

MYSZTECH SOLUTION (MTS) POINT OF SALES (POS) SYSTEM

MOHAMAD MOHSIN BIN ISMAIL

Bachelor of Computer Science (Software
Engineering) with Honours

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MOHAMAD MOHSIN BIN ISMAIL

Date of Birth

Title : MYSZTECH SOLUTION (MTS) POINT OF SALES (POS) SYSTEM

Academic Session : 2022/2023

I declare that this thesis is classified as:

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

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

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

Certified by:

(Student's Signature)

New IC/Passport Number
Date: 14 July 2023

(Supervisor's Signature)

TS DR. AZLEE BIN ZABIDI
SENIOR LECTURER
FACULTY OF COMPUTER SYSTEMS
& SOFTWARE ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK, 26300 GAMBANG, KUANTAN
TEL: 06-546 2007 FAX: 06-546 2144

TS DR AZLEE BIN ZABIDI
Date: 14 July 2023

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

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

 (ii)

 (iii)

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the Bachelor's degree in Computer Science (Software Engineering)

(Supervisor's Signature)

Full Name : TS. DR. AZLEE ZABIDI

Position : Senior Lecturer

Date : 14 July 2023



STUDENT'S DECLARATION

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

(Student's Signature)

Full Name : MOHAMAD MOHSIN BIN ISMAIL

ID Number : CB20147

Date : 02 DECEMBER 2022

MYSZTECH SOLUTION (MTS) POINT OF SALES (POS) SYSTEM

MOHAMAD MOHSIN BIN ISMAIL

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy/Master of Science/Master of Engineering

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JUNE 2023

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to all those who have helped and supported me throughout the completion of my Final Year Project. The completion of this project would not have been possible without the guidance, support and encouragement of many individuals.

Firstly, I would like to extend my sincere thanks to my project supervisor, Ts Dr Azlee Zabidi, for providing valuable guidance, support and encouragement throughout the project. Your wealth of knowledge and experience, as well as your willingness to share your time and expertise, were truly invaluable. I truly appreciate the time and effort you have invested in me and my project, and your invaluable feedback and suggestions helped me to improve the quality of my work.

I would also like to thank Universiti Malaysia Pahang for providing me with the opportunity to undertake this project and for providing the necessary resources and facilities. The access to various resources such as journals, books and online databases helped me to conduct my research effectively and efficiently.

I am also grateful to my friends and classmates for their support and encouragement during the project. Your support and motivation helped me to stay focused and motivated throughout the project.

Lastly, I would like to thank my family for their love, support and encouragement throughout my academic journey. Their unwavering support and belief in me helped me to overcome any obstacles and challenges that I faced during the project.

I am truly grateful to everyone who has helped me to complete this project. Thank you all for making this project a success.

ABSTRAK

Sistem POS MYSZTECH dengan fungsi Susunatur Meja adalah satu penyelesaian komprehensif yang direka untuk meningkatkan kepuasan pelanggan dan mengoptimalkan operasi untuk pemilik perniagaan cafe dan restoran. Pembangunan sistem ini dilaksanakan menggunakan metodologi *Waterfall* dan direka khusus mengikut keperluan klien untuk memastikan ia memenuhi keperluan industri perkhidmatan makanan. Ciri utama sistem ini adalah fungsi susunatur meja yang inovatif, yang membolehkan pengesanan pesanan yang berkesan dan memudahkan pengalaman makan. Dengan pelbagai ciri pengurusan termasuk pengurusan jualan, pengurusan item, keserasian pencetak terma, pengurusan diskaun, pengurusan cukai, dan pengurusan cawangan, sistem ini memberikan membolehkan perniagaan mengurus operasi mereka dengan mudah. Dengan pembangunan sistem ini, perniagaan dapat mengoptimalkan operasi mereka dan membuat keputusan yang bijak untuk mencapai kejayaan dalam industri perkhidmatan makanan yang kompetitif.

ABSTRACT

The MYSZTECH POS System with Table Layout is a comprehensive solution designed to enhance the customer experience and optimize operations for cafe and restaurant owners. Developed using the waterfall methodology and tailored to specific client requirements, this system offers a range of management features, including sales management, item management, thermal printer compatibility, discount management, tax management, and outlet management. The key highlight of this solution is its innovative table layout functionality, which enables efficient order tracking and streamlines the dining experience. With real-time insights into customer behavior and seamless table management, businesses can optimize their operations and make informed decisions to drive success in the competitive food service industry.

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Introduction	1
1.2	Problem Statement	2
1.3	Proposed Solution	2
1.4	Aim and Objectives	3
1.5	Scope of Project	3
1.6	Significance	4
1.7	Thesis Organization.....	4
2	LITERATURE REVIEW	5
2.1	Introduction	5
2.2	Square.....	5
2.2.1	Features	6
2.2.2	Advantages and Disadvantages.....	6
2.3	Lightspeed	7
2.3.1	Features	7
2.3.2	Advantage and Disadvantage.....	8
2.4	Toast.....	8
2.4.1	Features	9
2.4.2	Advantage and Disadvantage.....	9
2.5	Summary	10
3	METHODOLOGY	11
3.1	Introduction	11
3.2	Methodology	11
3.2.1	Waterfall Model.....	12

3.2.2	Requirement Analysis	13
3.2.3	System Design	13
3.2.4	Implementation	14
3.2.5	System Testing.....	16
3.2.6	System Deployment	16
3.2.7	System Maintenance	16
3.3	Project Requirement.....	17
3.3.1	Functional Requirement.....	17
3.3.2	Non-Functional Requirement.....	17
3.3.3	Constraints	18
3.4	Propose Design.....	19
3.4.1	Flowchart	19
3.4.2	Context Diagram.....	23
3.4.3	Use Case Diagram.....	24
3.4.4	Use Case Description	25
3.4.5	Storyboard.....	46
3.4.6	User Interface.....	55
3.5	Data Design	72
3.6	Proof of Initial Concept.....	72
3.7	Validation and Testing Plan	73
3.7.1	User Acceptance Test	73
3.7.2	Usability Feedback Form.....	76
3.7.3	Table Layout Feedback Form	77
3.8	Potential Used of Proposed Solution.....	77
3.9	Gantt Chart.....	79

4	IMPLEMENTATION, RESULT AND DISCUSSION	80
4.1	Introduction	80
4.2	Implementation Process	80
4.2.1	Setup Version Control.....	81
4.2.2	Design Database Structure	82
4.2.3	Development of the API	85
4.2.4	API Development Coding.....	87
4.2.5	Mobile App Development Coding.....	96
4.2.6	Development of the MTS POS System	102
4.3	Testing and Result Discussion	120
5	CONCLUSION	121
5.1	Introduction	121
5.2	Discussion on User Acceptance	122
5.3	Limitation and Constraint.....	123
5.3.1	Cost	123
5.3.2	Timeline	123
5.3.3	Hardware Integration	124
5.4	Future Work	124
5.4.1	Generating Invoices for Clients	124
5.4.2	Credit Card Machines and Online Payment Gateways.....	124
5.4.3	Zoom In/ Out Capability for Table Layout.....	125
5.4.4	Enhance Reporting Module	125
6	REFERENCES	126

LIST OF TABLES

Table 2.1 Comparison of significance features.....	10
Table 3.1 List of the hardware used as the requirements.....	14
Table 3.2 List of the software used as the requirements.....	15
Table 3 Use Case Description for Login.....	25
Table 4 Use Case Description for Manage Sales.....	26
Table 5 Use Case Description for Manage Table Layout.....	30
Table 6 Use Case Description for Manage Item.....	34
Table 7 Use Case Description for Manage Receipts	36
Table 8 Use Case Description for Manage Outlets.....	38
Table 9 Use Case Description for Manage Category.....	40
Table 10 Use Case Description for Manage Category.....	42
Table 11 Use Case Description for Manage Tax	44
Table 3.12 Storyboard for Cashier.....	46
Table 3.13 Functionality Feedback Form	73
Table 14 Usability Feedback Form.....	76
Table 3.15 Table Layout Feedback Form.....	77

LIST OF FIGURES

Figure 1 Square logo.....	5
Figure 2 Square Floor Management	6
Figure 3 Lightspeed POS system.....	7
Figure 4 Managing Tables page.....	9
Figure 5 Waterfall Model Phases.....	13
Figure 6 Flowchart of the Ordering Process	19
Figure 7 Flowchart to View all Receipts	21
Figure 8 Flowchart to manage Items, Categories, Modifiers and Discounts.....	22
Figure 9 Context Diagram of the Application	23
Figure 10 Use Case Diagram	24
Figure 11 Use Case Login.....	25
Figure 12 Use Case Manage Sale	26
Figure 13 Use Case Manage Table Layout.....	30
Figure 14 Use Case Manage Items	34
Figure 15 Use Case Manage Receipts	36
Figure 16 Use Case Manage Outlets.....	38
Figure 17 Use Case Manage Discounts	40
Figure 18 Use Case Manage Category.....	42
Figure 19 Use Case Manage Tax	44
Figure 20 Login UI for Cashier	55
Figure 21 Main Sales UI.....	56
Figure 22 Table Layout UI	57
Figure 23 Item Details Popup UI.....	58
Figure 24 Popup when Press Save button UI.....	59
Figure 25 Order Details UI	60
Figure 26 Payment Details UI.....	61
Figure 27 Menu in the Side Navigation.....	62
Figure 28 List of Receipt UI.....	63
Figure 29 Item List UI	64
Figure 30 Add Item UI.....	65

Figure 31 List Category UI.....	66
Figure 32 Add Category UI	67
Figure 33 List Modifiers UI.....	68
Figure 34 Add Modifier UI.....	69
Figure 35 List Discounts UI.....	70
Figure 36 Add Discount UI.....	71
Figure 37 Gantt Chart	79
Figure 38 Gitlab Setup	81
Figure 39 List of migration files	82
Figure 40 Receipt Items table migration file snippet.....	83
Figure 41 MTS POS System database	84
Figure 42 Postman API Testing UI.....	85
Figure 43 Auth Controller–Verify User Email	87
Figure 44 Auth Controller-Login.....	88
Figure 45 Auth Controller-Logout.....	89
Figure 46 Auth Controller-Resend Verification Email.....	89
Figure 47 Auth Controller-Send Email Verification Code	90
Figure 48 Category Item Controller - Store	90
Figure 49 Category Item Controller - Update.....	91
Figure 50 Dining Setting Controller - Store	91
Figure 51 Dining Setting Controller - Update	92
Figure 52 Discount Item Controller - Store	92
Figure 53 Discount Item Controller - Update	93
Figure 54 Order Setting Controller- Store	93
Figure 55 Order Setting Controller - Update	94
Figure 56 Receipt Controller – Send Receipt	94
Figure 57 Sale Controller - Store	95
Figure 58 ItemBloc	96
Figure 59 LoginFormBloc	98
Figure 60 User Model	100
Figure 61 Login Form UI Code Snippet.....	101

Figure 62 Login UI for Cashier	102
Figure 63 Main Sales UI.....	103
Figure 64 Table Layout UI	104
Figure 65 Edit Table Layout UI.....	105
Figure 66 Item Details Popup UI.....	106
Figure 67 Popup when Press Save button UI.....	107
Figure 68 Order Details UI	108
Figure 69 Payment Details UI.....	109
Figure 70 Menu in the Side Navigation	110
Figure 71 List of Receipt UI.....	111
Figure 72 Item List UI	112
Figure 73 Add Item UI.....	113
Figure 74 List Category UI.....	114
Figure 75 Add Category UI	115
Figure 76 List Modifiers UI.....	116
Figure 77 Add Modifier UI.....	117
Figure 78 List Discounts UI.....	118
Figure 79 Add Discount UI.....	119

LIST OF ABBREVIATIONS

MTS

MYSTECH SOLUTION

POS

Point of Sales

UAT

User Acceptance Test

UI

User Interface

CHAPTER 1

1 INTRODUCTION

1.1 Introduction

The “information” or “digital” age, which started in the 1970s, is the name given to the contemporary technological era (Forsyth, 1999). Both large and small businesses have changed how they operate to improve output and grow their customer base. If a business is unable to keep up with technological developments, the relationship between the client and the end user is not adequately efficient. For instance, restaurant owners might go from traditional ways like paperwork to a more manageable system that can generate reports and provide real-time insight about their business without requiring them to spend more time and money just for auditing and analysis purposes.

A commonly used system among restaurant owners is Point-Of-Sale(POS) system. A POS system consists of hardware, software, and payment processing services used by businesses to receive cash or digital payments from clients. Today’s POS systems may also create reports, aid inventory management, and allow staff to clock in and out. (Lisa Anthony, 2022). When considering the Malaysian economy, it’s important to note that SMEs are the primary customers of the POS system.

As these works play a vital role in businesses, MYSZTECH Solution Enterprise, in its role as the POS System installation provider and the customer for this project, has identified the flaws present in most of the restaurant businesses even using the POS system. By identifying the flaws and limitations of existing POS systems, MYSZTECH Solution Enterprise aims to address the gaps and improve upon the current state of POS technology in the restaurant sector. Through this project, MYSZTECH Solution Enterprise strives to develop a robust and user-friendly POS system that not only meets the needs of Malaysian SMEs but also provides advanced features to enhance operational efficiency, improve customer service, and boost overall business performance.

1.2 Problem Statement

Nowadays, restaurants serve as important social hubs in almost every country and culture. In addition to providing food, restaurants also serve as a location for people to mingle, connect, and share memorable memories. Restaurants are also highly lucrative businesses, as the global restaurant industry market size was valued at USD 2,323.29 billion in 2021, and it is forecasted to grow from \$2,540.05 billion in 2022 to \$5,194.60 billion by 2029 (“Foot Orthotic Insoles Market Size, Share | Growth Analysis, 2029,” 2022). Nevertheless, despite restaurants’ vital position in our society and huge revenues, the restaurant business has undergone minimal changes in terms of technology for the last century.

One significant challenge in the restaurant industry is the lack of an efficient table layout management system within existing POS solutions. Without a comprehensive table layout functionality, restaurant owners and staff face difficulties in effectively managing table assignments, optimizing seating capacity, and providing seamless customer service. This results in operational inefficiencies, longer waiting times, potential overcrowding, and a subpar dining experience.

To tackle these challenges, it is imperative to develop an innovative POS system tailored explicitly for the F&B industry. Such a system should provide one standout feature which is intuitive table layout functionality. By undertaking this endeavor, MYSZTECH Solution Enterprise aims to revolutionize restaurant operations, elevate customer experiences, and foster business growth within the dynamic and competitive F&B industry.

