved as a PNG image in `./documents/document_temp/`, and the forecast data is saved as a CSV file in `./documents/forecast_file/`. Finally, the Python script outputs a simple JSON containing the image filename, which is returned to the `Node.js` backend.

The project's directory structure supports this workflow: the `/ML` folder contains the Python script and cache files, while `/documents/reports/` stores input CSVs, `/documents/forecast_file/` contains the output forecast data, and `/documents/document_temp/` holds the temporary image files. This setup ensures a clean separation of logic, data, and outputs, making the forecasting module both maintainable and scalable.

Example Code Snippet (Function to invoke python code):

```

exports.forecast = async (req, res) => {
  const forecast_type = req.body.forecast_type ? req.body.forecast_type : "";
  const id = req.body.prrptname ? req.body.prrptname : "";
  if (id == "") return returnError(req, 500, { id: "RECORDIDISREQUIRED" }, res);
  if (forecast_type == "") return returnError(req, 500, { forecast_type: "RECORDIDISREQUIRED" }, res);
  try {
    const pythonProcess = spawn("python", ["./ML/app.py"]);
    pythonProcess.stdin.write(JSON.stringify({
      forecast_type: forecast_type,
      prrpname: id
    }));
    pythonProcess.stdin.end();
    let responseData = "";
    pythonProcess.stdout.on("data", (chunk) => {
      console.log("Raw Python Response:", chunk.toString());
      responseData += chunk.toString();
    });
    pythonProcess.stderr.on("data", (data) => {
      console.error("Python Error:", data.toString());
    });
  }
  pythonProcess.on("close", async (code) => {
    if (code != 0) {
      console.error(`[PYTHON ERROR] Exit code: ${code}`);
      console.error(`[PYTHON STDERR]`, responseData);
      return returnError(req, 500, "Forecast Failed, Unexpected Error", res);
    }
    if (!responseData.trim()) {
      console.error("No response received from Python script.");
      return returnError(req, 500, "Empty Returning from Forecasting", res);
    }
    try {
      const jsonResponse = JSON.parse(responseData.trim());
      const t = await connection.sequelize.transaction();
      // Update forecast status in DB
      await prrptthis.update(
        forecast_type === "S"
          ? { prrpfcst: "Y" }
          : { prrpfcst: "N" },
        {
          where: { prrpname: id },
          transaction: t
        }
      );
      let ext = ".png";
      await psdocmas.create({ psocfnm: jsonResponse.filename, psocomm: jsonResponse.filename, psocudt: new Date(), psocext: ext.toString().toLowerCase() })
      // Save forecast image
      await common.writeFileImage(
        config.documentTempPath,
        config.forecastImagePath,
        jsonResponse.filename,
        req.user.psusrnum,
        7
      ).catch(async (err) => {
        await t.rollback();
        console.log("ERR Write Img: ", err);
        return returnError(req, 500, "Image Write Failed", res);
      });
      await t.commit();
      if (!res.headersSent) return returnSuccessMessage(req, 200, "Request Processed", res);
    }
    catch (error) {
      console.error("JSON Parsing Error:", error);
      await t.rollback();
      console.error("Raw Response:", responseData);
      return returnError(req, 500, "INVALIDJSONRESPONSE", res);
    }
  });
  catch (error) {
    console.error("Error in processing:", error);
    return returnError(req, 500, "UNEXPECTEDERROR", res);
  }
}

```

Figure 5.14: Code Snippet for Forecast getting handle the python code

Example code snippet for modified forecasting algorithm (ARIMA):

```

def sales_predict(report_name):
    # Load data
    df = pd.read_csv("./documents/reports/*report_name")

    # Convert date column
    df['Transaction_Date'] = df['Transaction_Date'].str.replace('/', '-')
    df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'], format="mixed")

    # Fill missing dates
    date_range = pd.date_range(start=df["Transaction_Date"].min(), end=df["Transaction_Date"].max())
    complete_dates = pd.DataFrame(date_range, columns=["Transaction_Date"])
    df_by_date = df.groupby("Transaction_Date").agg({"Transaction_Amount": ["sum"]}).reset_index()
    df_by_date.columns = ["Transaction_Date", "Transaction_Amount"]
    df_complete = pd.merge(complete_dates, df_by_date, on="Transaction_Date", how="left").fillna(0)

    # Train ARIMA on all available data
    p, d, q = 5, 0, 5
    model = ARIMA(df_complete['Transaction_Amount'], order=(p, d, q))
    model_fit = model.fit()

    # Forecast next 30 days
    forecast_steps = 30
    future_dates = pd.date_range(start=df_complete["Transaction_Date"].iloc[-1] + pd.Timedelta(days=1), periods=forecast_steps)
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create forecast DataFrame
    df_forecast = pd.DataFrame({
        "Transaction_date": future_dates,
        "arima_pred": forecast_values
    })

    base_name = os.path.splitext(report_name)[0]

    # Plot actual + forecast
    plt.figure(figsize=(14, 6))
    sns.lineplot(data=df_complete, x="Transaction_Date", y="Transaction_Amount", label="Actual", color="blue")
    sns.lineplot(data=df_forecast, x="Transaction_Date", y="arima_pred", label="Forecast (Next 30 Days)", color="red", linestyle="dashed")
    plt.title("Sales Forecast for Next 30 Days")
    plt.grid()
    plt.ylim(0)
    plt.xticks(rotation=45)
    plt.savefig("./documents/document_temp/sales-*base_name*.png", dpi=300)

    # Save forecast to CSV
    df_forecast.to_csv("./documents/forecast_file/forecastSales-*base_name*.csv", index=False)
    return return_json("sales-*base_name*.png")

def order_predict(report_name):
    # Load data
    df = pd.read_csv("./documents/reports/*report_name")

    # Convert date column
    df['Transaction_Date'] = df['Transaction_Date'].str.replace('/', '-')
    df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'], format="mixed")

    # Fill missing dates
    date_range = pd.date_range(start=df["Transaction_Date"].min(), end=df["Transaction_Date"].max())
    complete_dates = pd.DataFrame(date_range, columns=["Transaction_Date"])

    # Count number of orders per day
    df_by_date = df.groupby("Transaction_Date").size().reset_index(name="Order_Count")

    # Merge with full date range
    df_complete = pd.merge(complete_dates, df_by_date, on="Transaction_Date", how="left").fillna(0)

    # Train ARIMA on order count
    p, d, q = 5, 0, 5
    model = ARIMA(df_complete['Order_Count'], order=(p, d, q))
    model_fit = model.fit()

    # Forecast next 30 days
    forecast_steps = 30
    future_dates = pd.date_range(start=df_complete["Transaction_Date"].iloc[-1] + pd.Timedelta(days=1), periods=forecast_steps)
    forecast_values = model_fit.forecast(steps=forecast_steps)

    # Create forecast DataFrame
    df_forecast = pd.DataFrame({
        "Transaction_date": future_dates,
        "arima_pred": forecast_values
    })

    base_name = os.path.splitext(os.path.basename(report_name))[0]

    # Plot actual + forecast
    plt.figure(figsize=(14, 6))
    sns.lineplot(data=df_complete, x="Transaction_Date", y="Order_Count", label="Actual", color="blue")
    sns.lineplot(data=df_forecast, x="Transaction_Date", y="arima_pred", label="Forecast (Next 30 Days)", color="red", linestyle="dashed")
    plt.title("Order Forecast for Next 30 Days")
    plt.grid()
    plt.ylim(0)
    plt.xticks(rotation=45)
    plt.savefig("./documents/document_temp/order-*base_name*.png", dpi=300)

    # Save forecast to CSV
    df_forecast.to_csv("./documents/forecast_file/forecastOrder-*base_name*.csv", index=False)
    return return_json("order-*base_name*.png")

if __name__ == "__main__":
    request = read_input()
    print(f"[{DEBUG}] Incoming request: {request}", file=sys.stderr)
    report_name = request.get("reportname")
    forecast_type = request.get("forecast_type")

    if not report_name:
        raise ValueError("Missing 'reportname' in input JSON")

    if forecast_type == "S":
        sales_predict(report_name)
    elif forecast_type == "O":
        order_predict(report_name)
    else:
        raise ValueError("Invalid or missing 'forecast_type'. Expected 'S' or 'O'")


```

Figure 5.15: Code Snippet for Modified ARIMA Algorithm to fit in the system

5.2 Testing

5.2.1 Test Plan

Overview

This test plan is designed to evaluate the **StallSync intelligent management system**. It focuses on how the system works for both merchant and admin users. The system has two main parts. One is the management side. The other is the client ordering side.

The goal of this testing process is to find any errors or problems during development. This helps make sure the system meets all the required features and works as expected.

This plan uses black box testing. This method checks the system's output based on input. It does not look at how the code is written. The focus is on whether the system behaves correctly. The main focus is on admin features. These include staff and member management, announcements, order and product handling, merchant and reward features, transaction tracking, reports, and system settings.

The testing uses an **incremental approach**. This method tests each part of the system step by step. It helps find errors early. It also helps fix wrong links or assumptions between modules. This method is helpful for admin tasks. It makes sure each module is tested well before the next one is added. By the time the full system is tested, all earlier parts have already been checked. This gives a full and strong review of how the system handles stalls, orders, and staff.

Test Scope and Test Item

The test plan's project scope encompasses conducting black box testing on the website application. This type of testing focuses on evaluating the external functionalities of the application. Specifically, functional testing will be employed to verify the incorporation of all proposed features and to ensure their smooth operation. The below listed modules are the high priority to be tested module, while the other modules will not be listed but will be tested too.

In User Account modules, I will focus on the user authentication and the profile management. This includes user login and logout, user account creation, forgot password, change password, user profile update, and user account deletion. These functionalities to be tested are focusing mainly on admins, staff, owners but not the member/customer.

In the Staff module, I will focus on the basic CRUD (Create, Read, Update, and Delete) function that is used to manage the staff. All the staff can be viewed and managed by the admin while the merchant owner or the merchant staff can only view and manage the staff that are from the same merchant.

In the Merchant module, I will focus on the basic CRUD (Create, Read, Update, and Delete) function that is used to manage the merchant. All the merchants can be viewed and managed by the admin while the merchant owner or the merchant staff can only view and manage the merchant that they are from.

In the Member module, I will focus on the basic Read and Update function that is used to manage the member. Only the admin can view and manage all the members (customers).

In the Product module, I will focus on the basic CRUD (Create, Read, Update, and Delete) function that is used to manage the products. All the products can be viewed and managed by the admin while the merchant owner or the merchant staff can only view and manage the product from their merchant.

In the Reward module, I will focus on the basic CRUD (Create, Read, Update, and Delete) function that is used to manage the rewards. All the rewards can be viewed by the admin, merchant staff and merchant owner. Only admin can manage the rewards

In the Announcement module, I will focus on the basic CRUD (Create, Read, Update, and Delete) function that is used to manage the announcements. All the announcements can be viewed and managed by the admin.

In the Transaction module, I will focus on the basic Read function that is used to view the transactions. All the transactions can be viewed by the admin, merchant staff and merchant owner.

In the Reporting module, I will focus on the basic read function, generate report and forecast that is used to manage the reports. All the reports can be viewed by the admin, merchant staff and merchant owner.

Table 5.1: Test Item

User Account Module	<ul style="list-style-type: none"> - Login - Forgot Password - Change password - Create Account - Update Account - Delete Account - List Account
Staff Module	<ul style="list-style-type: none"> - Create Staff - Update Staff - Read Staff - Delete Staff
Merchant Module	<ul style="list-style-type: none"> - Create Merchant - Update Merchant - Read Merchant - Delete Merchant
Member Module	<ul style="list-style-type: none"> - Read Member

	<ul style="list-style-type: none">- Update Member
Product Module	<ul style="list-style-type: none">- Create Product- Update Product- Read Product- Delete Product
Reward Module	<ul style="list-style-type: none">- Create Reward- Update Reward- Read Reward- Delete Reward
Announcement Module	<ul style="list-style-type: none">- Create Announcement- Update Announcement- Read Announcement- Delete Announcement
Transaction Module	<ul style="list-style-type: none">- Read Transaction
Reporting Module	<ul style="list-style-type: none">- List report- Generate Report- Download Report- Forecast Report

Test Strategy

The main goal of a test strategy is to ensure user satisfaction by confirming that the software works as expected and meets user needs. It focuses on both usability and reliability. Usability means the software should be easy to use and simple to navigate. Reliability means the system should run smoothly and remain stable during normal use. A key part of the strategy is defect detection, which involves finding and fixing bugs that could cause errors or system failures. This process is part of Quality Assurance (QA), which is a planned set of steps used to check and maintain the overall quality of the software. QA looks at usability, performance, and reliability to make sure the system matches what users expect.

This test plan uses an **incremental testing method**. Each module will be tested together with the ones that came before it. This helps to check how modules work as a group. For example, the second module will be tested with the first. Then the third module will be tested with both the first and second. This will continue until all modules are fully tested together.