1.3 Proposed Solution

To address the challenges faced by restaurant owners and operators in the F&B industry, the proposed solution is to develop a comprehensive POS system with a focus on introducing robust table layout functionality. This innovative feature aims to streamline and enhance the efficiency of order management and customer service within restaurants.

The table layout functionality allows restaurant owners and operators to design and customize their seating arrangements according to their specific needs. With the intuitive layout editor, tables, chairs, and other elements can be freely arranged, enabling optimal space utilization

and personalized dining experiences. This flexibility ensures that the POS system adapts seamlessly to the unique layout of each establishment.

In addition to the table layout functionality, the proposed POS system will include essential features commonly found in general POS systems. It will provide comprehensive capabilities for managing sales, items, and receipts. The restaurant staff will be able to process transactions, track sales, apply discounts, and manage payment methods, both cash and digital. The system will also offer seamless integration with external receipt printers.

By combining the table layout functionality with comprehensive POS features, MYSZTECH Solution Enterprise aims to provide restaurant owners with a holistic solution that optimizes operations, improves order management, and enhances customer service. The proposed POS system will empower F&B establishments to efficiently manage sales, items, and receipts while creating a seamless and personalized dining experience through its flexible table layout functionality.

1.4 Aim and Objectives

This project aims to develop a basic mobile application for the POS system and a web application for the self-ordering module.

Objectives:

- i. To study the requirements for POS system.
- ii. To develop a mobile application with basic POS system functionality including the table layout.
- iii. To test the proposed system.

1.5 Scope of Project

The scope of this study is:

- i. POS system runs on tablets 10inch that uses android OS only.
- ii. Implementation of the system at restaurants in Selangor.
- iii. Integrated with cash drawer, and Bluetooth thermal printer.

1.6 Significance

The significances of this study are:

- i. To enhance cashier's workflow and improve transaction efficiency.
- ii. To ensure accurate order processing and minimize errors for improved customer satisfaction.
- iii. To create a seamless and personalized dining experience for customers through the table layout functionality.

1.7 Thesis Organization

This thesis consists of three chapters which are Introduction, Literature Review, and Methodology. Chapter 1 discusses the introduction of the project including the current issues, problems, objective and scope. Chapter 2 contains information about the literature review for three existing systems along with the comparison of the advantages and disadvantages of the system. Chapter 3 discusses the project's methodology that is used to develop this system. This includes the project requirement, propose design, proof of initial concept and testing.

CHAPTER 2

2 LITERATURE REVIEW

2.1 Introduction

This chapter will discuss the literature overview of the existing system that can help to understand more about the requirements of the system by comparing it with commonly used systems at restaurants and food and beverage (F&B) businesses which are Square, Lightspeed and Toast.

2.2 Square



Figure 1 Square logo

Square POS is a POS system that was launched in 2009 by Jack Dorsey and Jim McKelvey. It is a mobile point-of-sale (POS) system that allows businesses to accept payment, manage inventory and track sales. Square is a popular choice for small businesses and entrepreneurs. It is easy to use, affordable and packed with features. (“POS (Point of Sale) Software for your business | Square,” 2023)

2.2.1 Features

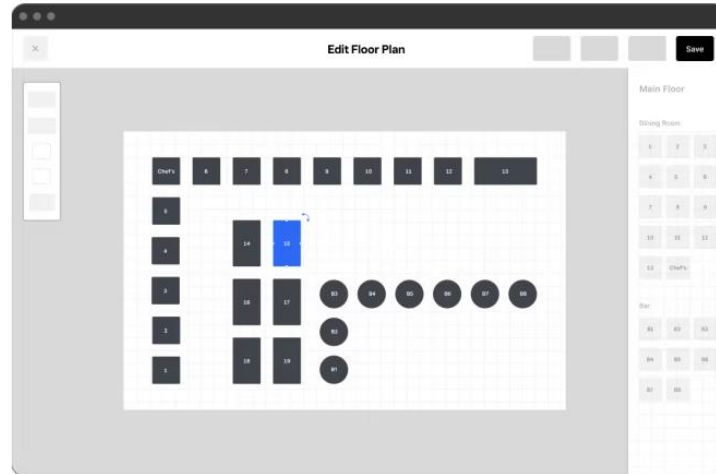


Figure 2 Square Floor Management

Square Floor Management helps the user to customize floor plans. They can create a floor plan that accurately reflects the layout of the restaurant, including the number and type of tables. It also can give the user its status. The floor plan will show the status of each table in restaurants, such as 'open', 'seated', 'waiting' or 'closed'. This information can help the user to keep track of tables and ensure that customers are seated promptly. Other than that, Square POS also includes a variety of tools that help the user to manage their table, send alerts to staff or print table tickets.

2.2.2 Advantages and Disadvantages

The advantages of Square POS floor management are increased customer satisfaction and reduced wait times. The customer will feel appreciated for being seated promptly and knowing the status of their table. These features also can help to reduce wait times for customers, which can lead to increased sales. However, disadvantages for this floor management are no offline mode since it requires an internet connection to function. This can be a problem for businesses that have spotty internet services or that need to be able to process payments offline.

2.3 Lightspeed

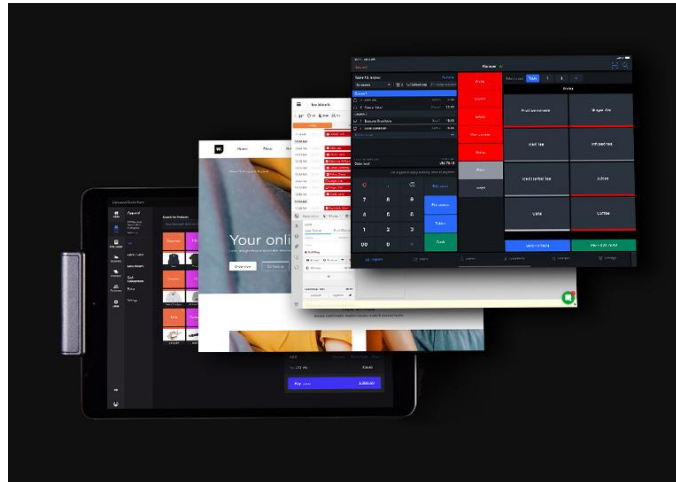


Figure 3 Lightspeed POS system

The second system is Lightspeed. Lightspeed was launched in 2009 and it is a cloud-based point-of-sale (POS) system for businesses of all sizes. It offers a variety of features, including table layout, inventory management, customer management and reporting. Lightspeed is used by over 100,000 businesses in over 10 countries, and it headquarters in Toronto, Canada. Lightspeed has the features of inventory management, customer management, table layout and reporting. (“Bakery & Coffee Shop POS System | Lightspeed Restaurant Point of Sale,” 2023)

2.3.1 Features

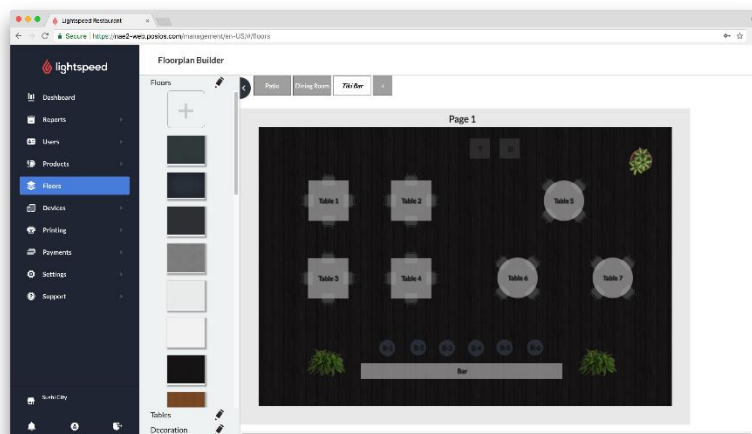


Table layout features in Lightspeed allow businesses to create a digital floor plan of their restaurant or store. This floor plan can then be used to track table availability, assign tables to customers, and send orders to the kitchen. When tables are properly laid out, it can help the staff to more easily take orders, deliver food and clear tables. This can lead to faster service and happier customers. If the restaurant's layout changes, staff can easily update the table layout in Lightspeed POS. This can be helpful if they want to move or add tables or host special events.

2.3.2 Advantage and Disadvantage

The advantages of Table Layout in Lightspeed are reduced wait times and improved efficiency. Table layout can help businesses by reducing the waiting times by allowing customers to see which tables are available and choose their own table. It also can help to improve efficiency by automating tasks such as assigning tables to customers and track table availability. However, the disadvantage is it required additional fees since this feature is an add-on to the Lightspeed POS system.

2.4 Toast



The third system is the Toast system. Toast POS system was launched in 2013. It is a cloud-based point-of-sale (POS) system that helps restaurants manage their operations. Toast POS system offers a variety of features, including order and inventory management, customer relationship management (CRM), gift card management, mobile ordering and more. Toast POS is used by over 100,000 restaurants in the United States and Canada. It is a popular choice for restaurants of all sizes, from small cafes to large chains. (Toast, 2019)

2.4.1 Features

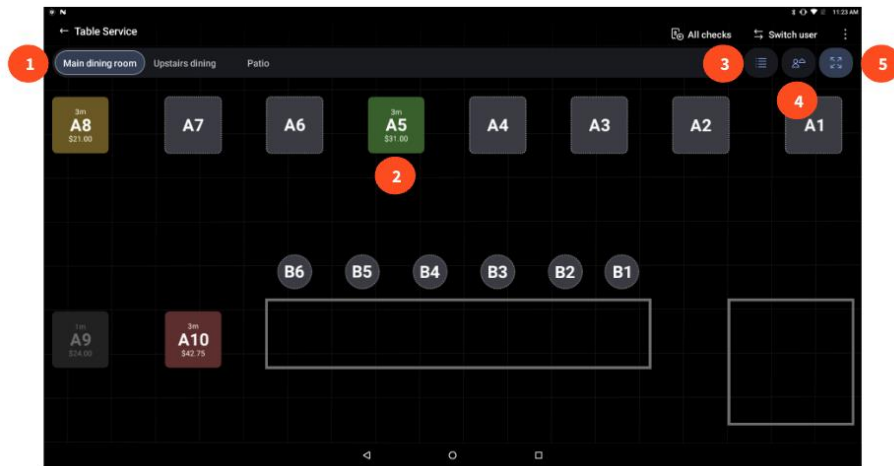


Figure 4 Managing Tables page

Toast POS system also includes a table management feature that allows staff to easily manage the availability of tables, track the status of each table and assign tables to the customer as they arrive. This feature provides a real-time view of table availability to help staff see which tables available and which tables are occupied. It also can help staff track the status of each table to help them notify which tables are dirty and which are ready for service. If the customer wants to hold a party, Toast POS table management allows staff to easily assign tables for customers based on the size of the party and their references.

2.4.2 Advantage and Disadvantage

The advantages of using Toast POS table management system are increased sales and improved customer service. Toast POS table management can help restaurants increase sales by providing them with insights into their customers and their buying habits. Toast POS can also help restaurants run promotions and targeted marketing campaigns. However, the disadvantage of this feature is it is not very flexible. It is designed for a traditional dining room layout, with tables that are assigned to serves. This can be limiting to restaurants with a more casual atmosphere or to different seating arrangements, such as high tops or bar seating.

Table 2.1 Comparison of significance features

Features	Application			
	Square	Lightspeed	Toast	Propose Application
Able to change table name	/	/	/	/
Able to set Predefine Order to the Tables	/			/
Disable Table function	/	/	/	/
Freely place the table on the floor plan	/	/		/
Accurately represent real floor plan		/		/

2.5 Summary

Based on the comparison of existing system above, we can conclude the most important features for the table layout is accurately represent real floor plan. This is important because it allows the user to optimize table placement and improve staff efficiency. Represent real floor plan here means, the user can see the position of the cashier, multiple type of tables and the kitchen position. When user can see the actual layout of restaurants, it will help for better understanding on maximize efficiency and space. Disable table function also important as user able to disable any table that closed or booked. Other than that, not all the existing system has the function of predefined order in their system. This function will allow user to connect the table with their predefined order. Hence, the findings that has been discovered will be included in the development of the project.

CHAPTER 3

3 METHODOLOGY

3.1 Introduction

This chapter will discuss the details of methodology for MTS POS System that is used to develop the project which is the waterfall model. This framework has been chosen because the project timeframe is short, and the requirements are not changing frequently.

3.2 Methodology

There are a few stages to be determined in order to find the results. This includes further details of the activity in the waterfall model and user requirements. Requirements from stakeholders are gathered at the start of the project, then a sequential project plan is made to take those requirements into account. The proposed design includes Flowchart of the system, context diagram, use case diagram, Entity Relationship Diagram, interface design, storyboard, database design, planning for implementation & testing and Gantt Chart. Every subtopic in this chapter, which is under waterfall model, explained how the steps are conducted.

3.2.1 Waterfall Model

Waterfall model consists of a series of six main phases to create and build a fully functioning system. For each phase of development, it establishes clear endpoints or objectives. Once certain endpoints or objectives have been achieved, they cannot be revisited. This cycle is also used by the designer/developer as a reference to avoid any mistakes occurring that will drag the timeline while developing a successful system.

There are six phases in the waterfall model, which are:

- i) **Requirement Analysis** - During this stage, all potential system requirements are gathered and outlined in a requirement specification document.
- ii) **System Design** - In this phase, the requirement specifications from the first phase are examined, and the system design is created. This system design aids in determining the overall system architecture as well as the hardware and system requirements.
- iii) **Implementation** – The models, logic, and requirements specification specified in the earlier phases are used to develop the source code. The system is typically coded in smaller components or units before being integrated.
- iv) **System Testing** – At this phase, problems that need to be fixed are found through quality assurance, unit, system, and beta tests. This can cause the debugging part of the coding process to be repeated. The waterfall process continues if the system passes integration and testing.
- v) **System Deployment** – The product or application is released into a live environment after being determined to be fully functional.
- vi) **System Maintenance** – To continuously improve, update, and enhance the product and its functionality, corrective, adaptive, and perfective maintenance is carried out. Release of patch updates and new versions may fall under this category.

Based on the detailed phases of the waterfall model, Figure 5 Waterfall Model Phases shows the phases of the cycle (Sinha & Das, 2021).

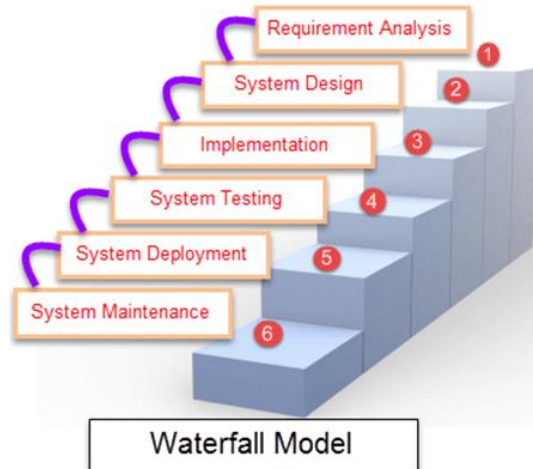


Figure 5 Waterfall Model Phases

3.2.2 Requirement Analysis

In a Waterfall Model the requirement analysis phase is the phase to gather the requirements and do a feasibility study. This includes crucial identification details like the deadlines, guidelines, planning, and requirements. All those information needs to be documented properly. It is also important to take into account any limitations and constraints like duration to complete the system and budget constraints which may impact the development process. Besides, during this phase also, I need the answer for the questions that is not covered in the requirements document such as do we have to support multiple language, and how many users expected to use the application. (“What is Waterfall model- Examples, advantages, disadvantages & when to use it?,” 2022)

3.2.3 System Design

In this phase, software developers create a technical solution to the problems outline in the previous stage, which may include scenarios, layouts and data models. This system design aids in determining the overall system architecture as well as the hardware and system requirements. Then, it is converted into a physical design using specialized hardware and software. For example, for this project the hardware that can be identified is the tablet 10 inch, thermal printer and cash drawer.

3.2.4 Implementation

Implementation phase is where we will use the artifacts that have been prepared in previous phase and ensure the solution follows the finalized design to do the coding. Besides, there are also activities in this phase which are the development of IT infrastructure and the development of the database (Imam, 2018). This also includes flow charts for the process of this study to ensure the system processes are arranged neatly.

To develop a functioning mobile application, the hardware and software required for this project need to be identified. The hardware that had been used in this research is a laptop, Wi-Fi, printer and external hard drive. A laptop is the most crucial tool needed as all the questionnaires, journals, papers and reports were stored on a laptop. Moreover, the software needed to run for the development of this application will use this laptop. Wi-Fi is also the most important tool, as nowadays all the references and data are in the cloud. Next is a printer that is needed to print all the required files and this proposal. The external hard drive is used to store all the important documents and as the place to store the project back up. The specification of hardware required in developing this project is shown in Table 3.1 List of the hardware used as the requirements

Table 3.1 List of the hardware used as the requirements

HARDWARE	PURPOSE
Laptop Asus TUF Intel® Core™ i5-8200H CPU @ 2.30GHz, 20.0GB DDR4 RAM	<ul style="list-style-type: none">• To store all literature review• To develop the system
Wi-fi / LAN Adapter	<ul style="list-style-type: none">• For an internet access
Printer	<ul style="list-style-type: none">• To print out this project's documents
External hard drive	<ul style="list-style-type: none">• Back-up any necessary and important files

The software used in this research is Microsoft Word office 365 ProPlus as the software to prepare the documentation of this thesis. For the reference of this documentation, Mendeley Desktop is used. Then for the preparation of the slide presentation, Microsoft PowerPoint is used. Microsoft Excel was used to summarize the questionnaire in Google Form and to create the Gantt Chart. All the diagrams were created by using Microsoft Visio Professional 2019. The specification of software required in developing the prototype is shown in Table 3.2 List of the software used as the requirements respectively.

Table 3.2 List of the software used as the requirements

SOFTWARE	PURPOSE
Microsoft Word office 365 ProPlus	<ul style="list-style-type: none"> • To prepare the documentation of this thesis
Mendeley Desktop	<ul style="list-style-type: none"> • Help in preparing the references needed for this project • To create the citation • To insert the bibliography at the references page
Microsoft PowerPoint office 365 ProPlus	<ul style="list-style-type: none"> • To prepare the presentation slide
Microsoft Excel office 365 ProPlus	<ul style="list-style-type: none"> • To summarize the questionnaire to see the result
Microsoft Visio Professional 2019	<ul style="list-style-type: none"> • To create the diagram such as, context diagram, ERD, DFD and Flowchart
Google Chrome	<ul style="list-style-type: none"> • To find information online
Visual Studio Code	<ul style="list-style-type: none"> • To develop the front-end and back-end of the system
MySQL	<ul style="list-style-type: none"> • For the online database
Photoshop CC 2019	<ul style="list-style-type: none"> • To design the graphic
Figma	<ul style="list-style-type: none"> • To make a sketch and mockup

3.2.5 System Testing

The system needs to be tested to ensure there are no errors and all requirements have been met before it is released to the client. In this phase, we need to create the test cases using design documents, personas, and user case scenarios. The main objective of this phase is to increase the quality of the product and to ensure client's positive experience when using the system. It is crucial for the testing team to know the system flow and identify areas that need to be improved.

3.2.6 System Deployment

The deployment stage includes implementing the system in the real environment after it has been tested in the previous phase. For this system, it will be deploy to SSD VPS server for the best performance.

3.2.7 System Maintenance

System maintenance includes making changes to the system or specific area in order to improve performance. These changes are initiated by the client or defects discovered during live system user. The developed system is regularly maintained and supported by the software developer.

3.3 Project Requirement

3.3.1 Functional Requirement

1. The cashier should be able to key in the sales.
2. The cashier should be able to manage the menus.
3. The cashier should be able to print the receipt.
4. The cashier should be able to manage the outlets.
5. The cashier should be able to manage the discounts.
6. The cashier should be able to manage the taxes.
7. The cashier should be able to manage table layout
8. The cashier should be able to manage the sales layout.

3.3.2 Non-Functional Requirement

1. **Usability:** The system should have a user-friendly interface that is intuitive and easy to navigate. It should provide clear instructions, error messages, and feedback to guide users through various operations. Additionally, the system should support customization and personalization options to cater to the specific needs and preferences of individual users.
2. **Security:** The system should have robust security measures in place to protect sensitive data, including customer information, payment details, and transaction records. It should employ encryption protocols, access controls, and secure authentication mechanisms to prevent unauthorized access, data breaches, and fraud.
3. **Reliability:** The system should be reliable and available for use during business hours. It should have built-in mechanisms to handle failures, recover from errors, and maintain data integrity. Additionally, it should be able to handle unexpected events, such as power outages or network disruptions, and gracefully resume normal operation.

3.3.3 Constraints

1. **Cost:** As the development of the system using Laravel Framework to develop the API, it cannot be hosting under shared hosting because the limitation to access the terminal. Hence, I need to subscribe to a VPS SSD server that will increase the cost of the development. Besides, I also need to find the correct hardware to integrate with.
2. **Timeline:** As this project will need to be developed using Laravel and Flutter, it will require me a lot of time to learn both framework and develop the system. Hence the system performance may not achieve the target performance.
3. **Hardware Integration:** Finding and integrating appropriate, compatible, and cost-effective hardware is a crucial requirement for the POS system, as it needs to seamlessly integrate with thermal printers and cash drawers.

3.4 Propose Design

3.4.1 Flowchart

3.4.1.1 Flowchart for Cashier

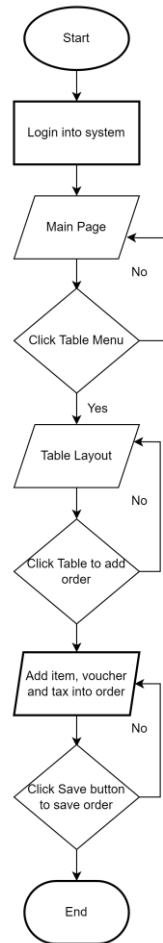
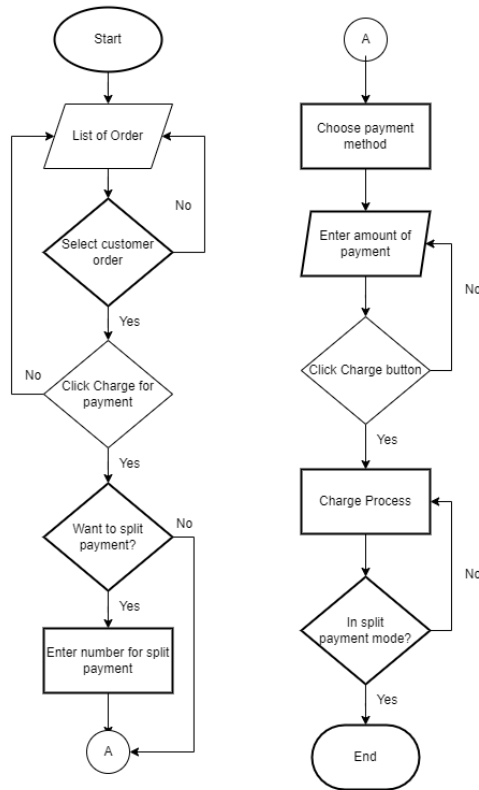


Figure 6 Flowchart of the Ordering Process

////



Flowchart of the Payment Process

Figure 6 Flowchart of the Ordering Pro shows the flow of mobile application that will be used by the cashier to manage the sale. Firstly, the cashier needs to login into the system using their registered credentials by the admin. After login, the system will display the main page. Whenever customers arrive at the restaurant, the cashier can view the availability of seat by clicking table icon from main page. It will display the Table Layout and if cashier wants to add order, they can press Table to add order. Besides that, they can add other items, vouchers, and tax with the order into the bill and press save button to save the order. When customers want to make payments, the cashier will press the open order button and it will display a list of orders. The cashier will choose customer order and press charge for the payment. If a customer wants to split their payment, the cashier will only need to enter the number for split payments and then choose the payment method for each bill. After that, press the charge button once the cashier has already entered the amount for the payment. If customers requested for e-receipt, the cashier needs to enter customer email and press send button. If not, the cashier only needs to print the receipt for the customer.

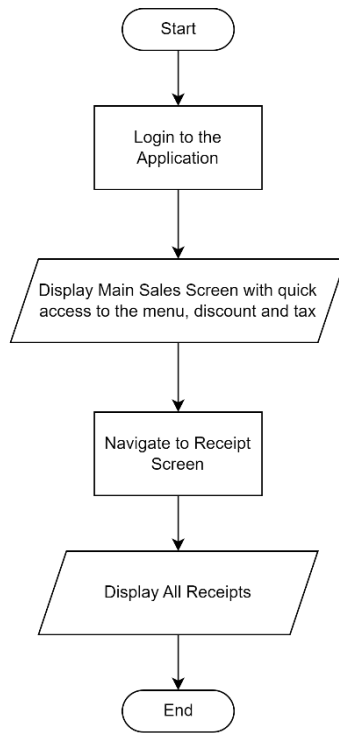


Figure 7 Flowchart to View all Receipts

Figure 7 Flowchart to View all Receipts shows the flow when the cashier needs to view all the receipts. The cashier needs to login to the system and choose *Receipt* from the side navigation. It will display the list of receipts. Cashiers can view each receipt details by clicking the receipts.

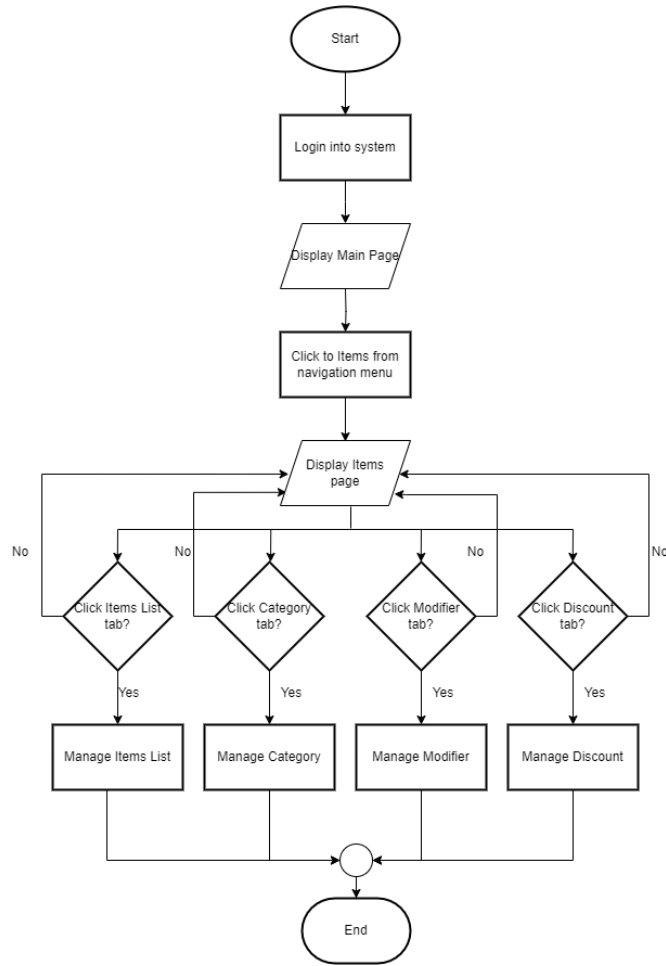


Figure 8 Flowchart to manage Items, Categories, Modifiers and Discounts

Figure 8 Flowchart to manage Items, Categories, Modifiers and Discounts shows the flow when the cashier wants to manage items, categories, modifiers and discounts. The cashier just needs to login to the system then choose the *Items* from the side navigation. Then the cashier can choose whether they want to manage items, categories, modifiers or discounts.