The plan also follows a **bottom-up testing approach**. Testing starts with the smaller, lower-level modules. These are then connected to higher-level ones. This method is known for being easy to use. It also gives strong and reliable test results.

There are two important documents in this process. These are the test plan and the test cases. The test plan gives a full outline of the testing steps. It includes important details about how testing will be done. It also helps teams plan time and cost. This plan keeps everyone on the same page, especially the testing team and the stakeholders.

The second document is the test cases. This shows the exact steps used to check the system. It includes things like what needs to be ready before the test, what to do during the test, what input to use, and what results to expect. It also shows what should happen after the test. These test cases help make sure the system does what it is supposed to do. They also keep the testing clear and well organised.

Test Data Requirements (Test Strategy)

In black box testing, preparing test data is very important. It helps check if the software works as it should. The data must cover many different input types and situations.

Testers should include both valid and invalid inputs. They should also test edge cases and special situations. These cases help show how the system reacts to different inputs.

One useful method is equivalence partitioning. This method splits inputs into groups that should act the same. For example, if a field accepts numbers, tests should use regular numbers, negative values, decimals, and very large or small numbers. This shows how well the system handles each type.

Test data should include both normal and extreme values. Normal values check everyday use. Extreme values test the system's limits. This helps make sure the software can handle stress and unusual input.

Negative test cases are also important. These tests use wrong or badly formatted inputs. For example, an email field can be tested with missing symbols or wrong layouts. This checks if the system can catch and reject incorrect input.

Real-life data is helpful too. This includes real names, addresses, and other inputs that users might enter. It checks if the system can work well with real-world content. Using a mix of test data types gives better results. It helps confirm that the software is strong, reliable, and ready to use.

Test Environment (Test Strategy)

To establish a conducive test environment for the StallSync Intelligent Management System from the administration perspective, several hardware and software components are essential to ensure a comprehensive testing process. The following elements need to be set up:

1. Minimum Requirements hardware:

Table 5.2: Minimum Requirement Hardware

PC Web browser	
Processor	1GHz or faster
RAM	At least 2GB
Storage	Sufficient space to accommodate OS and browser
Mouse	Required
Keyboard	Required
Internet Connection	Required

2. Minimum Requirement Software:

Table 5.3: Minimum Requirement Software

PC Web browser	
Operating System	Windows 7 or later, macOS X 10.10 or later
Browser	Latest versions of Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.

3. Valid Email Address: A valid email address is necessary to test the email-sending function in different scenarios. It ensures that features like email verification can be checked properly and work as expected.
4. Network Connectivity: A stable network connection is essential for launching and testing the website. It allows the application to connect with external services and resources, ensuring smooth data flow and proper system functionality.
5. Database Management: A MySQL (XAMPP) database system is necessary for smooth communication with the application's database. It helps store and retrieve data efficiently, which supports stable and reliable performance during testing.

Test Completion Criteria (Test Strategy)

- All planned test cases are executed, with priority on critical paths
- Achieved at least 95% test case pass rate, with no critical defects
- All high and medium-severity defects are resolved and retested
- No critical defects remain open
- Test results, defect logs, and incident reports are documented and shared
- The selected module passes the test and having the expected result

Testing Activities and Estimates - STLC (Test Strategy)

The Software Testing Life Cycle (STLC) includes six key phases. These are requirement analysis, test planning, test case development, test environment setup, test execution, and test cycle closure. Each phase helps to make the testing process complete and reliable.

In the requirement analysis phase, the testing team studies the software requirements. The goal is to understand what the system should do. This step helps both developers and testers stay on the same page. It also helps find unclear or missing requirements early. During this phase, the team estimates the time, cost, and effort needed for the testing process. It also builds a shared understanding of goals. This helps improve planning, reduce risk, and support better communication.

The test planning phase sets the direction for the testing work. It defines what will be tested and how it will be done. The team lists the tools, resources, and testing methods to use. It also prepares the schedule and identifies any risks. Roles and responsibilities are shared among team members. A clear plan helps everyone work together toward the same goal.

In the test case development phase, the team writes detailed test cases. Each test case has input data, steps to follow, and expected results. Test cases cover different types of input, both valid and invalid. The team also prepares performance, security, and regression test cases. These help check system behavior under different conditions. Writing test cases early helps create a strong and organised testing process.

Next is the test environment setup. This phase prepares everything needed to run the tests. The team sets up hardware, software, databases, and network settings. The test environment should be similar to the real system. After setup, the team runs basic checks to make sure everything works. They also set access rights and create backups. Automation tools can help reduce setup errors. A good test environment ensures that test results are accurate.

Test execution is the phase where the team runs the test cases. They use the test data and follow the steps written earlier. Each result is checked against the expected outcome. If any defects are found, they are recorded in a tracking tool. The team also performs regression testing to check if new changes cause new issues. Progress is tracked using metrics like pass/fail rates and defect counts. The team can improve this phase by using test tools and prioritising important test cases.

The last phase is test cycle closure. This phase ends the testing process. The team makes sure all planned tests are complete. They check that major defects are fixed or properly recorded. Any skipped tests must be explained. The team collects test data and prepares a summary report. This report shows the testing results, coverage, and known risks. Stakeholders review the report and give approval for release. All test files are saved for future use. Lessons learned are shared to improve the next testing cycle.

5.2.2 Test Cases

This section explains the test case used in the testing process. It acts as a checklist to confirm that all system functions work correctly and meet the given requirements. The test case uses a logic-based approach, where each module is checked using clear and consistent rules. The main modules tested include the user account module, product module, reward module, order module, and others. Each module will be tested for two key functions to review different parts of its performance.

While there are many ways to test software, this phase focuses on functional testing. This method is widely used and helps make sure each module works as planned. For example, the user account module will be tested to check if users can log in and manage their accounts. The product module will be tested to confirm that products are shown properly and users can add, view, or remove items from the system.

The test case follows a standard format that all team members will use. This ensures that the process stays clear and consistent. Each test case includes several parts: Test Case ID, Test Case Name, Designed By, Design Date, Executed By, Execution Date, Short Description, Pre-Conditions, Testing Steps, and Post-Conditions. By focusing on functional testing, the team aims to confirm that every feature works as expected. This helps keep the system reliable and ensures it meets user needs.

User Account Module

Test Case #: UA_001_Create	Test Case Name: Create Account Positive TC 1
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: It provides the access for creation of an account without login. It can be used for create an account for any admin, owner or staff by admin.	

Pre-conditions: -

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Add” button	Display the user account creation form	Pass	
3	Input UserId : test		-	
4	Input Name: Testing Team		-	
5	Input Password: 123456		-	
6	Select Contact No. Prefix: MY+60		-	
7	Input contact No. 182272227		-	
8	Input email: zheheng@gmail.com		-	
9	Select User role: admin		-	
10	Click Create Button	Display Success Message “Record Created”	Pass	

Post-conditions: Systems allows user to login by (user Id and password)
--

Test Case #: UA_002_Create	Test Case Name: Create Account Negative TC 1
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: The handled error when creating user account. (Mandatory field is empty)	

Pre-conditions: -

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Add” button	Display the user account creation form	Pass	
3	Chick Save Button	Userid, name, password, contact no., email address, and user role display error message “Field is required”	Pass	

Post-conditions:

The record is unable to be created and error message will be shown under each field considered mandatory

Test Case #: UA_003_Create		Test Case Name: Create Account Negative TC 3
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: The handled error when creating user account. (Over the field length)		

Pre-conditions:

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Add” button	Display the user account creation form	Pass	
3	Enter email: 12345678900987654321123456 78900987654321123@gmail.co m	Display Error Message “Length cannot more than 50”	Pass	
4	Input Name: Yeoh Zhe Hengssssssssssssssssssssssssss ssssssssssssssssss	Display Error Message “Length cannot more than 50”	Pass	
5	Input UserId: dsdsdssssssssssssssssssssssss ssssssssssssssssssssssssssssss ssss	Display Error Message “Length cannot more than 50”	Pass	

Post-conditions: User account unable to create

Test Case #: UA_004_Create	Test Case Name: Create Account Negative TC 4
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: The handled error when creating user account. (Invalid format)	

Pre-conditions:

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Add” button	Display the user account creation form	Pass	
3	Enter contact No.: 011112342222222222222222 222222222222222222222222 22	Display Error Message “Invalid Phone No. Format”	Pass	
4	Input Email: 123gg.com	Display Error Message “Invalid Email Format”	Pass	

Post-conditions: User account unable to create

Test Case #: UA_005_Login	Test Case Name: User Login Positive Test Case
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: To test user login function with valid username and valid password.	

Pre-conditions: User must created with valid account , Must pass UA_001_CreateAND User is not logged in

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to system	System displays login page	Pass	
2	Enter username :test		Pass	
3	Enter password:123456		Pass	
4	Click login button	User successfully login to the account	Pass	

Post-conditions:

Systems allows user to login and redirect to home page.

Test Case #: UA_006_Login		Test Case Name: User Login Negative Test Case
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: To test user login function with invalid username or invalid password.		

Pre-conditions: User must registered with valid account, Must pass UA_001_Create AND user is not logged in

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to system	System displays login page	Pass	
2	Enter username :test		Pass	
3	Enter password:1234567 (an invalid password)		Pass	
4	Click login button	Display error message “Incorrect Username or password”	Pass	

Post-conditions:

Systems not allows user to login and redirect to home page.

Test Case #: UA_007_Edit	Test Case Name: User Account Update Positive Test Case
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: To test user update function with changing certain field	

Pre-conditions: Must pass UA_001_Create
--

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Edit” button beside ideal be updated record	Display the user account edit form	Pass	
3	Input Name: Testing Teams		-	
4	Click Save Button	Display Success Message “Record Updated”	Pass	

Post-conditions:

Record updated and redirect user to list page

Test Case #: UA_008_Delete	Test Case Name: User Account Delete Positive Test Case
System: StallSync: Intelligent Management	Module: User Account
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Short Description: To test user account delete function with an existence record	

Pre-conditions: An user account record is created

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	Show User account list	Pass	
2	Click “Delete” button beside ideal be removed record	Display the user account delete confirmation dialog	Pass	
3	Select “Confirm” in dialog	Display Success Message “Record Updated”	Pass	

Post-conditions:

Record updated and redirect user to list page

Test Case #: UA_009_ChangePassword		Test Case Name: User Account Change Password Positive Test Case
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: To test user account password changing function with valid inputs		

Pre-conditions: Must pass UA 001 Create

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Change Password Page	System displays Change Password page	Pass	
2	Enter old password: 123456		-	
3	Enter new password: abc123		-	
4	Enter confirm password: abc123		-	
5	Click save button	User successfully change the password	Pass	

Post-conditions:

User password is changed

Test Case Template

Test Case #: UA_010_ChangePassword		Test Case Name: User Account Change Password Negative Test Case 1
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: To test user account password changing function with invalid inputs		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Change Password Page	System displays Change Password page	Pass	
2	Enter old password: 123456		-	
3	Enter new password: abc123		-	
4	Enter confirm password: abc12		-	
5	Click save button	Display error message “New Password & Confirm Password Must be Same.”	Pass	

Post-conditions:

User password unable to change

Test Case #: UA_011_ChangePassword		Test Case Name: User Account Change Password Negative Test Case 2
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: To test user account password changing function with invalid inputs		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Change Password Page	System displays Change Password page	Pass	
2	Enter old password: abcdef		-	
3	Enter new password: abc123		-	
4	Enter confirm password: abc123		-	
5	Click save button	Display error message “Current Password is incorrect”	Pass	

Post-conditions:

User password unable to change

Test Case #: UA_012_ChangePassword		Test Case Name: User Account Reset Password Positive Test Case
System: StallSync: Intelligent Management		Module: User Account
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Short Description: To test user account password changing function with invalid inputs		

Pre-conditions: User must registered with valid account / Must pass UA_001_Login

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to login page	System displays login page	Pass	
2	Click “Forgot Password?”	Display Forgot Password For	-	
3	Enter email: yeohzh-wp22@gmail.com		-	
4	Click Submit button	Show email successfully sent message	Pass	
5				

Post-conditions:

Systems send user backup password through email and direct user back to login page

Test Case #: TC_UserModule_Login_001	Test Case Name: User Login
---	-----------------------------------

Test Case #: UA_013_list	Test Case Name: User Account List, Sort and filter
System: StallSync: Intelligent Management	Module: User Account
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test user account list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to user account page	System displays user account page, with list of account	Pass	
2	Select role = Admin	Shows a list record that role is admin	Pass	
3	Select status = Active	Shows a list record that role is admin and status is active	Pass	
4	Input search = "zh"	Shows a list record that role is admin and status is active, and the userid or name is contain "zh"	Pass	

Post-conditions:

Systems shows a list of user accounts that comply with the sort and filter requirement will be shown

Staff Module

Test Case #: S_001_List		Test Case Name: Staff List, Sort and Filter
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page (/staff)	System displays staff page with list of staff	Pass	
2	input search: a	System displays a list of staff where staffId or name is contain 'a'	Pass	
3	select staff type: admin	System displays a list of staff where staffId or name is contain 'a', and staff type is 'admin'	Pass	
4	select nation: Malaysia	System displays a list of staff where staffId or name is contain 'a', staff type is 'admin' and nationality is malaysia	Pass	
5	select staff status: yes	System displays a list of staff where staffId or name is contain 'a', staff type is 'admin', nationality is malaysia and staff status is yes	Pass	

Post-conditions: A list of staff records that comply with the requirement of sort and filter will be shown

Test Case #: S_002_Create	Test Case Name: Staff create test case positive
System: StallSync: Intelligent Management	Module: Staff
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff create function with the valid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Profile picture: Select a file	System display file uploaded	Pass	
4	Input Name: Tan Ah Kao	-	Pass	
5	input email: zennnmgg@gmail.com		Pass	
6	select any user type		Pass	
7	select any user account		Pass	
8	select any merchants		Pass	
9	select join: 22/06/2025		Pass	
10	select any bank		Pass	
11	input bank username: Tan Ah Kao		Pass	
12	input bank account: 2134544554		Pass	

13	select any contact no prefix		Pass	
14	input current contact no.: 123456789		Pass	
15	input emergency contact no.: 123456666		Pass	
16	select any nationality		Pass	
17	select any id type		Pass	
18	input id number: 0808081400909		Pass	
19	Input line address 1: 78, Jalan Ghandi		Pass	
20	Input line address 2: Bandar Tun Razak		Pass	
21	Input postcode: 53500		Pass	
22	Input City: Setapak		Pass	
23	Input State: Kuala Lumpur		Pass	
24	Switch Same Address: Yes (Green)	The Home address line 1, home address 2, home address postcode, home address city and home address state should be same accordingly	Pass	
25	Click save button	System shows successful message	Pass	

Post-conditions:

The staff record is created and saved into database

Test Case #: S_003_Create	Test Case Name: Staff create test case negative
System: StallSync: Intelligent Management	Module: Staff
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff create function with the no input	

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: S_004_Create	Test Case Name: Staff create test case negative 2
System: StallSync: Intelligent Management	Module: Staff
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff create function with the invalid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input bank account: 1234567890123456789012345		Pass	
4	Click save button	The error message show under the “bank account” Field length cannot more than 25	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: S_005_Edit		Test Case Name: Staff update test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input bank account: 2130021700		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: S_006_Delete		Test Case Name: Staff delete test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Merchant Module

Test Case #: MCH_001_List		Test Case Name: Merchant List, Sort and Filter
System: StallSync: Intelligent Management		Module: Merchant
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test merchant list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page (/merchant)	System displays merchant page with list of merchant	Pass	
2	input search: a	System displays a list of merchant where merchantId or name is contain 'a'	Pass	
3	select merchant status: yes	System displays a list of merchant where merchantId or name is contain 'a', and merchant status is 'yes'	Pass	

Post-conditions: A list of merchant records that comply with the requirement of sort and filter will be shown
--

Test Case #: MCH_002_Create		Test Case Name: Merchant create test case positive
System: StallSync: Intelligent Management		Module: Merchant
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test merchant create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page	System displays merchant page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Profile picture: Select a file	System display file uploaded	Pass	
4	upload store profile picture	System shows the uploaded file	Pass	
5	upload Store front picture	System shows the uploaded file	Pass	
6	Input merchant id: MM		Pass	
7	input name: Masakan Malaysia		Pass	
8	input description: This is malaysian taste		Pass	
9	input ssm no: A2021309939		Pass	
10	upload SSM Cert	System shows the uploaded file	Pass	
11	Select Join Date: 20/06/2025		Pass	
12	Select bank: CIMB		Pass	
13	Input bank account no.:		Pass	

	2189929292			
14	input bank username: MM Sdn Bhd		Pass	
15	Select type (any)		Pass	
16	Click save button	System shows successful message	Pass	

Post-conditions:

The merchant record is created and saved into database

Test Case #: MCH_003_Create		Test Case Name: merchantcreate test case negative
System: StallSync: Intelligent Management		Module: merchant
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test merchant create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page	System displays merchant page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: MCH_004_Create	Test Case Name: merchant create test case negative 2
System: StallSync: Intelligent Management	Module: merchant
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test merchant create function with the invalid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page	System displays merchant page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input bank account: 12345678901234567890123456		Pass	
4	Click save button	The error message show under the “bank account” Field length cannot more than 25	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: MCH_005_Edit	Test Case Name: merchant update test case positive
System: StallSync: Intelligent Management	Module: merchant
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test merchant update function with the valid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page	System displays merchant page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input bank account: 2130021799		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: MCH_006_Delete	Test Case Name: merchant delete test case positive
System: StallSync: Intelligent Management	Module: merchant
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test merchant delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to merchant page	System displays merchant page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions: Systems deleted the record from the database

Announcement Module

Test Case #: A_001_List	Test Case Name: Announcement List, Sort and Filter
System: StallSync: Intelligent Management	Module: Announcement
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Announcement list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Announcement page (/announcement)	System displays Announcement Page with list of announcement	Pass	
2	input search: a	System displays a list of Announcement where announcementId or title is contain 'a'	Pass	

Post-conditions: A list of Announcement records that comply with the requirement of sort and filter will be shown
--

Test Case #: A_002_Create		Test Case Name: Announcement create test case positive
System: StallSync: Intelligent Management		Module: Announcement
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Announcement create function with the valid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Announcement page	System displays Announcement page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input announcement title: Happy Christmas		Pass	
4	Select Announcement Type: Event		Pass	
5	Input Announcement Message: Let’s Celebrate the Stall Sync Anniversary		Pass	
6	Upload Attachment	System shows uploaded attachment	Pass	
7	Click save button	System shows successful message	Pass	

Post-conditions:

The **Announcement** record is created and saved into database

Test Case #: A_003_Create		Test Case Name: Announcement create test case negative
System: StallSync: Intelligent Management		Module: Announcement
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Announcement create function with the no input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Announcement page	System displays Announcement page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: A_004_Create		Test Case Name: Announcement create test case negative 2
System: StallSync: Intelligent Management		Module: Announcement
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Announcement create function with the invalid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Announcement page	System displays Announcement page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input announcement title: 12345678909876543234567890-09876543212345678909876543234567890-09876	The error message show under the “announcement title” Field length cannot more than 50”	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: A_005_Edit	Test Case Name: Announcement update test case positive
System: StallSync: Intelligent Management	Module: Announcement
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff Announcement function with the valid input	

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input Announcement title: Happy StallSync Anniversary		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: A_006_Delete	Test Case Name: Announcement delete test case positive
System: StallSync: Intelligent Management	Module: Announcement
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Announcement delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Announcement page	System displays Announcement page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Product Module

Test Case #: P_001_List		Test Case Name: Product List, Sort and Filter
System: StallSync: Intelligent Management		Module: Product
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page (/product)	System displays Product page with list of product	Pass	
2	input search: a	System displays a list of Product where productId or name is contain 'a'	Pass	
3	select product type: bread	System displays a list of Product where productId or name is contain 'a', and product type is bread	Pass	
4	select product category: Korean	System displays a list of Product where productId or name is contain 'a', product type is bread and product category is korean	Pass	
5	select product status: yes	System displays a list of staff Product productId or name is contain 'a', product type is bread, product category is korean and staff status is yes	Pass	

Post-conditions: A list of product records that comply with the requirement of sort and filter will be shown

Test Case #: P_002_Create		Test Case Name: Product create test case positive
System: StallSync: Intelligent Management		Module: Product
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Product create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page	System displays Product page	Pass	
2	click the add button	System display the “add” form	Pass	
3	input product name: Nasi Lemak		Pass	
4	input Description: Served with egg, cucumber, sambal, peanuts and rice		Pass	
5	select product category: Local		Pass	
6	select product type: Rice		Pass	
7	select merchant: MM		Pass	
8	Upload Product image	System shows the upload image	Pass	
9	Input Product price: 5.99		Pass	
10	Switch able to takeaway: yes (green)		Pass	
11	input takeaway charges: 0.10		Pass	
12	Click save button	System shows successful message	Pass	

Post-conditions:

The **Product** record is created and saved into database

Test Case #: P_003_Create	Test Case Name: Product create test case negative
System: StallSync: Intelligent Management	Module: Product
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff Product function with the no input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page	System displays Product page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: P_004_Create	Test Case Name: Product create test case negative 2
System: StallSync: Intelligent Management	Module: Product
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Product create function with the invalid input	

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page	System displays Product page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input product price: 9999999999		Pass	
4	Click save button	The error message show under the Product Price” Invalid Value Length - Maximum 999999999.99 Characters	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: P_005_Edit	Test Case Name: Product update test case positive
System: StallSync: Intelligent Management	Module: Product
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Product update function with the valid input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page	System displays Product page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input product price: 6.99		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: P_006_Delete		Test Case Name: Staff Product test case positive
System: StallSync: Intelligent Management		Module: Product
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Product delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Product page	System displays Product page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Inventory Module

Test Case #: INV_001_List	Test Case Name: Inventory List, Sort and Filter
System: StallSync: Intelligent Management	Module: Inventory
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Inventory list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the first row product	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click stock out tab	System shows a list of stock out history	Pass	
5	select from date (20/06/2025)	System display a list of record (Stock out) from 20/06/2025	Pass	
6	Select to date (20/07/2025)	System display a list of record (stock out) from 20/07/2025	Pass	

Post-conditions: A list of inventory records that comply with the requirement of sort and filter will be shown

Test Case #: S_002_Create		Test Case Name: Roles create test case positive
System: StallSync: Intelligent Management		Module: Roles
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the product that is countable	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click the add button	System display the “add” form	Pass	
5	input vendor name: AB Supply Sdn Bhd		Pass	
6	input stock price :129.90		Pass	
7	input quantity: 100		Pass	
8	Click save button	System shows successful message	Pass	

Post-conditions:

The Inventory record is created and saved into database, direct user back to product detail

Test Case #: RO_003_Create		Test Case Name: Inventory create test case negative
System: StallSync: Intelligent Management		Module: Inventory
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Inventory create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the product that is countable	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click the add button	System display the “add” form	Pass	
5	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: RO_004_Create		Test Case Name: Inventory create test case negative 2
System: StallSync: Intelligent Management		Module: Inventory
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Inventory create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the product that is countable	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click the add button	System display the “add” form	Pass	
5	User input Quantity: 9999999999		Pass	
6	Click save button	The error message show under the quantity” Invalid value Length - Maximum 999999999	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: RO_005_Edit		Test Case Name: Inventory update test case positive
System: StallSync: Intelligent Management		Module: Inventory
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Inventory update function with the valid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the product that is countable	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click the edit button beside ideal be updated record	System display the detail of the inventory record	Pass	
5	User input quantity: 1000		Pass	
6	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: RO_006_Delete		Test Case Name: Inventory delete test case positive
System: StallSync: Intelligent Management		Module: Inventory
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Inventory delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to product page (/product)	System displays product page with list of product	Pass	
2	click the view button for the product that is countable	System displays the detail of the product record and a tab of switching	Pass	
3	click Stock in tab	System shows a list of stock in history	Pass	
4	click the delete button beside ideal be deleted record	System display the confirmation message	Pass	
5	Click confirm	System display deleted successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Reward Module

Test Case #: R_001_List		Test Case Name: Reward List, Sort and Filter
System: StallSync: Intelligent Management		Module: Reward
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Reward list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page (/reward)	System displays Reward page with list of Reward	Pass	
2	input search: a	System displays a list of Reward where rewardId, description or name is contain 'a'	Pass	
3	select reward status: Available	System displays a list of Reward where rewardId, description or name is contain 'a', and reward status is Available	Pass	
4	select discount type: Percentage	System displays a list of Reward where rewardId description or name is contain 'a', reward status is Available and discount type is percentage	Pass	

Post-conditions: A list of Reward records that comply with the requirement of sort and filter will be shown

Test Case #: R_002_Create		Test Case Name: Reward create test case positive
System: StallSync: Intelligent Management		Module: Reward
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Reward create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page	System displays Reward page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Reward Code: AugDeal		Pass	
4	Input Reward Name: August Deal		Pass	
5	Input Description: August Sale for every food		Pass	
6	Select reward Type: Percentage		Pass	
7	Input Discount value: 0.2		Pass	
8	Input Quantity: 100		Pass	
9	select from date: 01/08/2025		Pass	
10	select to date: 31/08/2025		Pass	
11	Click save button	System shows successful message	Pass	

Post-conditions:

The staff record is created and saved into database

Test Case #: R_003_Create	Test Case Name: Reward create test case negative
System: StallSync: Intelligent Management	Module: Reward
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Reward create function with the no input	