3.4.2 Context Diagram

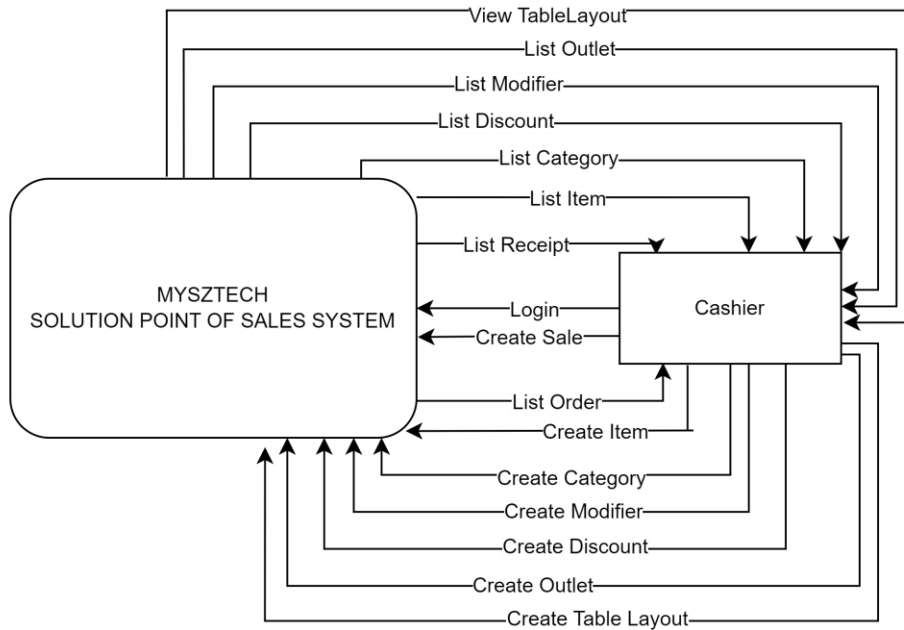


Figure 9 Context Diagram of the Application

The MTS POS system is a point-of-sale system that the cashiers can login to the system, manage table layouts, create a sale, view a list of orders, create new items, categories, discounts, outlets and modifiers. This enables cashiers to manage the sales, settings, and financial aspects of the restaurant. The system also allows cashiers to view the list of orders and receipts, making it easy for them to process payments and reconcile sales. The ability to create new items, categories, discounts and modifiers allows cashiers to manage the menu and offers in real-time.

3.4.3 Use Case Diagram



Figure 10 Use Case Diagram

Figure 10 Use Case Diagram shows the use case diagram for the MTS POS System with the cashier role comprises the following modules: Login, Manage Sale, Manage Table Layout, Manage Items, Manage Receipt, Manage Outlets, Manage Discount, Manage Category, and Manage Tax. These modules enable the cashier to perform various tasks within the system, starting with securely logging in, processing sales transactions, managing table layouts for table-based services, generating and managing receipts, manage outlets, applying discounts, organizing items into categories, and managing tax-related operations. These modules collectively empower the cashier to efficiently navigate and manage sales, inventory, table layouts, receipts, outlets, discounts, categories, and taxes within the MTS POS System.

3.4.4 Use Case Description

3.4.4.1 Login Use Case Description

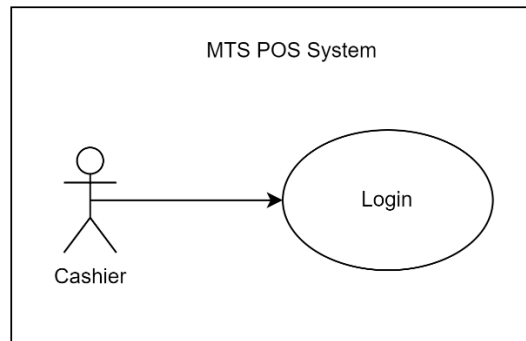


Figure 11 Use Case Login

Table 3 Use Case Description for Login

Use Case ID	UC100-MTS
Brief Description	This use case is used by cashier to login to the system.
Actor	Cashier
Pre-Conditions	Not applicable.
Basic Flow	<ol style="list-style-type: none"> 1. The system validates the cashier's password and logs him/her into the system. [E1: Invalid Credentials] 2. The system displays the Sales Module and the use case ends.
Alternative Flow	Not applicable.
Exception Flow	<p>E1: Invalid Credentials [UC101_MTS]</p> <p>If in the basic flow the system cannot find the email or the password is invalid, an error message is displayed. The actor can type in a new email or password or choose to cancel the operation, at which point the use case ends.</p>
Post-Conditions	User logged to the system.

3.4.4.2 Manage Sale Use Case Description

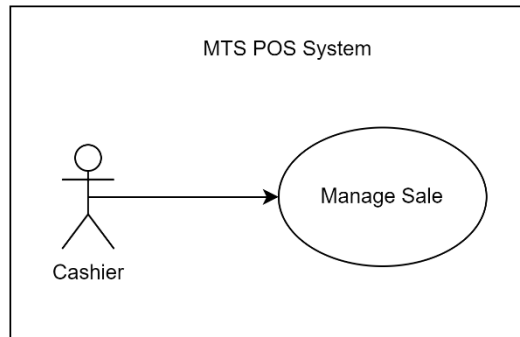


Figure 12 Use Case Manage Sale

Table 4 Use Case Description for Manage Sales

Use Case ID	UC200-MTS
Brief Description	This use case is used by cashier to manage the sales.
Actor	Cashier
Pre-Conditions	1. User already logged in to the system.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier login to the system or press Sales menu. 2. The system display the list of items and the order details. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a. Add new item to the order [A1: Add item to Order] b. Remove item from order [A2: Remove item from order] c. Edit item details in the order [A3: Edit item in order] d. Add discounts to the order [A4: Add discount to order]

	<ul style="list-style-type: none"> e. Remove discount from order [A5: Remove discount from order] f. Add taxes to the order [A6: Add taxes to order] g. Remove tax from order [A7: Remove tax from order] h. Save order [A8: Save order] i. Open existing order [A9: Open order] j. Charge customer [A10: Charge customer] k. Edit items arrangement [A11: Edit items arrangement] <p>4. The use case ends.</p>
Alternative Flow	<p>A1: Add item to order [UC201_MTS]</p> <ul style="list-style-type: none"> 1. The Cashier press any of the item shows at the items layout. 2. The system display popup of item details form. 3. The cashier enter the order details which are the quantity order, the variants, the add-on, the discount and the taxes. 4. The Cashier press <<Save>> button. 5. The system save the order details and show to the cashier at the right side of the screen. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Remove item from order [UC202_MTS]</p> <ul style="list-style-type: none"> 1. The Cashier slide any of the item shows in the right side layout. 2. The system remove the item. 3. The use case goes back to step 3 in basic flow (Cashier). <p>A3: Edit item in order [UC203_MTS]</p> <ul style="list-style-type: none"> 1. The Cashier press any of the item in the right side layout. 2. The system display popup with the edit form details.

3. The cashier update the order details which are the quantity order, the variants, the add-on, the discount and the taxes.
4. The Cashier press <<Save>> button.
5. The system save the order details and show to the cashier at the right side of the screen.
6. The use case goes back to step 3 in basic flow (Cashier).

A4: Add discount to order [UC204_MTS]

1. The Cashier press any of the discount shows at the items layout.
2. The system added the discounts to the order an automatically recalculate the price for the open order.
3. The use case goes back to step 3 in basic flow (Cashier).

A5: Remove discount from order [UC205_MTS]

1. The Cashier press the <<Discounts>> button in the right side layout.
2. The system display the list of discounts applied to the order
3. The cashier press trash icon.
4. The system unapplied the discount from the open order.
5. The use case goes back to step 3 in basic flow (Cashier).

A6: Add tax to order [UC206_MTS]

1. The Cashier press any of the tax shows at the items layout.
2. The system added the tax to the order an automatically recalculate the price for the open order.
3. The use case goes back to step 3 in basic flow (Cashier).

A7: Remove tax from order [UC207_MTS]

1. The Cashier press the <<Taxes>> button in the right side layout.
2. The system display the list of taxes applied to the order

3. The cashier press trash icon.
4. The system unapplied the tax from the open order.
5. The use case goes back to step 3 in basic flow (Cashier).

A8: Save order [UC208_MTS]

1. The Cashier already open order at the right side of the screen.
2. The cashier press the <<Save Order>> button.
3. The system display popup to select the predefined order.
4. The cashier press the <<Save>> button.
5. The system save the order details clear the order at the right side of the screen.
6. The use case goes back to step 3 in basic flow (Cashier).

A9: Open order [UC209_MTS]

1. The Cashier press the <<Open Order>> button at the right side layout.
2. The system display the list of save order.
3. The cashier press the required order to open.
4. The system display the selected order at the side layout.
5. The use case goes back to step 3 in basic flow (Cashier).

A10: Charge customer [UC210_MTS]

1. The cashier already open order.
2. The cashier press the <<Charge> button.
3. The system display the summary of order.
4. The cashier press <<Charge Order>> button
5. The system generate a receipt and remove from list of orders.
6. The use case goes back to step 3 in basic flow (Cashier).

A11: Edit items arrangement [UC211_MTS]

1. The cashier long press the items layout.

	<ol style="list-style-type: none"> 2. The system enter items edit mode. 3. The cashier arrange the items. 4. The cashier add or remove existing item on the layout. 5. The cashier press the <<Save> button. 6. The system save the arrangement of the items on the layout. 7. The system generate a receipt and remove from list of orders. <p>The use case goes back to step 3 in basic flow (Cashier).</p>
Exception Flow	Not applicable.
Post-Conditions	<ol style="list-style-type: none"> 1. Sales created or updated. 2. Items layout updated.

3.4.4.3 Manage Table Layout Use Case Description

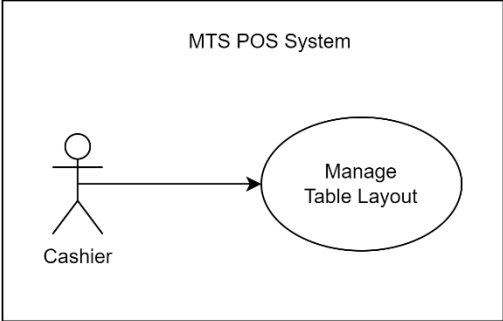


Figure 13 Use Case Manage Table Layout

Table 5 Use Case Description for Manage Table Layout

Use Case ID	UC300-MTS
Brief Description	This use case is used by cashier to manage table layout in the system.
Actor	Cashier

Pre-Conditions	5. User already logged in to the system.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press the table layout at the top of the sales main page. 2. The system display the table layout. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a) Add new section [A1: Add section] b) Edit section [A2: Edit section] c) Delete section [A3: Delete section] d) Assign order to the table [A4: Assign order] e) Open order from the table [A5: Open order] f) Edit table layout [A6: Edit table layout] g) Delete table [A7: Delete table] h) Edit table [A8: Edit table] 4. The use case ends.
Alternative Flow	<p>A1: Add section [UC301_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display list of section and the table layout. 3. The Cashier press <<Add Section>> button. 4. The Cashier enter the section name. 5. The Cashier press <<Save> button. 6. The system save new section. 7. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit section [UC302_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display list of section and the table layout. 3. The Cashier long press the existing section name. 4. The Cashier update the section name. 5. The Cashier press <<Save> button.

6. The system update the section.
7. The use case goes back to step 3 in basic flow (Cashier).

A3: Delete section [UC303_MTS]

1. The Cashier press <<Edit>> button.
2. The system display list of section and the table layout.
3. The Cashier long press the existing section name.
4. The Cashier press the <<Delete>> button.
5. The system delete the section.
6. The use case goes back to step 3 in basic flow (Cashier).

A4: Assign order [UC304_MTS]

1. The Cashier press on the required table.
2. The system redirect the cashier to the main sales page.
3. The Cashier input the order details.
4. The Cashier press <<Save> button.
5. The use case goes back to step 3 in basic flow (Cashier).

A5: Open order [UC305_MTS]

1. The Cashier press on the required table.
2. The system redirect the cashier to the main sales page.
3. The Cashier input the order details.
4. The Cashier press <<Save> button.
5. The use case goes back to step 3 in basic flow (Cashier).

A6: Edit table layout [UC306_MTS]

1. The Cashier press <<Edit>> button.
2. The system display list of section and the table layout.
3. The Cashier drag and drop the table from the side to the layout and arrange it accordingly.
4. The Cashier enter the table name.
5. The Cashier choose the predefined order.
6. The Cahier

	<ol style="list-style-type: none"> 7. The Cashier press <<Save> button. 8. The system update the table layout. 9. The use case goes back to step 3 in basic flow (Cashier). <p>A7: Delete table [UC307_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display list of section and the table layout. 3. The Cashier long press the existing table. 4. The Cashier press the <<Delete>> button. 5. The system delete the table. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A8: Edit table [UC307_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display list of section and the table layout. 3. The Cashier long press the existing table. 4. The Cashier update the table details. 5. The Cashier press the <<Save>> button. 6. The system save the table. 7. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	Not applicable.
Post-Conditions	The table layout are updated and displayed in the system.

3.4.4.4 Manage Items Use Case Description

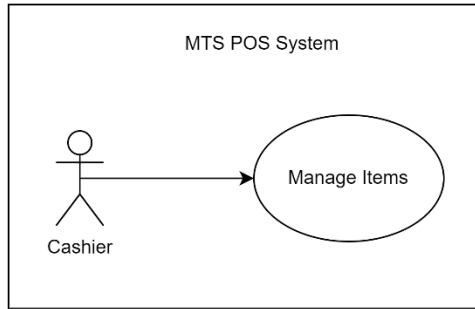


Figure 14 Use Case Manage Items

Table 6 Use Case Description for Manage Item

Use Case ID	UC400-MTS
Brief Description	This use case is used by cashier to add, edit, delete item.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Manage Item menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Manage Item menu. 2. The system display the list of existing item. 3. The Cashier can do the following options: 4. Add new item [A1: Add item] 5. Edit item [A2: Edit item] 6. Delete item [A3: Delete item] 7. The use case ends.
Alternative Flow	<p>A1: Add item [UC401_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Create Item>> button. 2. The system display Add New Item form. 3. The Cashier key in item details.

	<ol style="list-style-type: none"> 4. The Cashier press <<Save>> button. 5. The system save new item details. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit item [UC402_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display Edit item form. 3. The Cashier edit item details. 4. The Cashier press <<Save>> button. 5. The Cashier save edited item details. 6. The use case continues to step 3 in basic flow (Cashier). <p>A3: Delete item [UC403_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Delete>> button. 2. The system display delete confirmation. 3. The Cashier press <<Yes>> button. 4. The system delete item record. 5. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	The cashier aborts the use case: The system does not delete the item. The use case ends.
Post-Conditions	The item are updated and displayed in the system.

3.4.4.5 Manage Receipt Use Case Description

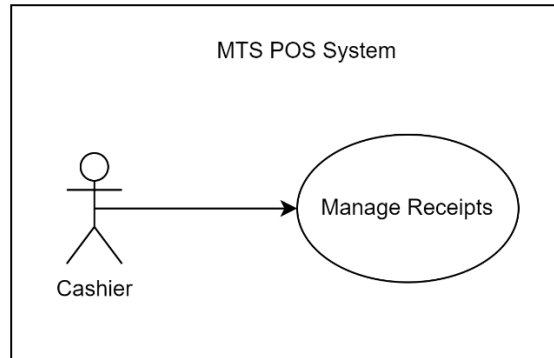


Figure 15 Use Case Manage Receipts

Table 7 Use Case Description for Manage Receipts

Use Case ID	UC500-MTS
Brief Description	This use case is used by cashier to manage receipts.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Receipts menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Receipts menu. 2. The system display the list of receipts. 3. The cashier can do the following options: <ol style="list-style-type: none"> a. View receipt details [A1: View receipt details] b. Send receipt to email [A2: Send receipt to email] c. Refund receipt [A3: Refund receipt] 4. The use case ends.
Alternative Flow	<p>A1: View receipt details [UC501_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press any of the receipt from the list of receipts. 2. The system display the details of the receipt.

	<p>3. The use case goes back to step 3 in basic flow (Cashier).</p> <p>A2: Send receipt to email [UC502_MTS]</p> <ol style="list-style-type: none"> 1. The cashier already open specific receipt. 2. The cashier press the <<Send Email>> button. 3. The system display popup to enter the email. 4. The cashier enter the customer email. 5. The cashier press <<Send>> button 6. The system send the receipt to the customer email. 7. The use case goes back to step 3 in basic flow (Cashier). <p>A3: Send receipt to email [UC503_MTS]</p> <ol style="list-style-type: none"> 1. The cashier already open specific receipt. 2. The cashier press the <<Refund>> button. 3. The system display popup for confirmation. 4. The cashier press the <<Yes>> button. 5. The system change the receipt status to refund. 6. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	Not applicable.
Post-Conditions	Not applicable.

3.4.4.6 Manage Outlets Use Case Description

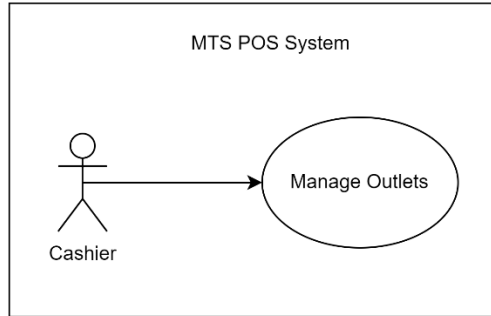


Figure 16 Use Case Manage Outlets

Table 8 Use Case Description for Manage Outlets

Use Case ID	UC600-MTS
Brief Description	This use case is used by cashier to add, edit, delete outlet.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Manage Outlet menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Manage Outlet menu. 2. The system display the list of existing outlet. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a. Add new outlet [A1: Add outlet] b. Edit outlet [A2: Edit outlet] c. Delete outlet [A3: Delete outlet] <p>The use case ends.</p>
Alternative Flow	<p>A1: Add outlet [UC601_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Create Outlet>> button. 2. The system display Add New Outlet form.

	<ol style="list-style-type: none"> 3. The Cashier key in outlet details. 4. The Cashier press <<Save>> button. 5. The system save new outlet details. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit outlet [UC602_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display Edit outlet form. 3. The Cashier edit outlet details. 4. The Cashier press <<Save>> button. 5. The Cashier save edited outlet details. 6. The use case continues to step 3 in basic flow (Cashier). <p>A3: Delete outlet [UC603_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Delete>> button. 2. The system display delete confirmation. 3. The Cashier press <<Yes>> button. 4. The system delete outlet record. 5. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	The cashier aborts the use case: The system does not delete the outlet. The use case ends.
Post-Conditions	The outlet are updated and displayed in the system.

3.4.4.7 Manage Discount Use Case Description

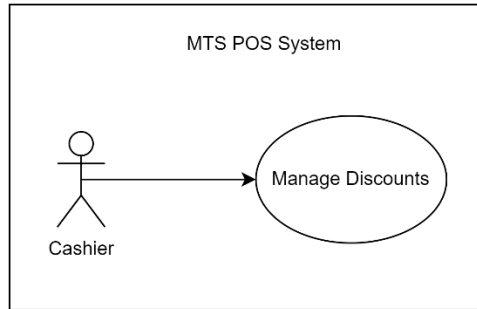


Figure 17 Use Case Mange Discounts

Table 9 Use Case Description for Manage Category

Use Case ID	UC700-MTS
Brief Description	This use case is used by cashier to add, edit, delete category.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Manage Category menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Manage Category menu. 2. The system display the list of existing category. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a. Add new category [A1: Add category] b. Edit category [A2: Edit category] c. Delete category [A3: Delete category] <p>The use case ends.</p>
Alternative Flow	<p>A1: Add category [UC701_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Create Category>> button. 2. The system display Add New Category form.

	<ol style="list-style-type: none"> 3. The Cashier key in category details. 4. The Cashier press <<Save>> button. 5. The system save new category details. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit category [UC702_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display Edit category form. 3. The Cashier edit category details. 4. The Cashier press <<Save>> button. 5. The Cashier save edited category details. 6. The use case continues to step 3 in basic flow (Cashier). <p>A3: Delete category [UC703_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Delete>> button. 2. The system display delete confirmation. 3. The Cashier press <<Yes>> button. 4. The system delete category record. 5. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	The cashier aborts the use case: The system does not delete the category. The use case ends.
Post-Conditions	The category are updated and displayed in the system.

3.4.4.8 Manage Category Use Case Description

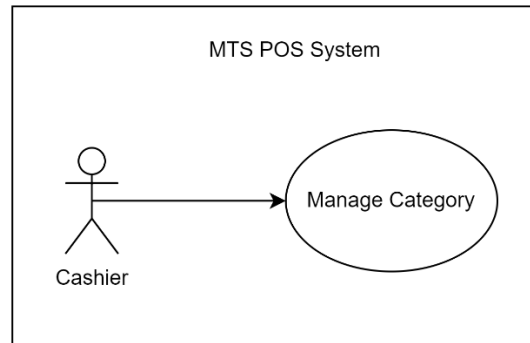


Figure 18 Use Case Manage Category

Table 10 Use Case Description for Manage Category

Use Case ID	UC800-MTS
Brief Description	This use case is used by cashier to add, edit, delete category.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Manage Category menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Manage Category menu. 2. The system display the list of existing category. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a. Add new category [A1: Add category] b. Edit category [A2: Edit category] c. Delete category [A3: Delete category] 4. The use case ends.
Alternative Flow	<p>A1: Add category [UC801_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Create Category>> button. 2. The system display Add New Category form.

	<ol style="list-style-type: none"> 3. The Cashier key in category details. 4. The Cashier press <<Save>> button. 5. The system save new category details. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit category [UC802_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display Edit category form. 3. The Cashier edit category details. 4. The Cashier press <<Save>> button. 5. The Cashier save edited category details. 6. The use case continues to step 3 in basic flow (Cashier). <p>A3: Delete category [UC803_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Delete>> button. 2. The system display delete confirmation. 3. The Cashier press <<Yes>> button. 4. The system delete category record. 5. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	The cashier aborts the use case: The system does not delete the category. The use case ends.
Post-Conditions	The category are updated and displayed in the system.

3.4.4.9 Manage Tax Use Case Description

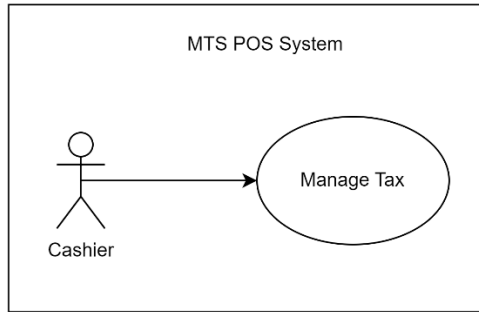


Figure 19 Use Case Manage Tax

Table 11 Use Case Description for Manage Tax

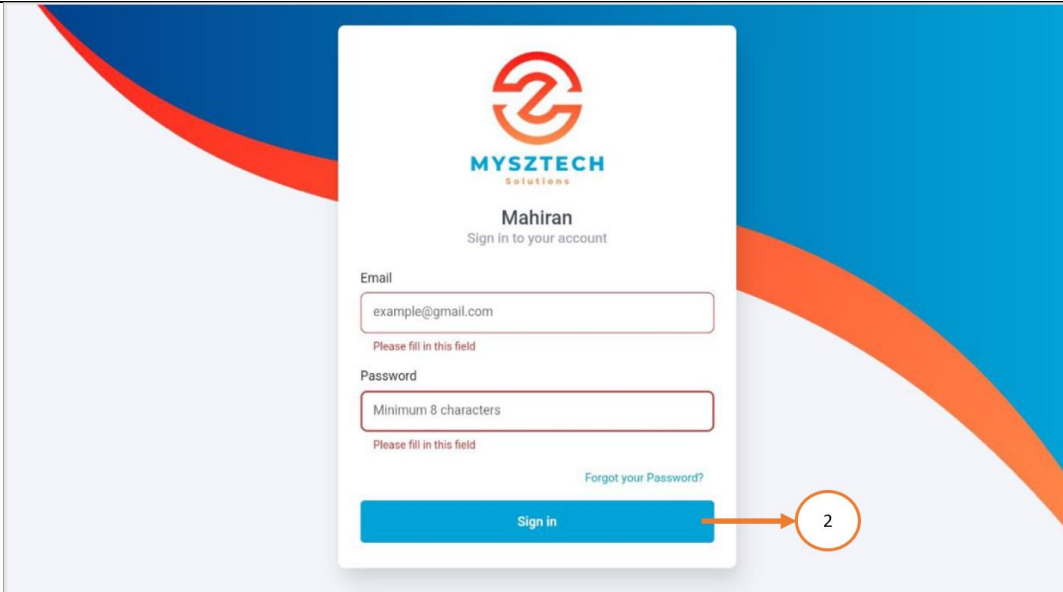
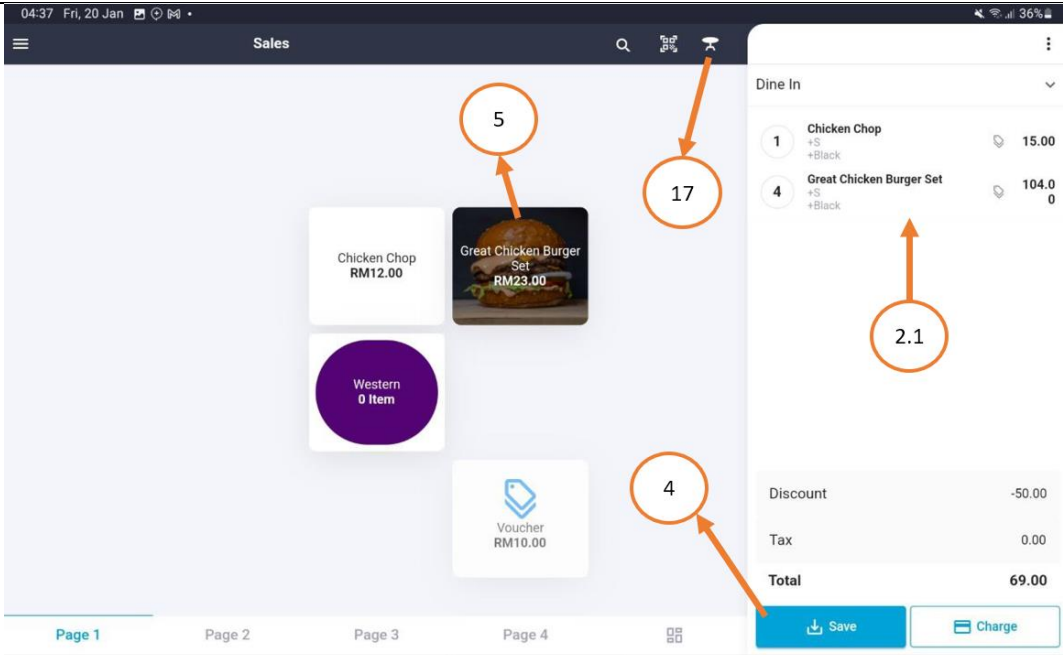
Use Case ID	UC900-MTS
Brief Description	This use case is used by cashier to add, edit, delete tax.
Actor	Cashier
Pre-Conditions	<ol style="list-style-type: none"> 1. User already logged in to the system. 2. User choose Manage Tax menu.
Basic Flow	<p>Cashier</p> <ol style="list-style-type: none"> 1. The use case begins when Cashier press Manage Tax menu. 2. The system display the list of existing taxes. 3. The Cashier can do the following options: <ol style="list-style-type: none"> a. Add new tax [A1: Add tax] b. Edit tax [A2: Edit tax] c. Delete tax [A3: Delete tax] 4. The use case ends.
Alternative Flow	<p>A1: Add tax [UC901_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Create Tax>> button. 2. The system display Add New Tax form. 3. The Cashier key in tax details.