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page	System displays Reward page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: R_004_Create		Test Case Name: Reward create test case negative 2
System: StallSync: Intelligent Management		Module: Reward
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Reward create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page	System displays Reward page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input Quantity: 9999999999		Pass	
4	Click save button	The error message show under the “quantity” Invalid value Length - Maximum 999999999	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: R_005_Edit		Test Case Name: Reward update test case positive
System: StallSync: Intelligent Management		Module: Reward
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Reward update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page	System displays Reward page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input quantity: 1000		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: R_006_Delete		Test Case Name: Reward delete test case positive
System: StallSync: Intelligent Management		Module: Reward
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Reward delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Reward page	System displays Reward page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Member Module

Test Case #: M_001_List	Test Case Name: Member List, Sort and Filter
System: StallSync: Intelligent Management	Module: Member
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Member list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Member page (/mbrProfile)	System displays Member page with list of Member	Pass	
2	input search: a	System displays a list of Member where member id, name or email is contain 'a'	Pass	
3	select member type: bronze	System displays a list of Member where member id, name or email is contain 'a', and member type is bronze	Pass	

Post-conditions: A list of member records that comply with the requirement of sort and filter will be shown

Test Case #: M_002_Edit		Test Case Name: Member update test case positive
System: StallSync: Intelligent Management		Module: Member
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Member update function with the valid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Member page	System displays Member page	Pass	
2	click the edit button in a row with the ideal to be removed record	System display the form with the records	Pass	
3	User input member name: Joyce		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Order Module

Test Case #: O_001_List		Test Case Name: Order List, Sort and Filter
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Order list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page (/order)	System displays Order page with list of order	Pass	
2	input Order Phone No.: 68	System displays a list of Order where phone number is contain ‘68’	Pass	
3	select Merchants: MM	System displays a list of order phone number is contain ‘68’, and merchant is ‘MM’	Pass	
4	select from date: 20/06/2025	System displays a list of order phone number is contain ‘68’, merchant is ‘MM’, and order from 20/06/2025 until today	Pass	
5	select to date: 30/06/2025	System displays a list of order phone number is contain ‘68’, merchant is ‘MM’, and order from 20/06/2025 until 30/06/2025	Pass	

Post-conditions: A list of Order records that comply with the requirement of sort and filter will be shown

Test Case #: O_002_Edit		Test Case Name: Order update test case positive
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Order update function when it is 'New' Status		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin, order status is New

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page	System displays order page	Pass	
2	click the view button in a row with the ideal to be updated status record	System display the invoice of the order	Pass	
3	Click 'Update Status'	The status is move to Paid and shows successful message	Pass	

Post-conditions:

System updated the order status

Test Case #: O_003_Edit		Test Case Name: Order update test case positive
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Order update function when it is ‘Paid’		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin, order status is paid

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page	System displays order page	Pass	
2	click the view button in a row with the ideal to be updated status record	System display the invoice of the order	Pass	
3	Click ‘Update Status’	The status is move to Preparing and shows successful message	Pass	

Post-conditions:

System updated the order status

Test Case #: O_004_Edit		Test Case Name: Order update test case positive
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Order update function when it is ‘Preparing’		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin, order status is preparing

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page	System displays order page	Pass	
2	click the view button in a row with the ideal to be updated status record	System display the invoice of the order	Pass	
3	Click ‘Update Status’	The status is move to Completed, shows successful message and the update button will be disappeared	Pass	

Post-conditions:

System updated the order status

Test Case #: O_005_Download		Test Case Name: Order download test case positive
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To Order staff download function		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page	System displays order page	Pass	
2	click the view button in a row with the ideal to be download record	System display invoice of the order	Pass	
3	Click download button	Show print preview	Pass	
4	Click the print button in the preview screen	The invoice is downloaded successfully	Pass	

Post-conditions: System download the invoice to current using local machine

Transaction Module

Test Case #: T_001_List		Test Case Name: Transaction List, Sort and Filter
System: StallSync: Intelligent Management		Module: Order
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Transaction list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to order page (/order)	System displays Order page with list of order	Pass	
2	Click the more button and choose the transaction option	System displays Transaction page with list of Transaction for relevant order		
3	input Transaction Id: 1	System displays a list of Transaction where transaction id is contain '1'	Pass	
4	select Transaction Status: C - Completed	System displays a list of transaction id is contain '1', and Transaction status is C - Completed	Pass	
5	select from date: 20/06/2025	System displays a list of transaction id is contain '1', and Transaction status is C - Completed, and transaction from 20/06/2025 until today	Pass	
6	select to date: 30/06/2025	System displays a list of transaction id is contain '1', and Transaction status is C - Completed, and transaction from	Pass	

	20/06/2025 until 30/06/2025		
--	-----------------------------	--	--

Post-conditions: A list of Transactions records that comply with the requirement of sort and filter will be shown

General Type Module (Configuration)

Test Case #: GT_001_List	Test Case Name: General Type List, Sort and Filter
System: StallSync: Intelligent Management	Module: General Type
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test General Type list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page (/generalParameter)	System displays General Type page with list of General Type	Pass	
2	input search: a	System displays a list of General Type where General Type or description is contain 'a'	Pass	
3	select category: Business	System displays a list of General Type where General Type or description is contain 'a', and category is Business	Pass	

Post-conditions: A list of General Type records that comply with the requirement of sort and filter will be shown
--

Test Case #: GT_002_Create		Test Case Name: General Type create test case positive
System: StallSync: Intelligent Management		Module: General Type
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Type create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input General type code: TEST		Pass	
4	input general type description: Testing		Pass	
5	input length: 10		Pass	
6	select mandatory/optional: mandatory		Pass	
7	select category: business		Pass	
8	Click save button	System shows successful message	Pass	

Post-conditions:

The General Type record is created and saved into database

Test Case #: GT_003_Create		Test Case Name: General Type create test case negative
System: StallSync: Intelligent Management		Module: General Type
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Type create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: GT_004_Create		Test Case Name: General Type create test case negative 2
System: StallSync: Intelligent Management		Module: General Type
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Type create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input General Type Code: ABCDEFGHIJKLM		Pass	
4	Click save button	The error message show under the general type code” Length cannot more than 10	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: GT_005_Edit		Test Case Name: General Type update test case positive
System: StallSync: Intelligent Management		Module: General Type
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Type update function with the valid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input description: Testing		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: GT_006_Delete		Test Case Name: General Type delete test case positive
System: StallSync: Intelligent Management		Module: General Type
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Type delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

General Code Module (Configuration)

Test Case #: GC_001_List	Test Case Name: General Code List, Sort and Filter
System: StallSync: Intelligent Management	Module: General Code
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test General Code list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	input search: a	System displays a list of General Code where General Code or description is contain 'a'	Pass	

Post-conditions: A list of General Code records that comply with the requirement of sort and filter will be shown

Test Case #: GT_002_Create		Test Case Name: General Code create test case positive
System: StallSync: Intelligent Management		Module: General Code
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Code create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	click the add button	System display the “add” form	Pass	
5	Input General Code: T		Pass	
6	Input General Code Description: Test		Pass	
7	Click save button	System shows successful message	Pass	

Post-conditions:

The staff record is created and saved into database

Test Case #: GT_003_Create		Test Case Name: General Code create test case negative
System: StallSync: Intelligent Management		Module: General Code
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Code create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	click the add button	System display the “add” form	Pass	
5	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: GT_004_Create		Test Case Name: General Code create test case negative 2
System: StallSync: Intelligent Management		Module: General Code
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Code create function with the invalid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	click the add button	System display the “add” form	Pass	
5	User input General Code: ABCDEFGHIJKLM		Pass	
6	Click save button	The error message show under the “General Code” Length Cannot more than 10	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: GT_005_Edit		Test Case Name: General Code update test case positive
System: StallSync: Intelligent Management		Module: General Code
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test General Code update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
5	User input general code description: Testing and Testing		Pass	
6	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: GT_006_Delete	Test Case Name: General Code delete test case positive
System: StallSync: Intelligent Management	Module: General Code
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test General Code delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to General Type page	System displays General Type page	Pass	
2	click the more button and select General Code	System display list of general code that is related to the select general type	Pass	
3	Navigate to General Code page (/generalParameter/generalCode)	System displays General Code page with list of General Code	Pass	
4	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
5	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Functions Module (Configuration)

Test Case #: F_001_List	Test Case Name: Function List, Sort and Filter
System: StallSync: Intelligent Management	Module: Function
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Function list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page (/functions)	System displays Function page with list of Function	Pass	
2	input search: a	System displays a list of Function where function code or description is contain 'a'	Pass	
3	select Function Group: Parameter	System displays a list of Function where function code or description is contain 'a', and Function Group is Parameter	Pass	

Post-conditions: A list of **Function** records that comply with the requirement of sort and filter will be shown

Test Case #: F_002_Create		Test Case Name: Function create test case positive
System: StallSync: Intelligent Management		Module: Function
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Function create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page	System displays Function page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input function code: TEST		Pass	
4	input description: Testing		Pass	
5	select function group: TEST		Pass	
6	Select Action: Add, View, Delete, Edit		Pass	
7	Click save button	System shows successful message	Pass	

Post-conditions:

The staff record is created and saved into database

Test Case #: F_003_Create		Test Case Name: Function create test case negative
System: StallSync: Intelligent Management		Module: Function
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Function create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page	System displays Function page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: F_004_Create		Test Case Name: Function create test case negative 2
System: StallSync: Intelligent Management		Module: Function
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Function create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page	System displays Function page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input Function Code: 12345678901		Pass	
4	Click save button	The error message show under the Function Code “Field length cannot more than 10”	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: F_005_Edit		Test Case Name: Function update test case positive
System: StallSync: Intelligent Management		Module: Function
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Function update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page	System displays Function page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input Description: Testing 123		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: F_006_Delete		Test Case Name: Function delete test case positive
System: StallSync: Intelligent Management		Module: Function
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Function delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Function page	System displays Function page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Roles Module (Configuration)

Test Case #: RO_001_List	Test Case Name: Roles List, Sort and Filter
System: StallSync: Intelligent Management	Module: Roles
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Roles list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page (/roleCode)	System displays Roles page with list of staff	Pass	
2	input search: a	System displays a list of staff where staffId or name is contain 'a'	Pass	

Post-conditions: A list of Roles records that comply with the requirement of sort and filter will be shown

Test Case #: RO_002_Create		Test Case Name: Roles create test case positive
System: StallSync: Intelligent Management		Module: Roles
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page	System displays Roles page	Pass	
2	click the add button	System display the “add” form	Pass	
3	input user role: STF		Pass	
4	input description: Staff		Pass	
5	input local description: local staff		Pass	
6	Click save button	System shows successful message	Pass	

Post-conditions:

The Roles record is created and saved into database

Test Case #: RO_003_Create		Test Case Name: Roles create test case negative
System: StallSync: Intelligent Management		Module: Roles
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page	System displays Roles page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: RO_004_Create		Test Case Name: Roles create test case negative 2
System: StallSync: Intelligent Management		Module: Roles
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page	System displays Roles page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input user role: ABCDEFGHIJKL		Pass	
4	Click save button	The error message show under the User role”Invalid value length - Maximum 10 character”	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: RO_005_Edit		Test Case Name: Roles update test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles update function with the valid input		

Pre-conditions: User must login/ pass the UA_005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page	System displays Roles page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input description: Testing		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: RO_006_Delete		Test Case Name: Roles delete test case positive
System: StallSync: Intelligent Management		Module: Roles
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Roles delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Roles page	System displays Roles page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions: Systems deleted the record from the database

Password Policy Module (Configuration)

Test Case #: PP_001_List		Test Case Name: Staff List, Sort and Filter
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff list function with the optional of sort and filter		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page (/staff)	System displays staff page with list of staff	Pass	
2	input search: a	System displays a list of staff where staffId or name is contain 'a'	Pass	
3	select staff type: admin	System displays a list of staff where staffId or name is contain 'a', and staff type is 'admin'	Pass	
4	select nation: Malaysia	System displays a list of staff where staffId or name is contain 'a', staff type is 'admin' and nationality is malaysia	Pass	
5	select staff status: yes	System displays a list of staff where staffId or name is contain 'a', staff type is 'admin', nationality is malaysia and staff status is yes	Pass	

Post-conditions: A list of staff records that comply with the requirement of sort and filter will be shown

Test Case #: S_002_Create		Test Case Name: Staff create test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Profile picture: Select a file	System display file uploaded	Pass	
4	Input Name: Tan Ah Kao	-	Pass	
5	input email: zennnmogg@.com		Pass	
6	select any user type		Pass	
7	select any user account		Pass	
8	select any merchants		Pass	
9	select join: 22/06/2025		Pass	
10	select any bank		Pass	
11	input bank username: Tan Ah Kao		Pass	
12	input bank account: 2134544554		Pass	
13	select any contact no prefix		Pass	