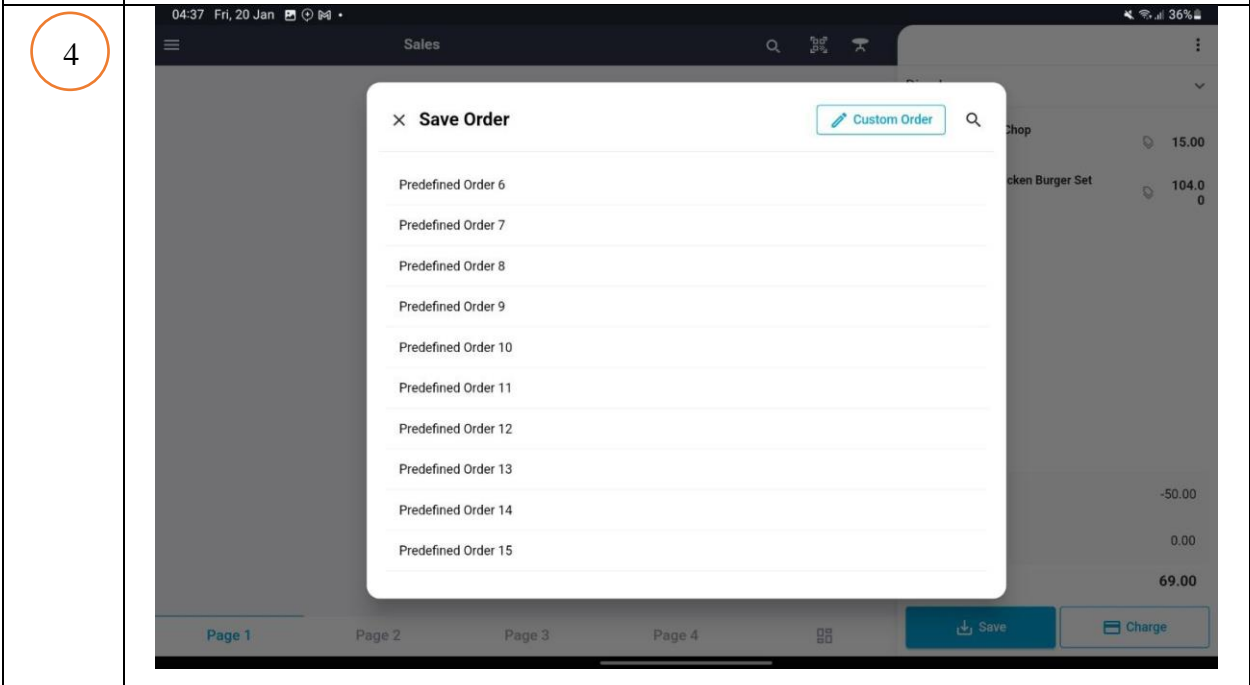
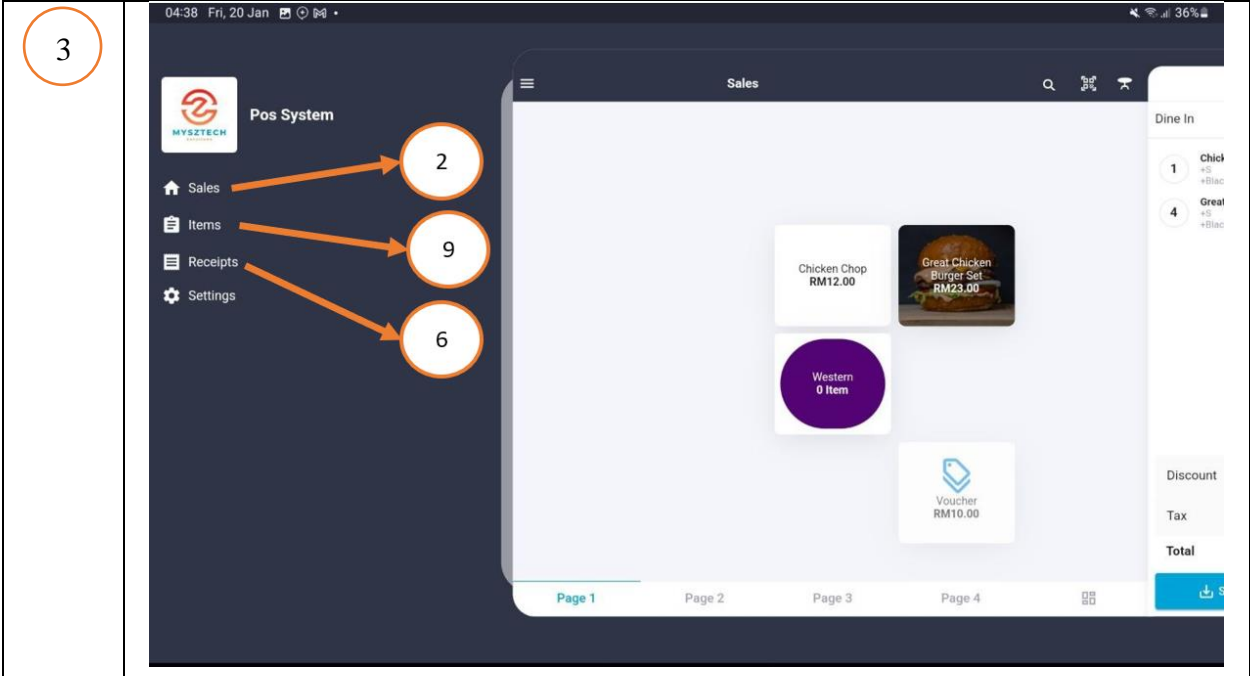
	<ol style="list-style-type: none"> 4. The Cashier press <<Save>> button. 5. The system save new tax details. 6. The use case goes back to step 3 in basic flow (Cashier). <p>A2: Edit tax [UC902_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Edit>> button. 2. The system display Edit tax form. 3. The Cashier edit tax details. 4. The Cashier press <<Save>> button. 5. The Cashier save edited tax details. 6. The use case continues to step 3 in basic flow (Cashier). <p>A3: Delete tax [UC903_MTS]</p> <ol style="list-style-type: none"> 1. The Cashier press <<Delete>> button. 2. The system display delete confirmation. 3. The Cashier press <<Yes>> button. 4. The system delete tax record. 5. The use case goes back to step 3 in basic flow (Cashier).
Exception Flow	The cashier aborts the use case: The system does not delete the tax. The use case ends.
Post-Conditions	The tax are updated and displayed in the system.

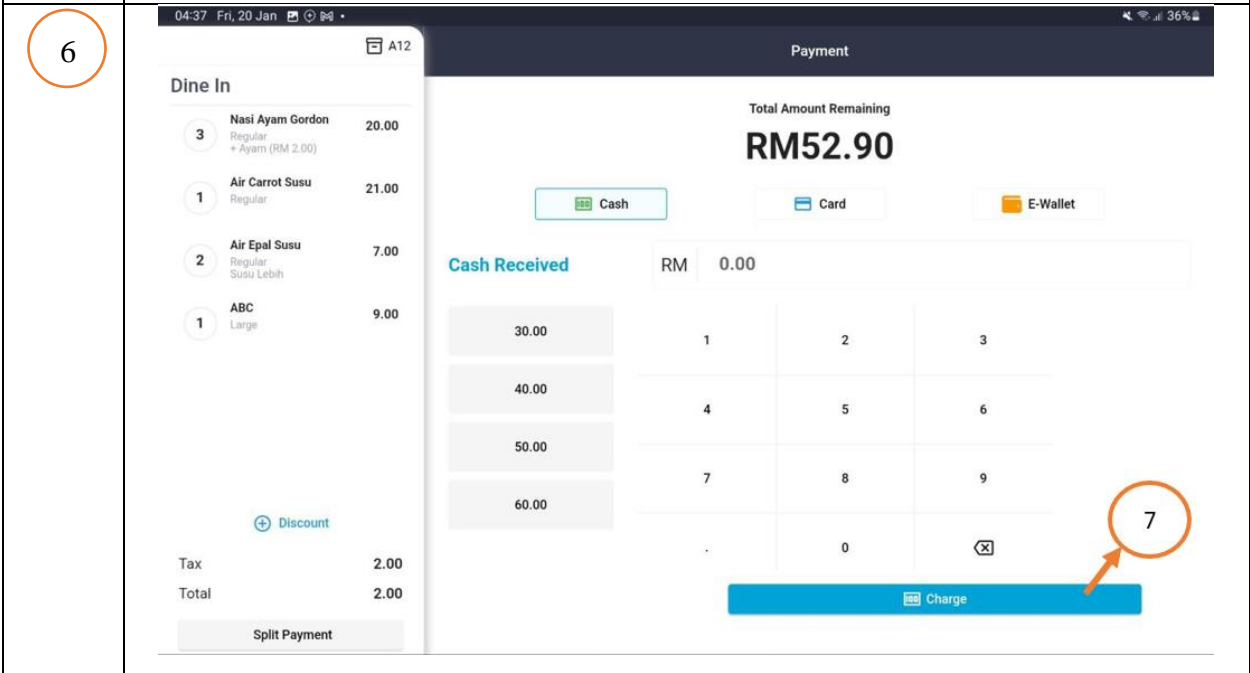
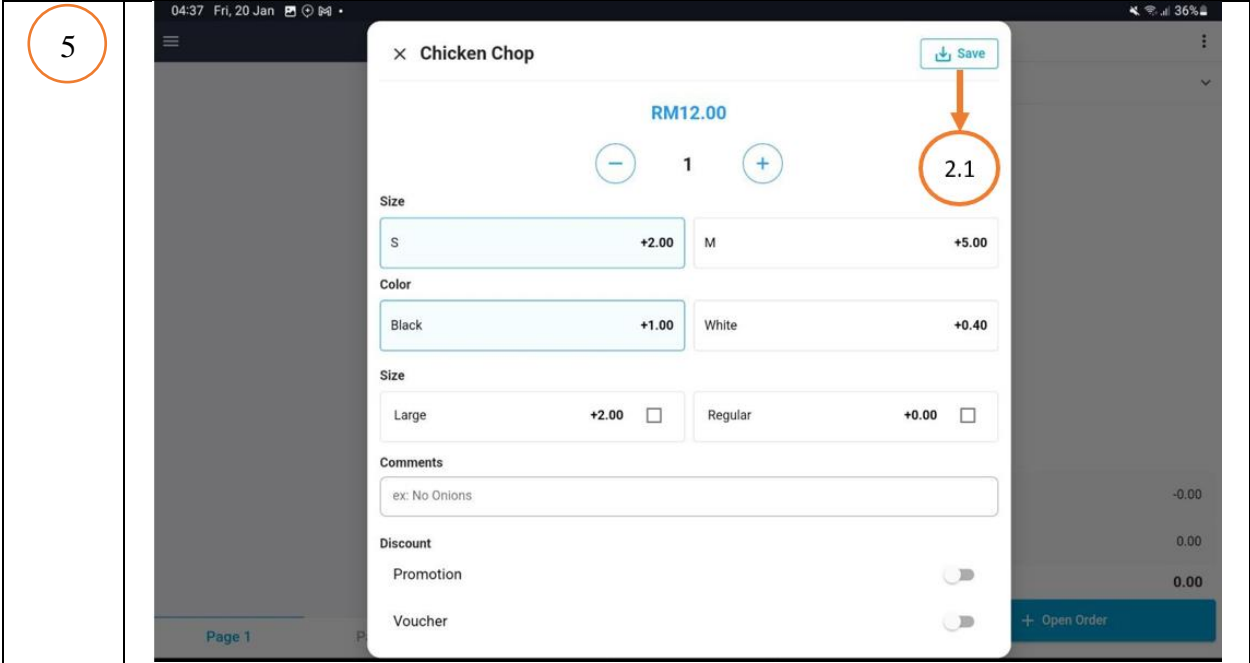
3.4.5 Storyboard

3.4.5.1 Storyboard for Cashier

Table 3.12 Storyboard for Cashier

No	User Interface
1	 <p>The image shows a sign-in screen for 'Mahiran' with the MYSZTECH Solutions logo. It features an email input field (containing 'example@gmail.com'), a password input field (with a 'Minimum 8 characters' hint), and a 'Sign in' button. A 'Forgot your Password?' link is also present. A circled '1' is in the top-left corner, and a circled '2' points to the 'Sign in' button.</p>
2	 <p>The image shows a 'Sales' screen with a grid of product cards: 'Chicken Chop RM12.00', 'Great Chicken Burger Set RM23.00', 'Western 0 Item', and 'Voucher RM10.00'. A right-hand panel displays a 'Dine In' order list with items '1 Chicken Chop +S +Black 15.00' and '4 Great Chicken Burger Set +S +Black 104.00'. Below the list are 'Discount -50.00', 'Tax 0.00', and 'Total 69.00'. At the bottom are 'Save' and 'Charge' buttons. A bottom navigation bar shows 'Page 1' through 'Page 4'. Circled numbers 5, 17, 2.1, and 4 point to specific elements: 5 points to the burger card, 17 points to the burger set item in the order list, 2.1 points to the burger set item in the order list, and 4 points to the 'Save' button.</p>





7

04:58 Fri, 20 Jan A12

Dine In

- 3 Nasi Ayam Gordon 20.00
Regular + Ayam (RM 2.00)
- 1 Air Carrot Susu 21.00
Regular
- 2 Air Epal Susu 7.00
Regular Susu Lebih
- 1 ABC 9.00
Large

Tax 2.00
Total 2.00

Split Payment

Total Paid **RM52.90** Change

RM7.10

Email Send Receipt

Print Receipt New Sale

2

8

05:13 Fri, 20 Jan #21 | 19 Oct 2022 - 11:46 PM

Receipts

Wednesday, 19 Oct 2022

- 188.17 11:46 PM #21

Tuesday, 18 Oct 2022

- 523.38 11:46 PM #1
- 537.41 11:46 PM #12
- 537.74 11:46 PM #13
- 284.94 11:46 PM #14
- 225.05 11:46 PM #15
- 696.21 11:46 PM #16
- 648.67 11:46 PM #17
- 100.30 11:46 PM #18

RM188.17 Normal

Sold By Hairi

Refund Send Receipt to Email Print Receipt

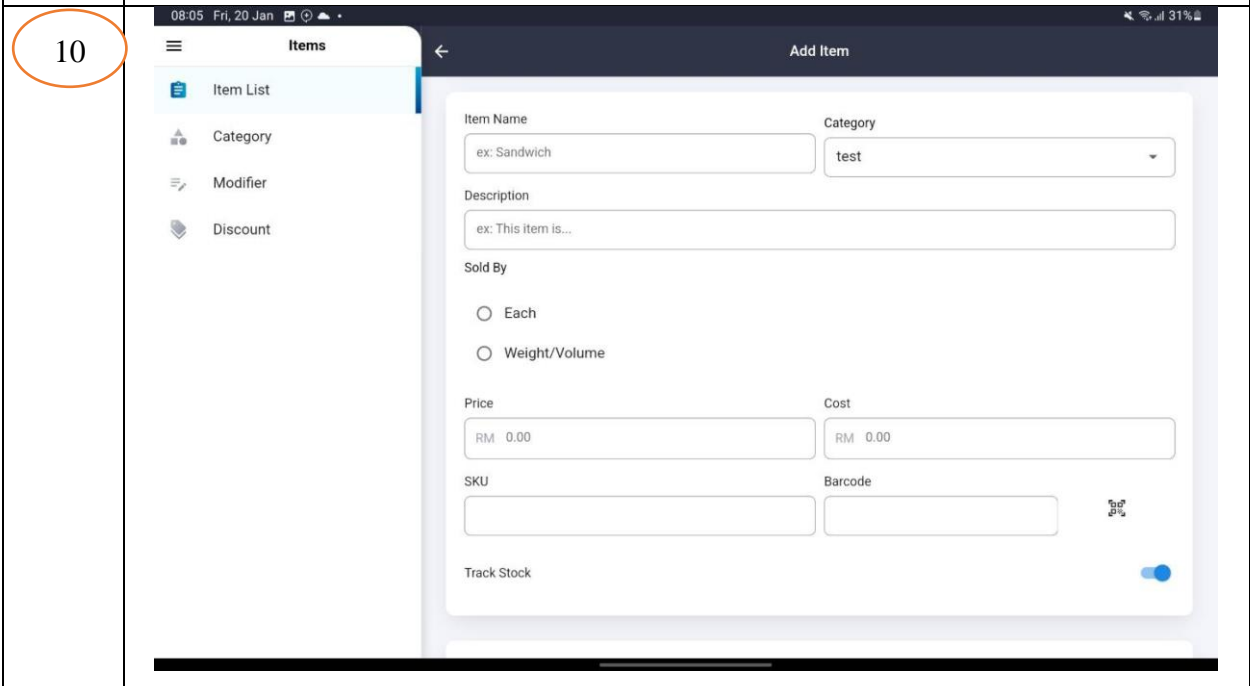
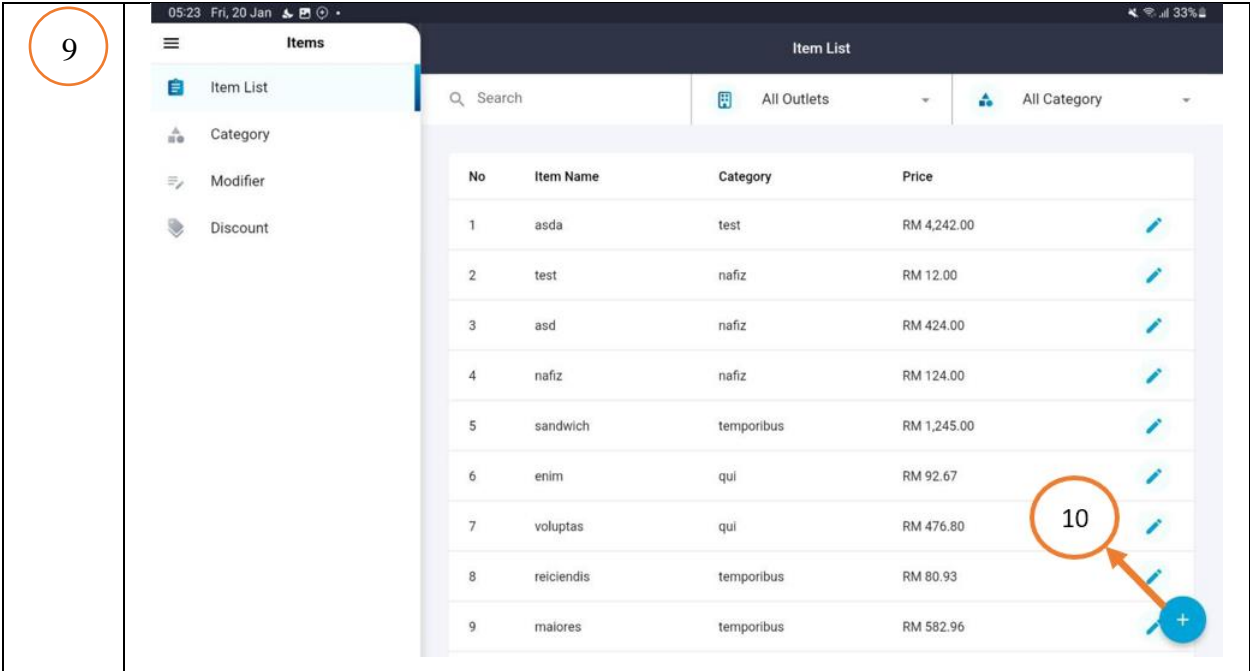
Wednesday, 19 Oct 2022 - 11:46 PM #21

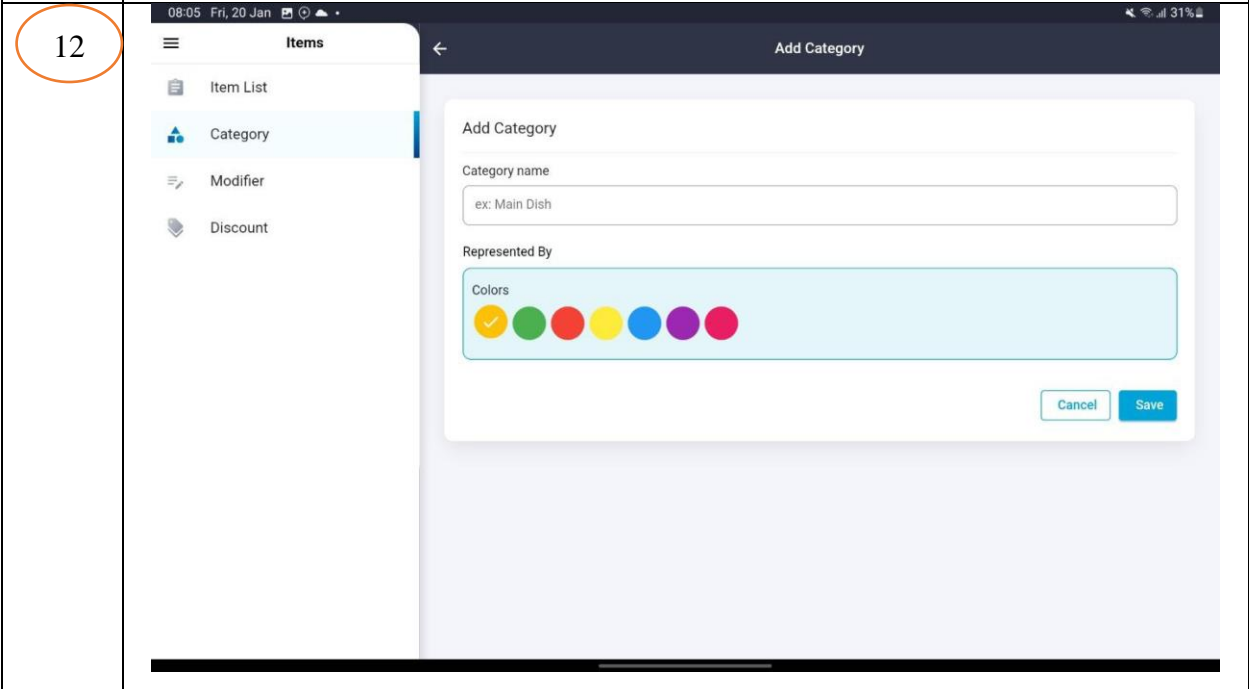
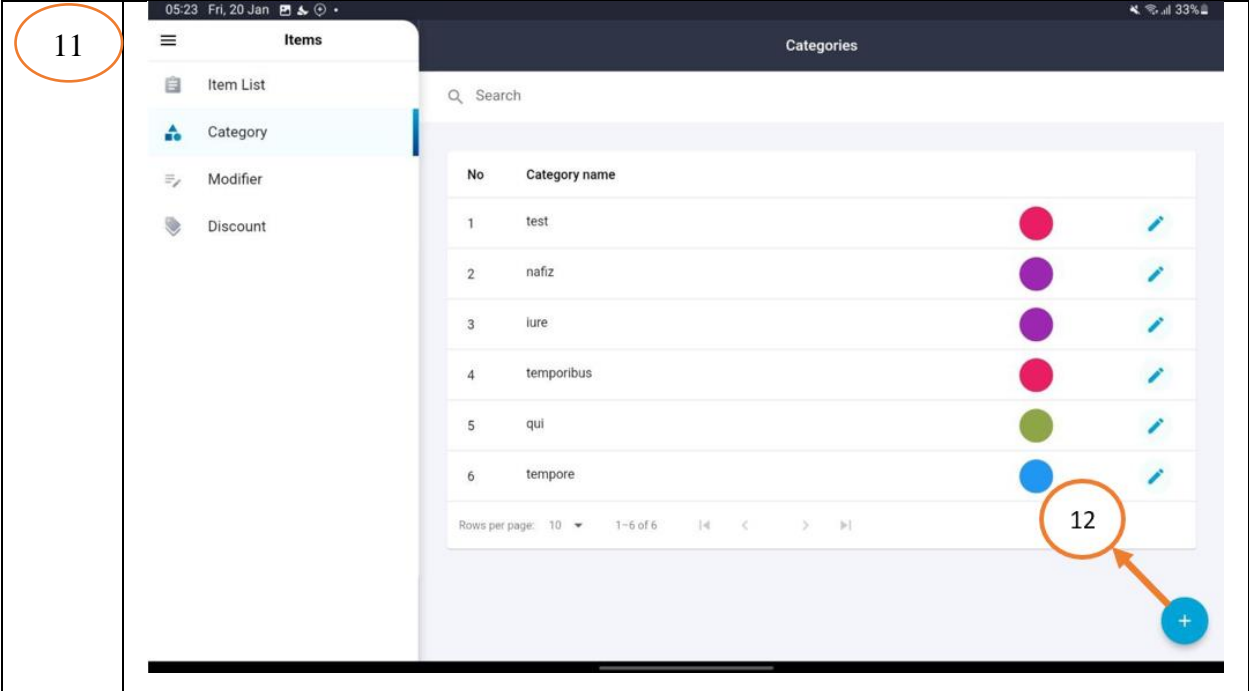
Order : pariatur
POS : HQ Cashier 1
Takeout

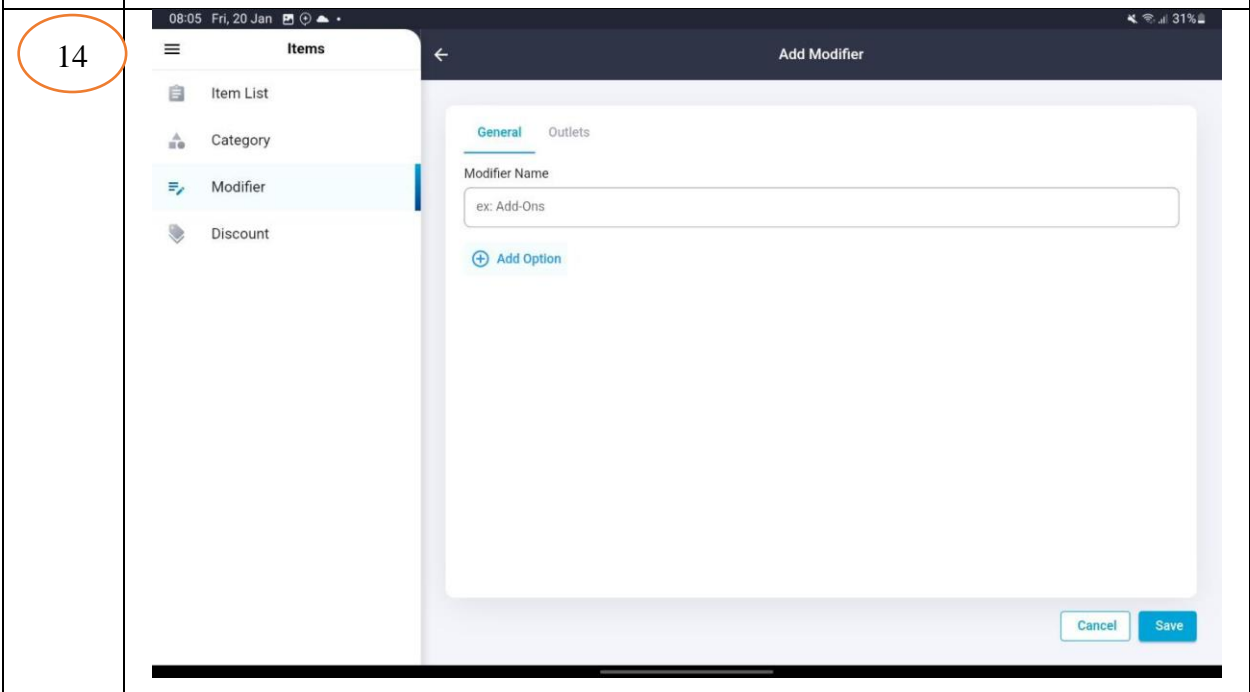
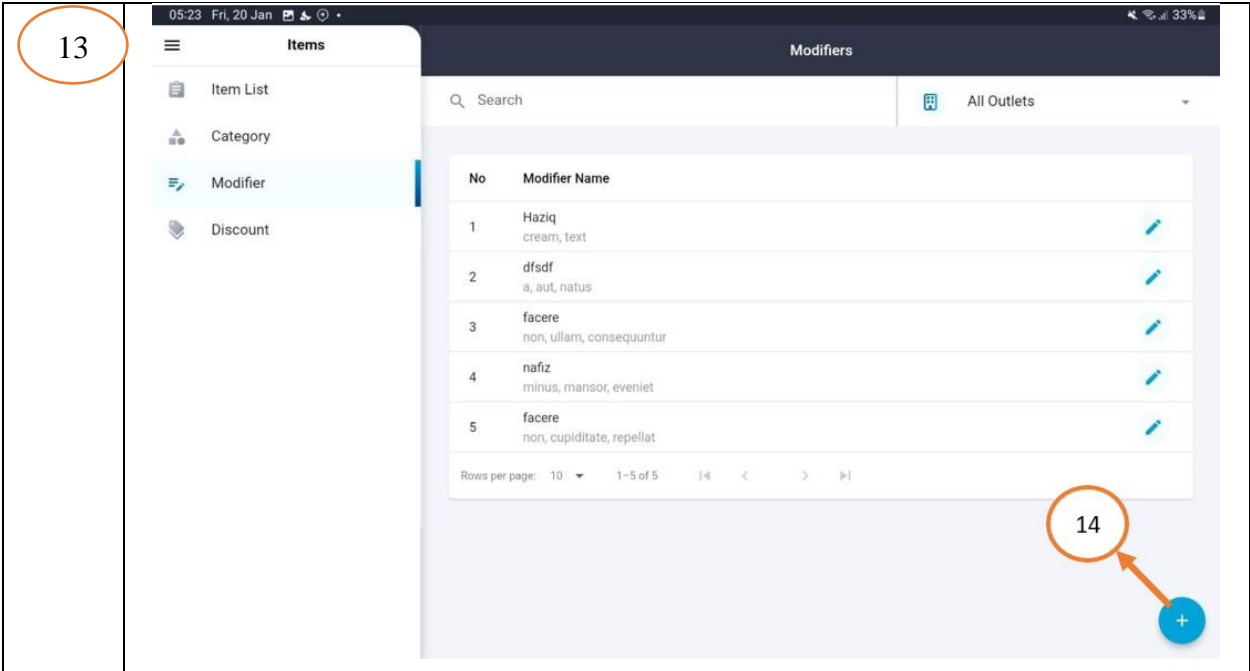
Items

- 4 natus 596.60 326.58
- 31383217 qui 167.03 685.61
- 73143 alias 638.46 668.78

Total 188.17
Cash 695.57
Change -507.40







15

Discounts

Search

All Outlets

No	Name	Value
1	424	RM424.00
2	est	4%
3	4242	RM6.00
4	ut	RM56.00
5	exercitationem	RM19.00
6	voluptas	RM88.00
7	aliquid	33%
8	ratione	RM3.00
9	autem	29%