14	input current contact no.: 123456789		Pass	
15	input emergency contact no.: 123456666		Pass	
16	select any nationality		Pass	
17	select any id type		Pass	
18	input id number: 0808081400909		Pass	
19	Input line address 1: 78, Jalan Ghandi		Pass	
20	Input line address 2: Bandar Tun Razak		Pass	
21	Input postcode: 53500		Pass	
22	Input City: Setapak		Pass	
23	Input State: Kuala Lumpur		Pass	
24	Switch Same Address: Yes (Green)	The Home address line 1, home address 2, home address postcode, home address city and home address state should be same accordingly	Pass	
25	Click save button	System shows successful message	Pass	

Post-conditions:

The staff record is created and saved into database

Test Case #: S_003_Create	Test Case Name: Staff create test case negative
System: StallSync: Intelligent Management	Module: Staff
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff create function with the no input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: S_004_Create	Test Case Name: Staff create test case negative 2
System: StallSync: Intelligent Management	Module: Staff
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test staff create function with the invalid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input bank account: 1234567890123456789012345		Pass	
4	Click save button	The error message show under the “bank account” Field length cannot more than 25	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: S_005_Edit		Test Case Name: Staff update test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input bank account: 2130021700		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: S_006_Delete		Test Case Name: Staff delete test case positive
System: StallSync: Intelligent Management		Module: Staff
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test staff delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to staff page	System displays staff page	Pass	
2	click the delete button in a row with the ideal to be updated record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions: Systems deleted the record from the database

System Table Module (Configuration)

Test Case #: ST_001_List	Test Case Name: File Management List, Sort and Filter
System: StallSync: Intelligent Management	Module: File Management
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Management list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page (/fileManagements)	System displays File Management page with list of File Management	Pass	
2	input search: a	System displays a list of File Management where file name or file description is contain 'a'	Pass	

Post-conditions: A list of File Management records that comply with the requirement of sort and filter will be shown

Test Case #: ST_002_Create	Test Case Name: File Management create test case positive
System: StallSync: Intelligent Management	Module: File Management
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Management create function with the valid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input file name: test		Pass	
4	Input file description: Testing File		Pass	
5	Select File Type: Functional		Pass	
6	Click save button	System shows successful message	Pass	

Post-conditions:

The File Management record is created and saved into database

Test Case #: ST_003_Create	Test Case Name: File Management create test case negative
System: StallSync: Intelligent Management	Module: File Management
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Management create function with the no input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: ST_004_Create		Test Case Name: File Management create test case negative 2
System: StallSync: Intelligent Management		Module: File Management
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test File Management create function with the invalid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	click the add button	System display the “add” form	Pass	
3	User input file name: testinggogogofgrgtui1234456		Pass	
4	Click save button	The error message show under the file name ” Field length cannot more than 25	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: ST_005_Edit	Test Case Name: File Management update test case positive
System: StallSync: Intelligent Management	Module: File Management
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Management update function with the valid input	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input file description: testing 2		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: ST_006_Delete	Test Case Name: File Management delete test case positive
System: StallSync: Intelligent Management	Module: File Management
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Management delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

System Table Key Module (Configuration)

Test Case #: STK_001_List	Test Case Name: File Key List, Sort and Filter
System: StallSync: Intelligent Management	Module: File Key
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test File Key list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Key page (/fileManagements/tableKeys)	System displays File Key page with list of File Key	Pass	
2	input search: a	System displays a list of File Key where file name or file description is contain 'a'	Pass	

Post-conditions: A list of File Key records that comply with the requirement of sort and filter will be shown

Test Case #: STK_002_Create		Test Case Name: File Key create test case positive
System: StallSync: Intelligent Management		Module: File Key
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test File Key create function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
	Click the more button beside any record, then choose table key	System displays File Key page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Select file key: psusrnumm		Pass	
4	Input file key sequence: 1		Pass	
6	Click save button	System shows successful message	Pass	

Post-conditions:

The File Key record is created and saved into database

Test Case #: STK_003_Create		Test Case Name: File Key create test case negative
System: StallSync: Intelligent Management		Module: File Key
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test File Key create function with the no input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	Click the more button beside any record, then choose table key	System displays File Key page	Pass	
3	click the add button	System display the “add” form	Pass	
4	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: STK_004_Edit		Test Case Name: File Key update test case positive
System: StallSync: Intelligent Management		Module: File Key
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test File Key update function with the valid input		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	Click the more button beside any record, then choose table key	System displays File Key page	Pass	
3	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
4	User input file key sequence: 3		Pass	
5	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: ST_005_Delete		Test Case Name: File Key delete test case positive
System: StallSync: Intelligent Management		Module: File Key
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test File Key delete function with the valid input		

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to File Management page	System displays File Management page	Pass	
2	Click the more button beside any record, then choose table key	System displays File Key page	Pass	
3	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
4	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

System Calendar Module (Configuration)

Test Case #: C_001_Create		Test Case Name: System Calendar (Annual Variable) create test case positive
System: StallSync: Intelligent Management		Module: System Calendar
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test System Calendar create function with the valid input		

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Variable PH tabs	System displays System Calendar (Annual Variable) page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Holiday Date: Select (2/7/2025)		Pass	
4	Input No. of days: 2		Pass	
5	input Description: Testing holiday		Pass	
6	Click save button	System shows successful message	Pass	

Post-conditions:

The **System Calendar (Annual variable)** record is created and saved into database

Test Case #: C_002_Create	Test Case Name: System Calendar (Annual Fixed) create test case positive
System: StallSync: Intelligent Management	Module: System Calendar
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar create function with the valid input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Fixed PH tabs	System displays System Calendar (Annual Fixed) page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Input Holiday Date: Select (2/8/2025)		Pass	
4	input Description: Testing holiday Fixed annual		Pass	
5	Click save button	System shows successful message	Pass	

Post-conditions:

The System Calendar (Annual fixed) record is created and saved into database

Test Case #: C_003_Create	Test Case Name: System Calendar (Annual Variable) create test case negative
System: StallSync: Intelligent Management	Module: System Calendar
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar create function with the no input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Variable PH tabs	System displays System Calendar (Annual Variable) page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: C_004_Create	Test Case Name: System Calendar (Annual Fixed) create test case negative
System: StallSync: Intelligent Management	Module: System Calendar
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar create function with the no input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Fixed PH tabs	System displays System Calendar (Annual Fixed) page	Pass	
2	click the add button	System display the “add” form	Pass	
3	Click save button	System shows the field is required under the mandatory field	Pass	

Post-conditions:

System should shows the error message and record unable to be created

Test Case #: S_005_Edit	Test Case Name: System Calendar (Annual Variable) update test case positive
System: StallSync: Intelligent Management	Module: System Calendar (Annual Variable)
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar (Annual Variable) update function with the valid input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Variable PH tabs	System displays System Calendar (Annual Variable) page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input description: second testing		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: S_006_Edit	Test Case Name: System Calendar (Annual Fixed) update test case positive
System: StallSync: Intelligent Management	Module: System Calendar (Annual Fixed)
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar (Annual Fixed) update function with the valid input	

Pre-conditions: User must login/ pass the UA_005_Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Fixed PH tabs	System displays System Calendar (Annual Fixed) page	Pass	
2	click the edit button in a row with the ideal to be updated record	System display the form with the records	Pass	
3	User input description: second testing Fixed		Pass	
4	Click save button	System shows record updated	Pass	

Post-conditions:

System save the edited record

Test Case #: S_007_Delete	Test Case Name: System Calendar (Annual Variable) delete test case positive
System: StallSync: Intelligent Management	Module: System Calendar (Annual Variable)
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar (Annual Variable) delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual Variable PH tabs	System displays System Calendar (Annual Variable) page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Test Case #: S_008_Delete	Test Case Name: System Calendar (Annual Fixed) delete test case positive
System: StallSync: Intelligent Management	Module: System Calendar (Annual Fixed)
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test System Calendar (Annual Fixed) delete function with the valid input	

Pre-conditions: User must registered with valid account / Must pass TC_UserModule_Register_001

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to System Calendar page (/calendar), navigate Annual FixedPH tabs	System displays System Calendar (Annual Fixed) page	Pass	
2	click the delete button in a row with the ideal to be removed record	System display confirmation message	Pass	
3	Click confirm	Show successful message	Pass	

Post-conditions:

Systems deleted the record from the database

Backup Module (Configuration)

Test Case #: B_001_List	Test Case Name: Backup List, Sort and Filter
System: StallSync: Intelligent Management	Module: Backup
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Backup list function with the optional of sort and filter	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Backup page (/backup)	System displays Backup page with list of Backup	Pass	
2	input search: a	System displays a list of Backup where file is contain 'a'	Pass	
3	select status: Completed	System displays a list of Backup where file is contain 'a' and status is Completed	Pass	
4	select from: 24/06/2025 select to: 26/06/2025	System displays a list of Backup where file is contain 'a', status is downloaded, and record is from 24/06/2025 and to 26/06/2025	Pass	

Post-conditions: A list of **Backup** records that comply with the requirement of sort and filter will be shown

Test Case #: B_002_Generate	Test Case Name: Backup create test case positive
System: StallSync: Intelligent Management	Module: Backup
Executed By: Yeoh Zhe Heng	Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng	Design Date: 22/06/2025
Short Description: To test Backup generate backup function	

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Backup page	System displays Backup page	Pass	
2	click the Backup now button	system display the confirmation message	Pass	
3	Click the confirm button	System display the backup successfully generated	Pass	

Post-conditions:

The Backup record is created and saved

Test Case #: B_003_Download		Test Case Name: Backup download test case positive
System: StallSync: Intelligent Management		Module: Backup
Executed By: Yeoh Zhe Heng		Execution Date: 02/07/2025
Design By: Yeoh Zhe Heng		Design Date: 22/06/2025
Short Description: To test Backup Download function		

Pre-conditions: User must login/ pass the UA 005 Login test case and user is admin

Step	Action	Expected System Response	Pass/Fail	Comments
1	Navigate to Backup page	System displays Backup page	Pass	
2	click the download button beside ideal to download record	System download the sql.enc into local machine	Pass	

Post-conditions:

System should download the file

5.3 Chapter Summary and Evaluation

This chapter explains how the StallSync: Intelligent Management web application was developed. It describes the tools and methods used during the development phase. The goal was to create a complete and working system. The chapter also discusses how the system was tested. It includes a test plan and several test cases. Each test case is linked to a specific feature of the system. The testing section defines the scope, the data used, the test environment, and how results were evaluated. These test cases help to confirm that the StallSync: Intelligent Management System works as expected. This part of the project shows how the system was built and how its quality was checked.

Chapter 6

Discussions and Conclusion

6 Discussions and Conclusion

This chapter is focusing on the evaluation of projects. The chapter includes the summary, achievements, contributions, limitations and future improvements, and issues and solutions.

Summary

StallSync: Intelligent Management is web-based and works to enhance processes in food courts and canteens. A major number of food stalls are still practicing manual processes. These are paper billing, handwritten billing and using spreadsheets. This may result in numerous mistakes, time wastage and improper planning. This is even worse during rush time.

These issues are addressed by StallSync using digital technologies and automation. It puts everything to a single platform. Stall owners, its managers, and administrators have an opportunity to monitor their sales, inventory, people, and make payments in a single place. Customers are also assisted to place orders and get recommendations with the system. It offers a very fluid and an effective business and customer experience.

StallSync is based on machine learning and time series modeling. Such technologies examine the data that was previously collected and provide the forecasts about the future. The system is able to predict the sales, customer demand and the staffing required. This assists the managers in making plans. It also lowers waste and enhances the quality of services.

The system of billing is completely digital. It substitutes cash-based and manual billing. It works with QR code, mobile wallets, and contactless cards. This increases the number of payment means of the customers. It also makes the check out process quick and erroneous.

Staff management is also simplified by StallSync. It monitors the real-time performance and schedule of staff. Managers have the ability to shift on demand. This prevents cases of overstaffing and understaffing. Proper staffing is good both to the customer and the staff.

The system is constructed in the Agile approach to software development. This practice allows adapting to shifting demands. The team has the possibility to improve at any point. Any documentation is changed to be in line with what is referred to as the final product. The software development involves testing, development, analysis, and design. All the steps provide completeness and validity of the system.

To conclude, StallSync is a progressive management program of food stalls. It focuses on integration of data analysis, automation and utilitarian tools. It enhances daily running. It also assists in increasing the size of stalls so that they can remain competitive. StallSync helps make better decisions, it can improve the speed of service, and it can increase satisfaction among all the people.

Achievements

StallSync: Intelligent Management System was well equipped, and the milestones of the project were successfully completed to the maximum rate of completion of 100% as per the requirements of the core objectives of this final year project. The system evidenced great achievements of modernization of functioning of food stalls and canteens and every goal was attained completely by using usage-friendly and innovative functions.

Its initial aim was to give effective data handling. StallSync achieves this objective through automatic organizing of the sales records, inventory information and transaction history. It eliminates problems with manual entry, which in most cases result in errors. The system would involve such dashboards, which display real time data. Sales trends and stock carrying are displayed by managers easily. It also secures sensitive information under secure access controls. The use of digital receipts eliminates time delays in generating the receipts, and the transactions are more reliable.

The second objective was carrying out the predictive analysis through its time series model. StallSync does this by learning from past sales. It is capable of predicting trends in the future like the customer traffic, busy time and product demand. This makes stall owners come up with improved planning. They are able to create stock in advance and plan proper numbers of staff. This saves money, minimizes waste and eases service during rush time.

The third objective was to come up with a full application. This has been achieved. StallSync app is a combination of numerous helpful characteristics. It allows live order tracking, monitoring of sales, updating of stock, and also staffing. These are all at one hassle-free interface. It eliminates the ancient way of paper menus, hand-written logs, and issue of paper bills. This makes operations done daily more efficient, faster and cleaner.

StallSync enhances the experience of the customer too. It is a digital interface where customers can place orders as well as pay. They do not have to wait in line and depend on personnel. This provides a faster service and less error. It also increases convenience in placing orders especially when it is busy.

Small and medium scale stall owners also benefit with the system. It makes them more in control of their business. By using real-time data and intelligent reports, they will be able to know what sells most and when to expect rush-hours. This assists them to have a better service to the customers and expand the business.

Another major accomplishment is the cost saving. StallSync eliminates paper, printing and reporting. It also assists managers to avoid cases of overstaffing or understaffing. Actual demand can be used when planning the staff schedules. This reduces the costs of labor, but the quality of service remains high.

Altogether, StallSync has achieved all its goals. It is an intelligent, integrated management solution of food stalls and canteens. It enhances accuracy of data, forecasts future trends and makes things straightforward. This activity indicates definite achievements and is viable in practical application in food service.

Contributions

The StallSync: Intelligent Management System offers real life solutions to the food and beverage industry nowadays. It has digital tools to address the problems that are experienced by food stalls and canteen operators most of the time. The system is efficient, fast and convenient and all these are high needs in the market today.

The modular design is one of them. This enables various categories of users such as system administrator, merchant and staff to utilize the system according to their work. Every user can view what he requires. This maintains the system ordered as well as safe. It is also easy to scale it down to small stalls or large canteens.

The Staff Module assists in management of employees. Managers are able to allocate shifts, monitor working hours and change schedules. This helps to synchronize billing and staffing levels with that of the customers. Consequently, employees are not so stressed and the level of service rises.

With the Merchant Module, the stall owners are the masters of their business. They are able to edit business pages, change their settings and monitor performance. This assists them to make fast judgments using real time data. This degree of control is significant in a high paced market.

The Review Module presents the feedback written by the customer in connection to the particular stall. Owners will be able to see how many ratings have been cast, average rating and expanded comments. Such reviews can guide companies to strengthen their foods, services and incurring their operations.

Menu items and inventory can be easily controlled with the help of Product and Inventory Modules. The owners could introduce products, edit them or delete them. They are also able to monitor stock on real time. The system utilizes the historical data so as to forecast the future needs. This eliminates shortage of products or surpluses. It also cuts down wastage and saves cost.

Customer loyalty is enabled by the Reward Module. Each purchase helps the customers gain points. They are able to exchange these points in future. This assists the stall owners in retaining their usual customers and increasing business.

This customer information is stored on the Member Module. It enables customized offers and personal promotions. This enhances consumer experience and boosts purchases.

The Order Module covers every kind of the order dine in, take away or delivery. Employees are in a position to update the status of every order. All payments are noted in the Transaction Module. These attributes minimize errors and accelerate the delivery.

The Announcement Module assists in communication. Admins and merchants are able to dispatch an update or promotions to the staff and customers. This makes important messages arrive and end up quickly and clearly.

The other useful modules are Roles Module, System Calendar and Backup Module. The tools enhance security, control access, facilitate timetables, and guard valuable information.

There is a certain purpose to each module. Combined, this forms an entirely complete and easily manipulated system and one that is flexible. The small vendors as well as large canteen operators can use stallSync. It not only boosts the business, but also maintains things in check.

The current market has become fast and digital. The customers desire rapid performances and ease. StallSync fulfills these requirements. It provides business owners with an intelligent means of running their processes. With the shift of the food and beverage industry into the digital era, StallSync is prepared to transform it in that direction with its user-friendly interface and versatile capabilities.

Limitations and Future Improvements

Limitations

Although the variety of features and improvement of the food stall and canteen administration is quite high in the StallSync, there are certain shortcomings at this current stage:

1. Low Offline Availability

StallSync is a web-based solution and requires a stable internet-connection. This can interfere with the performance of the system in areas with a low network connection. It does not provide offline capabilities yet, that can interrupt operations in case of network outage.

2. Non Tech Savvy Learning Curve

Even though the interface is user friendly, certain users, particularly users who are not conversant with digital systems may still need some training to enable them to use all functions. That can hamper the initial adoption process.

3. Third-Party Integration of delivery

At this point StallSync does not have delivery service management built-in. Such businesses that depend on food delivery need to process this differently or third-party solutions, which are not always in line with the system data and reports.

4. Device Dependency

StallSync is a browser based application; hence, depending on the device used to access it (e.g., mobile, tablet or desktop) the performance might differ. Cheap devices could take longer to load and this is especially when they deal with huge data reports or real-time dashboards.

5. Poor Language Facility

The existing product might just have a single language (e.g. English). This may pose a constraint to users that would want to operate in their local languages.

Future Improvements

To overcome these shortcomings and make StallSync even more, there are a few planned changes in future improvement:

1. Support of Offline Mode
A later incarnation can possibly have some offline support where some basic operations (such as taking orders) can proceed even during internet downtimes. Once a connection is established data will automatically sync.
2. On-boarding Training and In-App Training
It is possible to introduce a system of interactive tutorials and built-in help. This will ease the learning curve and confidence in use.
3. IDP API
The system could also be integrated to be connected to the common delivery services (e.g., GrabFood, Foodpanda) in order to simplify delivery app orders entries to minimize manual action, and to increase ordering accuracy.
4. Multi-language Support
The introduction of options of languages spoken will enhance accessibility, particularly in multilingual areas. This makes the staff and customers be friendlier with the system.
5. AI Recommendations and Advanced Analytics
Further improvements can be done with analytics being provided in greater detail (including customer segmentation and product performance analysis). By studying what goods to replenish and offer discounts on, merchants can also achieve their pricing and stock-promotion strategies with the help of AI-based recommendations.
6. Themes of the UI The UI themes are fully customizable.
Allowing them to customize the appearance of the system or theme would be beneficial to the relative usability and individual liking that can be extended in the various businesses.

Issues and Solutions

The development of StallSync: Intelligent Ordering System presented several challenges that tested both my technical skills and personal resilience. One of the major hurdles was learning and integrating multiple new technologies, including React.js, Node.js, Express.js, and MySQL. As these frameworks were unfamiliar to me at the start, I had to invest significant time in understanding their architecture and ensuring seamless integration between the frontend, backend, and database. To overcome this, I adopted a continuous testing approach—systematically validating each module as I progressed to detect integration issues early and maintain system stability.

Another significant challenge was project management under time constraints. Balancing the development workload with academic responsibilities was particularly demanding, especially with a strict project deadline. To stay on track, my teammate and I applied Agile principles, conducting regular meeting to prioritize tasks and monitor progress. This structured approach helped us stay focused, organized, and adaptive to changes while managing overlapping responsibilities.

Working as part of a team introduced its own set of challenges. Although disagreements were minimal, but differences in work styles. We addressed this by fostering open communication and mutual respect, creating an environment where both of us could voice their ideas. This collaborative mindset not only strengthened our team dynamics but also led to better decision-making and more refined solutions.

Reflecting on the entire journey, this project taught me far more than just technical implementation. I developed strong skills in time management, team coordination, and adaptability, all of which are essential traits for a successful developer. Most importantly, I learned to be modest to ask for advice and remain calm under pressure

which the qualities that will continue to guide me in future professional and academic endeavors.

References

- Adilla Faradila, A. Sagaf., & Ichsan, I. (2025). Development of a Mobile Web-Based Food and Beverage Ordering Application in a Youth Cafe With QR Code Technology. INOVTEK Polbeng - Seri Informatika, 10(1), 262–271. <https://doi.org/10.35314/way9vn18>
- Aditya, R., & Suhirman. (2024). Integrated Payment System to Improve Financial Management Using Payment Gateway. INOVTEK Polbeng - Seri Informatika, 9(2), 727–739. <https://doi.org/10.35314/yijrn52>
- Akcam, B. K. (2022). The Impact of Technology on the Starbucks Experience. *Journal of Information Technology Teaching Cases*, 13(1), 1–7.
<https://journals.sagepub.com/doi/full/10.1177/20438869221099305>
- Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. S., Stoica, I., & Zaharia, M. (2016). *Above the Clouds: A Berkeley View of Cloud Computing*. Science. <https://www.semanticscholar.org/paper/Above-the-Clouds%3A-A-Berkeley-View-of-Cloud-Armbrust-Fox/1f7190fc294246f83f1f331cc51e3264851d0d36>
- Asrani, H., Vishwakarma, S., Asrani, D., & Asrani, K. (2024). Point of Sale Systems. International Journal of Innovative Research in Computer Science & Technology (IJIRCST), 12(1), 2347–5552. <https://doi.org/10.55524/CSISTW.2024.12.1.63>
- AWS. (2023). *MongoDB & MySQL — The Difference Between MongoDB and MySQL* — Amazon Web Services, Inc. <https://aws.amazon.com/cn/compare/the-difference-between-mongodb-vs-mysql/>
- Christine, K., & Berlianto, M. (2022). *ANTECEDENT FACTORS AFFECTING REPURCHASE INTENTION ON SHOPEE FOOD* [Review of *ANTECEDENT FACTORS AFFECTING REPURCHASE INTENTION ON SHOPEE FOOD*]. II(01), 12. Institute of Business and Technology Indragiri. <https://journal.itbind.ac.id/index.php/jmbi/article/view/58/51>
- Digitise your restaurant with EasyEat's restaurant management software. (n.d.). Get.easyeat.ai. <https://get.easyeat.ai/>
- FeedMe. (2018). FeedMe Pos; FeedMe Pos Website. <https://feedme.ai/company/about>
- Fini Anjela, P.-A., Megawati, S., & Asep Taryana, S. (2024). CONSUMER PURCHASING BEHAVIOR OF ONLINE FOOD DELIVERY (OFD) APPLICATION USER. Jurnal Ilmu Keluarga Dan Konsumen, 17(2), 169–181. <https://doi.org/10.24156/jikk.2024.17.2.169>
- Fuentes, Greynier. “Council Post: Exploring the Rise of Digital Wallets and Ensuring Customer Satisfaction.” *Forbes*, Aug. 2024, 26.

www.forbes.com/councils/forbesbusinessdevelopmentcouncil/2024/08/26/exploring-the-rise-of-digital-wallets-and-ensuring-customer-satisfaction/

GeeksforGeeks. (2024). ARIMA vs SARIMA Model. *GeeksforGeeks*. <https://doi.org/10.9172/9011/4827>

Ghasemaghaei, M. (2019). Does data analytics use improve firm decision making quality? The role of knowledge sharing and data analytics competency. *Decision Support Systems*, 120, 14–24. ScienceDirect. <https://doi.org/10.1016/j.dss.2019.03.004>

Google Patents. (2001, July 5). Payment processing method and system. Google. <https://patents.google.com/patent/US20020046189A1/en>

Google Patents. (2013, December 31). Cashier system based on cloud computing and mobile internet technology. Google. <https://patents.google.com/patent/CN103745541B/en>

Harnidah , S., Farah Adibah , C. I., Muhammad Shahrim, A. K., Hazrina , G., & Mohd Mursyid , A. (2024). Enhancing Fast-Food Experiences: Customer Roles in Using Self-Service Technology [Review of *Enhancing Fast-Food Experiences: Customer Roles in Using Self-Service Technology*]. *International Journal of Academic Research in Business and Social Sciences*, 14(8), 1198–1208. hrmars. https://hrmars.com/papers_submitted/22120/enhancing-fast-food-experiences-customers-roles-in-using-self-service-technology.pdf

InfluxData. (2024). *Time Series Forecasting Methods*. InfluxData. <https://www.influxdata.com/time-series-forecasting-methods/>

Khadka, N. (2023, April 5). *LightGBM Algorithm: The Key to Winning Machine Learning Competitions - Dataaspirant*. <https://dataaspirant.com/lightgbm-algorithm/>

Khando, K., Islam, M. S., & Gao, S. (2022). The Emerging Technologies of Digital Payments and Associated Challenges: A Systematic Literature Review. *Future Internet*, 15(1), 21. mdpi. <https://doi.org/10.3390/fi15010021>

Kristoffersen, E., Blomsma, F., Mikalef, P., & Li, J. (2020). The smart circular economy: A digital-enabled circular strategies framework for manufacturing companies. *Journal of Business Research*, 120, 241–261. <https://doi.org/10.1016/j.jbusres.2020.07.044>

Lasa, Amaia Noguera, et al. “Performance Management That Puts People First | McKinsey.” [Www.mckinsey.com, 15 May 2024,](https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/in-the-spotlight-performance-management-that-puts-people-first) [www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/in-the-spotlight-performance-management-that-puts-people-first.](https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/in-the-spotlight-performance-management-that-puts-people-first)

Li, F. H., Rogers, L., Mathur, A., Malkin, N., & Chetty, M. (2019). *Keepers of the Machines: Examining How System Administrators Manage Software Updates For Multiple Machines*. SOUPS @ USENIX Security Symposium.

<https://www.semanticscholar.org/paper/Keepers-of-the-Machines%3A-Examining-How-System-For-Li-Rogers/a955b879ffe3f4ce1d842755ecbd3ef401e47171>

Liu, J., Xiao, Y., Chen, H., Ozdemir, S., Diddle, S., & Singh, V. (2010). A Survey of Payment Card Industry Data Security Standard. *IEEE Communications Surveys & Tutorials*, 12(3), 287–303. <https://doi.org/10.1109/surv.2010.031810.00083>

Maier, A., Lupu, M. I., Canja, C. M., Padureanu, C., Vasile Padureanu, & Olaru, A. S. (2024). MANAGE THE CREATION AND PRESENTATION OF BEVERAGE MENUS IN FOODSERVICE OPERATIONS. SWS International Scientific Conference on Social Sciences, 11, 413–420. <https://doi.org/10.35603/sws.iscss.2024/s04/31>

Markos, O. (2020, April 27). Time series forecasting- SARIMA vs Auto ARIMA models. Medium. <https://medium.com/analytics-vidhya/time-series-forecasting-sarima-vs-auto-arima-models-f95c76d71d8f>

Meriam Mahjoub, Asef Mdhaffar, Riadh Ben Halima, & Jmaiel, M. (2011). A Comparative Study of the Current Cloud Computing Technologies and Offers. <https://doi.org/10.1109/ncca.2011.28>

MyGovernment. (2024). MyGOV - The Government of Malaysia's Official Portal. [Www.malaysia.gov.my](https://www.malaysia.gov.my/portal/content/654). <https://www.malaysia.gov.my/portal/content/654>

Nandaniya, H. Technical Feasibility in Software: Types, Benefits, and Conducting Methods. (2024). Maruti Techlabs. <https://marutitech.com/technical-feasibility-in-software-engineering/>

Pandey, S., Quraishi, R., Salian, A., & Bhagat, S. (2024). Development and Evaluation of a Comprehensive Web-Based Canteen Food Ordering System. International Journal for Research in Applied Science and Engineering Technology, 12(11), 316–324. <https://doi.org/10.22214/ijraset.2024.65046>

Patil, H. B., Rathi, M. M., Wankhade, V. G., Mrunali Darokar, Yash Wankhade, & Neerja Dharmale. (2024). Elevating Canteen Management with a Modern Web Solution. 4, 1–6. <https://doi.org/10.1109/scecs61402.2024.10482188>

PINTILIUC, I.-G. (2018). Protection of personal data. *Logos Universality Mentality Education Novelty: Law*, 6(1), 37–40. <https://doi.org/10.18662/lumenlaw/05>

Prasad, E. (2021, May 20). What is Auto-ARIMA? Featurepreneur. <https://medium.com/featurepreneur/what-is-auto-arima-b8025c6d732d>

- Prosimo. (2024, September 27). *Prosimo*. Prosimo. <https://prosimo.io/why-is-cloud-computing-cost-effective/>
- Ram, K. (2013). Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, 8(1). <https://doi.org/10.1186/1751-0473-8-7>
- Research and Markets (2020). Cloud kitchen market by type, product type and nature: Global opportunity analysis and industry forecast, 2021-2027. Research and Markets. <https://www.researchandmarkets.com/r/vphvtl>.
- Samengon, H., Ishak, F. A. C., Ab Karim, M. S., Ghazali, H., & Arshad, M. M. Enhancing Fast-Food Experiences: Customer Roles in Using Self-Service Technology. <https://hrmars.com/index.php/IJARBSS/article/view/22120/Enhancing-Fast-Food-Experiences-Customers-Roles-in-Using-Self-Service-Technology>
- Samuel, O. (2023, November 6). *Top 10 Popular Web Development Stacks You Should Know in 2023*. Medium. <https://medium.com/@technophile/top-10-popular-web-development-stacks-you-should-know-in-2023-e0c4e4688c86>
- ShopeeFood (2024). Shopeefood.com.my. <https://www.shopeefood.com.my/merchants/shopee-partner>
- Simuka, J., Marume, T., Mzembu, T., & Tagwi, S. (2024). Role of Technology to Improve the Effectiveness of University Canteen Systems. *Journal of Business and Econometrics Studies*, 1–6. <https://doi.org/10.61440/jbes.2024.v1.13>
- Statista Research Department. (2024, December 4). *Digital payments in Southeast Asia - statistics & facts*. Statista.
- <https://www-statista-com.tarc.idm.oclc.org/topics/9451/digital-payments-in-southeast-asia/#topicOverview>
- Sufyan bin Uzayr. (2021). *Introduction to Visual Studio Code*. Semantic Scholar. <https://www.semanticscholar.org/paper/Introduction-to-Visual-Studio-Code-Uzayr/70f34c5e4041aedfbf8cc1be075c2638143b8c80>
- Türkmen, G., Sezen, A., & Şengül, G. (2024). Comparative Analysis of Programming Languages Utilized in Artificial Intelligence Applications: Features, Performance, and Suitability. *International Journal of Computational and Experimental Science and Engineering*, 10(3). <https://doi.org/10.22399/ijcesen.342>
- Uriawan, W., Faroj, R. Z., Hadid, R. A., Khoirunnisa, S., Julianto, S., & Sopian, A. R. (2024, July 2). Innovative Data Management Strategies in Point of Sale Application Development: Increasing Business Productivity. Preprints.org. <https://doi.org/10.20944/preprints202407.0241.v1>

Wikipedia Contributors. (2019, August 16). *Mean absolute error*. Wikipedia; Wikimedia Foundation.

https://en.wikipedia.org/wiki/Mean_absolute_error

Wilson, R. (2020, June 22). *How to Estimate the Cost of Software Development*. SphereGen.

<https://www.spherenet.com/cost-of-software-development/>

XGBoost Advantages and Disadvantages (pros vs cons) | XGBoosting. (2024). Xgboosting.com.

<https://xgboosting.com/xgboost-advantages-and-disadvantages-pros-vs-cons/>

Yutika, F. (2023). FACTORS INFLUENCING COFFEE SHOP ENTREPRENEUR' INTENTION USING

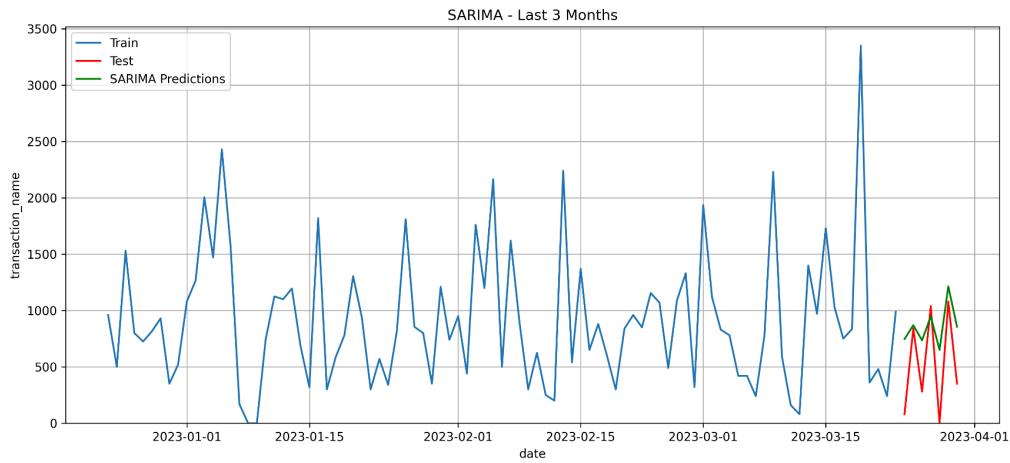
SHOPEE PARTNER APPLICATION. *Jurnal Ekonomi*, 12(01), 1196–1200.

<https://ejournal.seaninstitute.or.id/index.php/Ekonomi/article/view/1327>

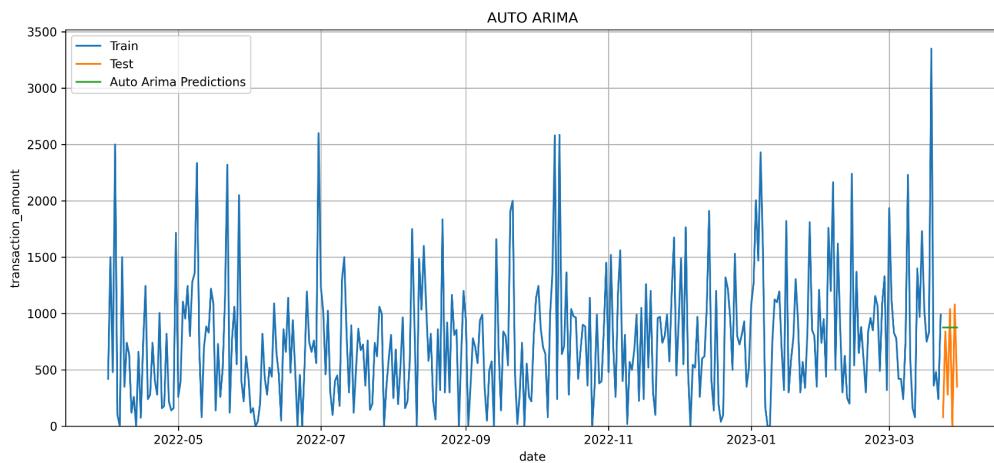
Zhu, R., Deng, Y., Sarkar, S., Kaoutar El Maghraoui, Ramasamy, H. V., & Bivens, A. (2016). Towards More Effective Solution Retrieval in IT Support Services Using Systems Log. *Lecture Notes in Computer Science*, 730–744. https://doi.org/10.1007/978-3-319-46295-0_52

Appendices

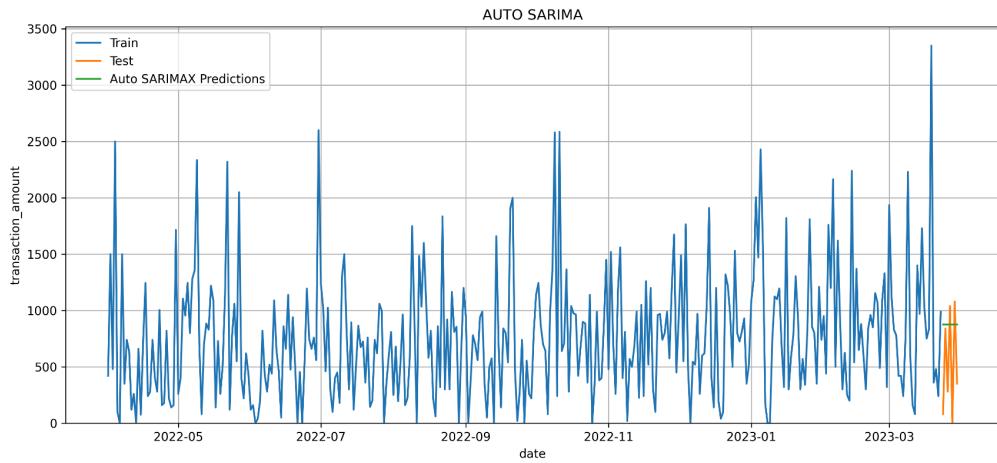
Appendix A: Time Series Models' Result and Source Code Link



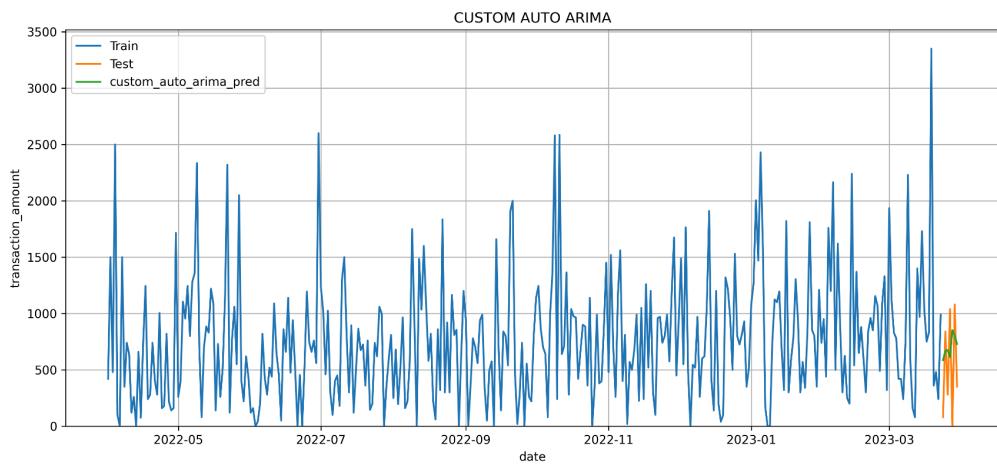
Appendix A.1. SARIMA Test. [Diagram].



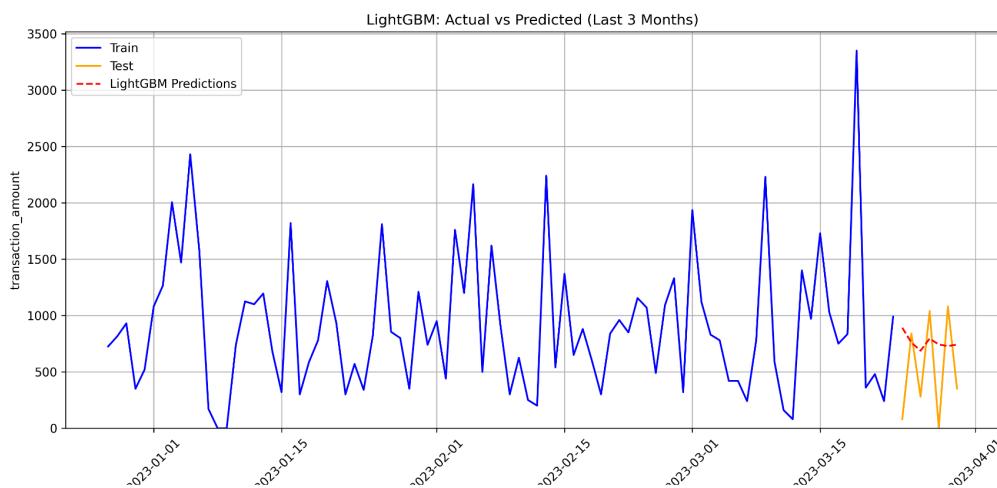
Appendix A.2. Auto ARIMA Test. [Diagram].



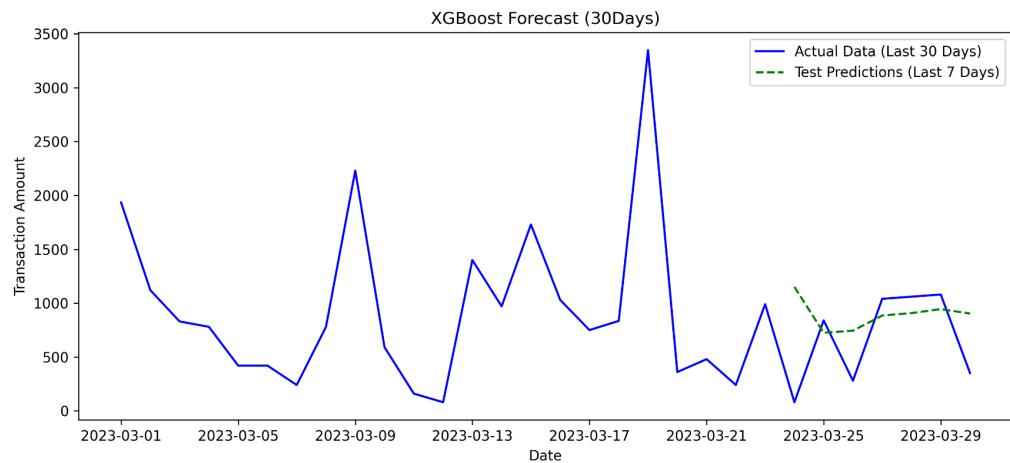
Appendix A.3. Auto SARIMA Test. [Diagram].



Appendix A.4. Custom Auto ARIMA Test. [Diagram].



Appendix A.5. LightGBM Test. [Diagram].



Appendix A.6. XGBoost. [Diagram].

Appendix B: Context Diagram

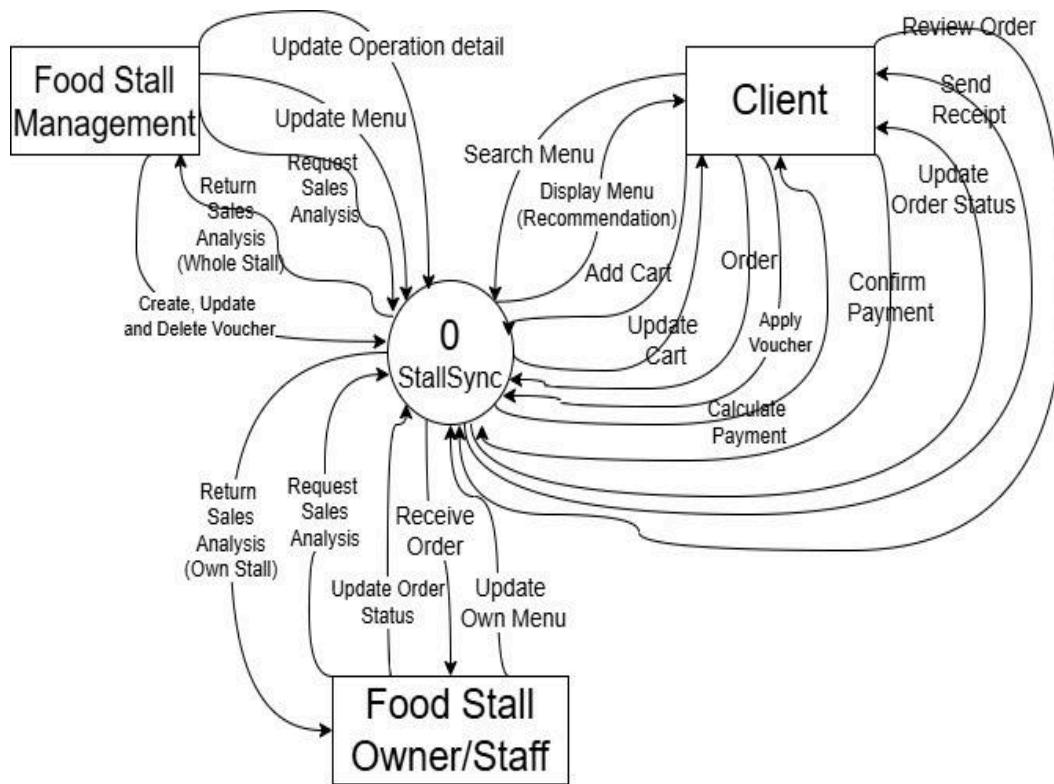


Figure B.1: Context Diagram for StallSync

The system interaction, consisting of the Admin and Client subsystems, is represented through a context diagram. This diagram provides a high-level, visual overview of the system and its interactions with external entities, clarifying the scope and defining the roles of each stakeholder.

Three distinct roles interact with the system. Firstly, the Food Stall Management Team holds a pivotal position, functioning as the system administrator. Their responsibilities extend beyond basic administration, encompassing the comprehensive management of merchant (Food Stall) data, product (menu) information, and user administration. This role is crucial for ensuring the system's integrity and functionality, as they possess the authority to add, update, and delete merchants, products, and other critical data. Furthermore, they leverage sales performance data to gain insights and make informed strategic decisions, directly impacting the system's efficiency and effectiveness.

Secondly, the Food Stall Owner/Staff focuses on the operational aspects of individual food stalls. The food stall owner is responsible for managing their stall's menu and handling incoming orders during operational hours. Food stall staff members play a vital role in serving submitted orders and ensuring the smooth and efficient operation of the food stall.

Lastly, the Client (Consumer) utilizes the system for convenient food ordering and transaction management. The system is designed to enhance the consumer's experience by providing personalized menu recommendations based on order history, top-selling items, and user-defined sort and filter options. This tailored approach aims

to streamline the ordering process and provide a more convenient and efficient service.

This context diagram serves as an abstraction, a foundational starting point for more detailed analysis and modeling activities.

Appendix C: Portfolio 1 Plagiarism Report

4/25/25, 10:02 PM

run_plagiarism

Originality report

COURSE NAME
FYP RSW & RDS 2025

STUDENT NAME
ZHE HENG YEOH

FILE NAME
run_plagiarism

REPORT CREATED
Apr 25, 2025

Summary

Flagged passages	29	2%
Cited/quoted passages	21	1%

Web matches

coursehero.com	5	0.7%
scribd.com	11	0.5%
chegg.com	8	0.3%
medium.com	3	0.2%
analyticsvidhya.com	1	0.2%
digipay.guru	1	0.1%
imaginovation.net	1	0.1%
fiveable.me	1	0.1%
jamezz.com	1	0.1%
studyx.ai	1	0.1%
geeksforgeeks.org	1	0.1%
oneidentity.com	1	0.1%
nvidia.com	1	0.1%
oregon-systems.com	1	0.1%
viiitorcloud.com	1	0.1%
futuremarketinsights.com	1	0.1%
research.com	1	0.1%
tandfonline.com	1	0.1%
walaw.press	1	0.1%
github.com	1	0.1%
nuvioo.com	1	0%
plumpos.com	1	0%
cudocompute.com	1	0%
webunlimited.com	1	0%
heinsohn.co	1	0%
linkedin.com	1	0%
aimlprogramming.com	1	0%

1 of 50 passages

<https://classroom.google.com/g/sr/NzMyOTY5NjkzOTAz/Nzc1NDcxNzkyMzAy/1D0too9m91MsAKTbfT9ICRVmtQuy-Een1jndkhEBeads>

1/12

Appendix C.1 Plagiarism Checking. [Diagram].

Appendix D: Github Project Repository

1. Forecast Algorithms (ARIMA, SARIMA, and more that are mentioned in the document) [URL.] <https://github.com/zhhhyzh/Time-series-algorithm>
2. Sample Code Standard in Backend [URL.]
<https://github.com/zhhhyzh/StallSync-standards>
3. StallSync Backend (Nodejs & Expressjs), Database Backup [URL.]
<https://github.com/zhhhyzh/StallSync-api>
4. StallSync Frontend for Administration (Reactjs) [URL.]
<https://github.com/zhhhyzh/stallsync-portal>
5. StallSync Testscript for data creation and generation (Python) [URL.]
<https://github.com/zhhhyzh/StallSync-testscript>
6. Recommendation Algorithm (Collaborative Recommendation, Python) [URL.]
<https://github.com/zhhhyzh/StallSync-Recommendation>
7. StallSync Frontend for Client (React Native)
[URL.]<https://github.com/WwwWKit/Stallsync-native>

Appendix E: StallSync Poster

The poster is a dark-themed promotional graphic for the StallSync project. It features a logo at the top left with a stylized 'S' inside a stall icon and the word 'STALLSYNC' below it. To the right is a wavy yellow line graphic. The poster is divided into two main sections: 'Project Objectives' on the left and 'Problem Statement' on the right, separated by a vertical line.

Project Objectives

- Launch personalized, AI-powered menu system (Month 3)
- Reduce customer anxiety and enhance patience
- Develop StallSync (Admin): Web-based management platform
- Develop StallSync (Client): Intelligent mobile/web food ordering
- Enable smart record keeping with real-time dashboards
- Use predictive analytics for sales, demand, and staffing forecasts

Project Scope

- Web-based management system for stall owners, staff, and administrators
- Mobile/web application for customers with personalized food recommendations and ordering

Problem Statement

- No data-driven insights for better decision-making
- Manual records cause errors and delays
- Outdated cash-only payment systems
- Overwhelming menu options create choice fatigue
- Menu anxiety and fear of choosing the wrong dish
- Cluttered menus reduce readability and satisfaction
- No real-time order status updates
- Long wait times frustrate customers
- Poor staff scheduling and performance tracking
- Difficult for canteen admins to manage multiple stalls
- Old systems cause IT inefficiencies

Visit our site to order food

Below the text are two mobile device screens. The left screen shows the 'StallSync Portal' login interface on a laptop, and the right screen shows the 'Intelligent Ordering System' login interface on a smartphone. Both screens feature a food stall icon with a burger, fries, and a drink. At the bottom of the poster, there is a footer section with the TARUMT logo, Node.js and React.js logos, and a blue bird icon.

Appendix E.1. Poster. [Diagram].

This page is intentionally left blank to indicate the back cover. Ensure that the back cover is black in color.