16

16

Add Discount

General Outlets

Discount Name

ex: Discount 10%

Discount Type

Percentage

Amount

Value

0.00

Restricted Access

Cancel Save

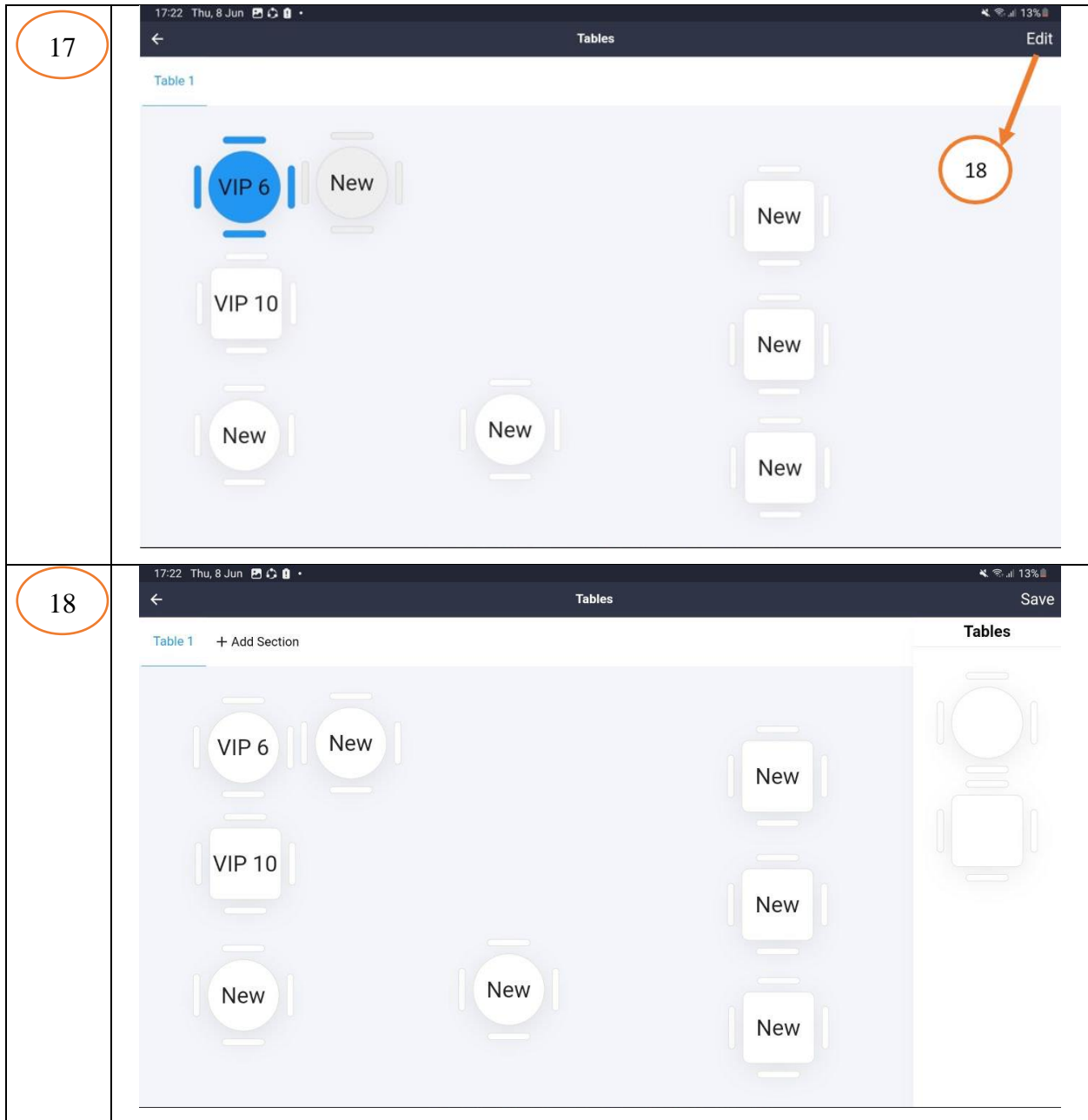


Table 3.12 Storyboard for Cashier shows the interfaces for cashier role. The numbering at the interface shows where the system will navigate once press it. Cashiers can login to the system using their unique credentials and then access the sales interface. This interface allows them to make sales, customize table layout, charge for the sale, and view a list of orders. To view previous receipts, cashiers can press the "receipts" tab, which will display a list of all previous sales.

Additionally, the "items" tab can be pressed to access the inventory management section, where cashiers can add, edit, and delete items, manage discounts, modifiers and categories.

3.4.6 User Interface

3.4.6.1 User Interface for Cashier

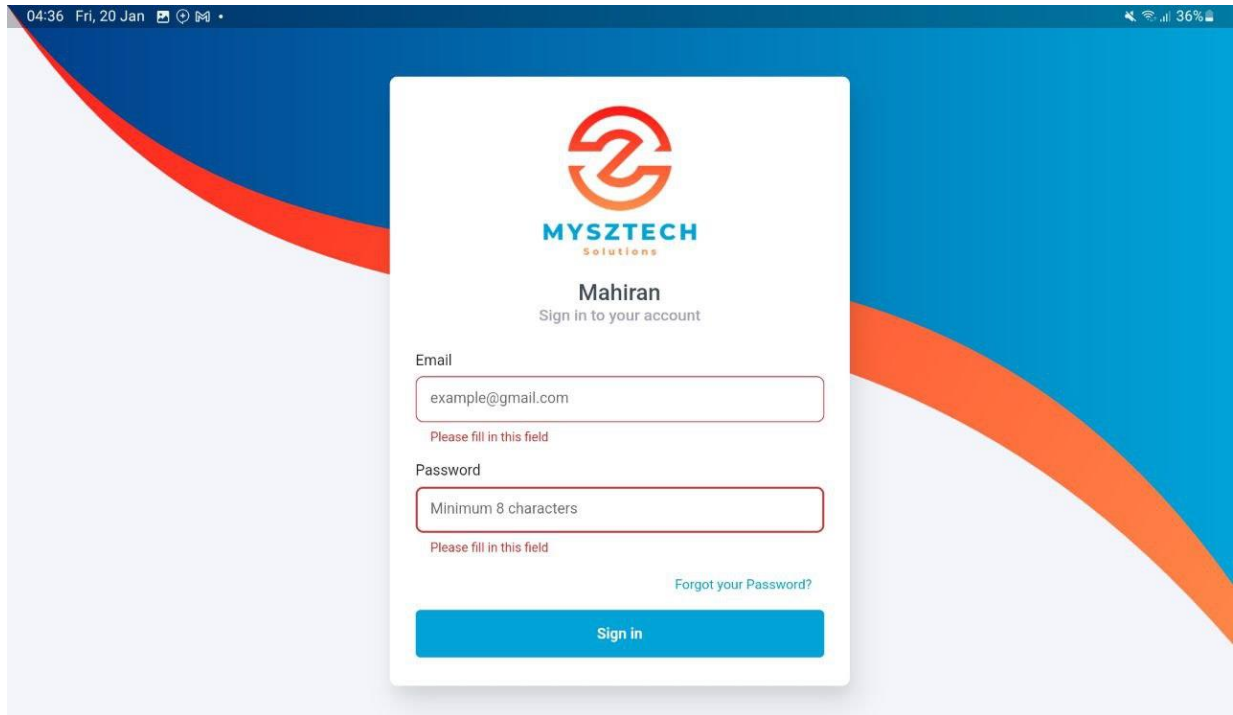


Figure 20 Login UI for Cashier

Figure 20 Login UI for Cashier shows the login interface for the cashier. The cashier needs to enter their email and password to login. If they forgot password, they can press the *Forgot your Password?* button. If the cashier doesn't enter the correct details, the system will show an invalid message.

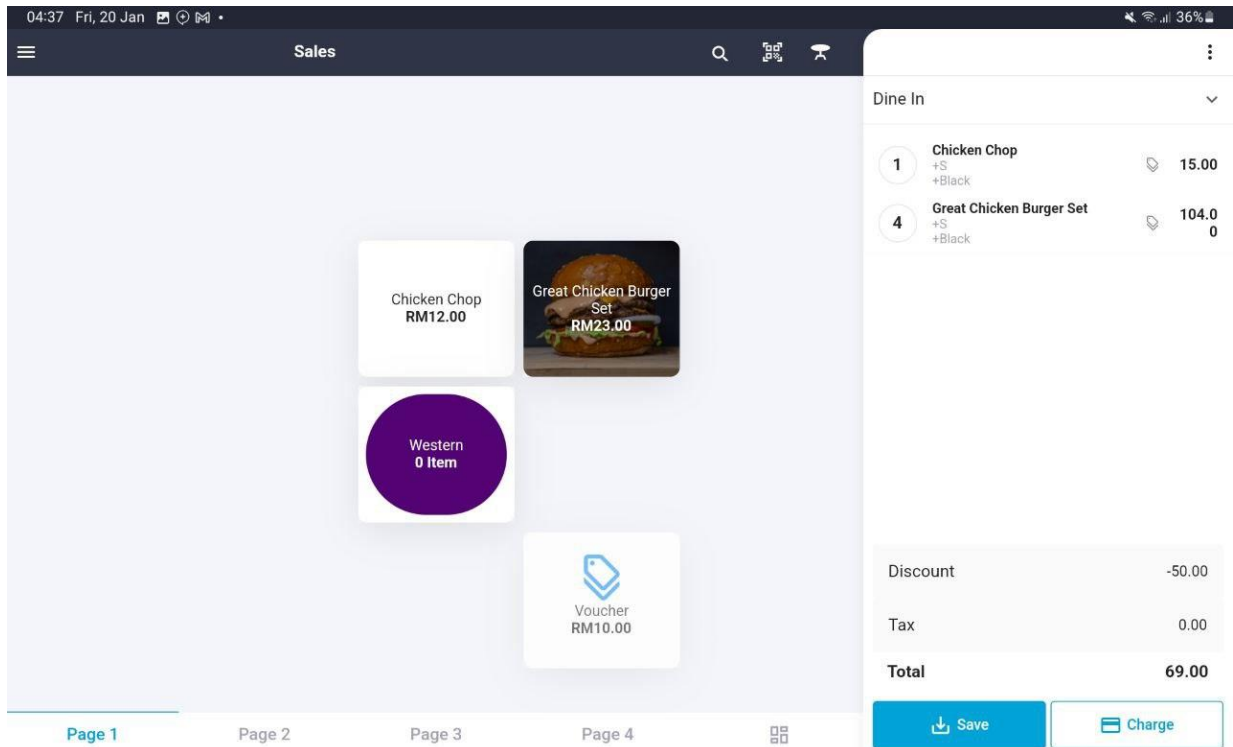


Figure 21 Main Sales UI

Figure 21 Main Sales UI shows the interface for the cashier to key in the sales. From this interface, the cashier can choose the menu requested by the customer along with the discount and tax if needed. The cashier can save the order first or directly charge the customer to make a payment. From the list on the right side, the cashier can see all the requested items by the customer and the total of the order.

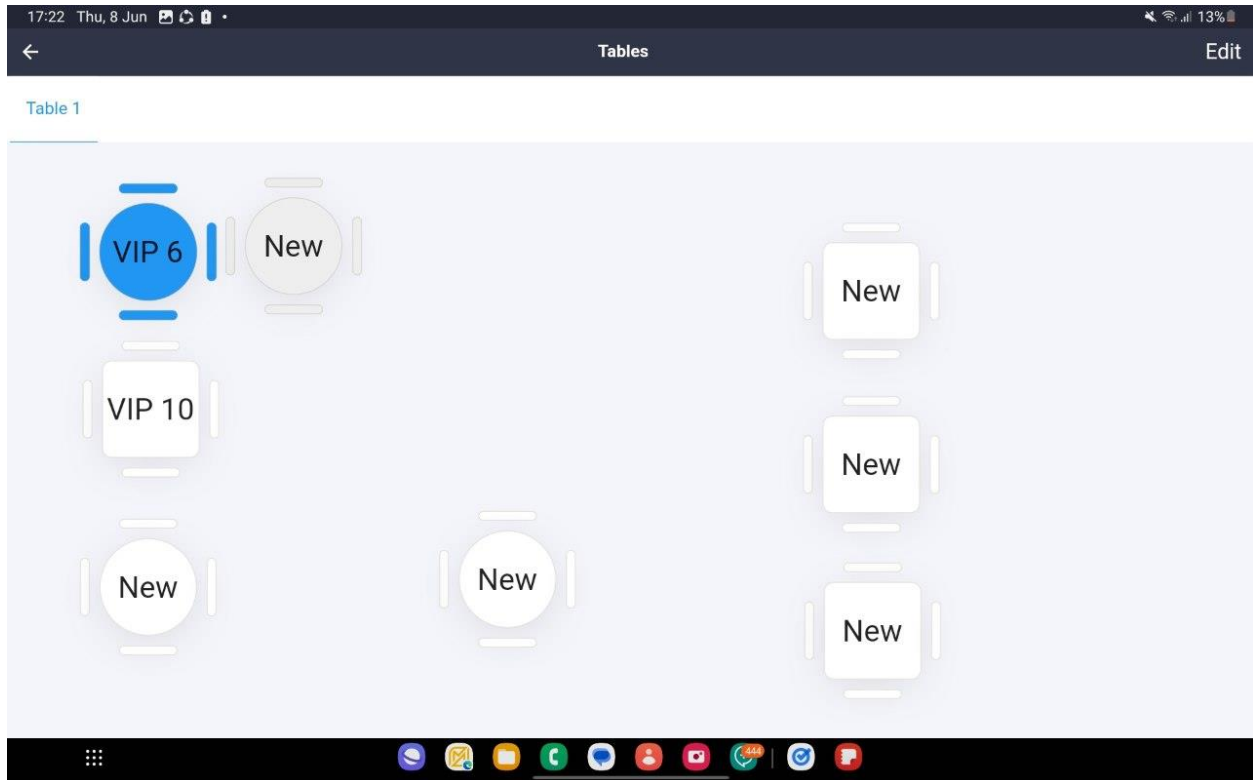


Figure 22 Table Layout UI

Figure 22 Table Layout UI shows the table layout that the cashier can use to make and order. There are an edit button on the top right. The table in blue color is the table that already has an order while, the gray table is the table that is unavailable, and the white table is the table is ready to be occupied.

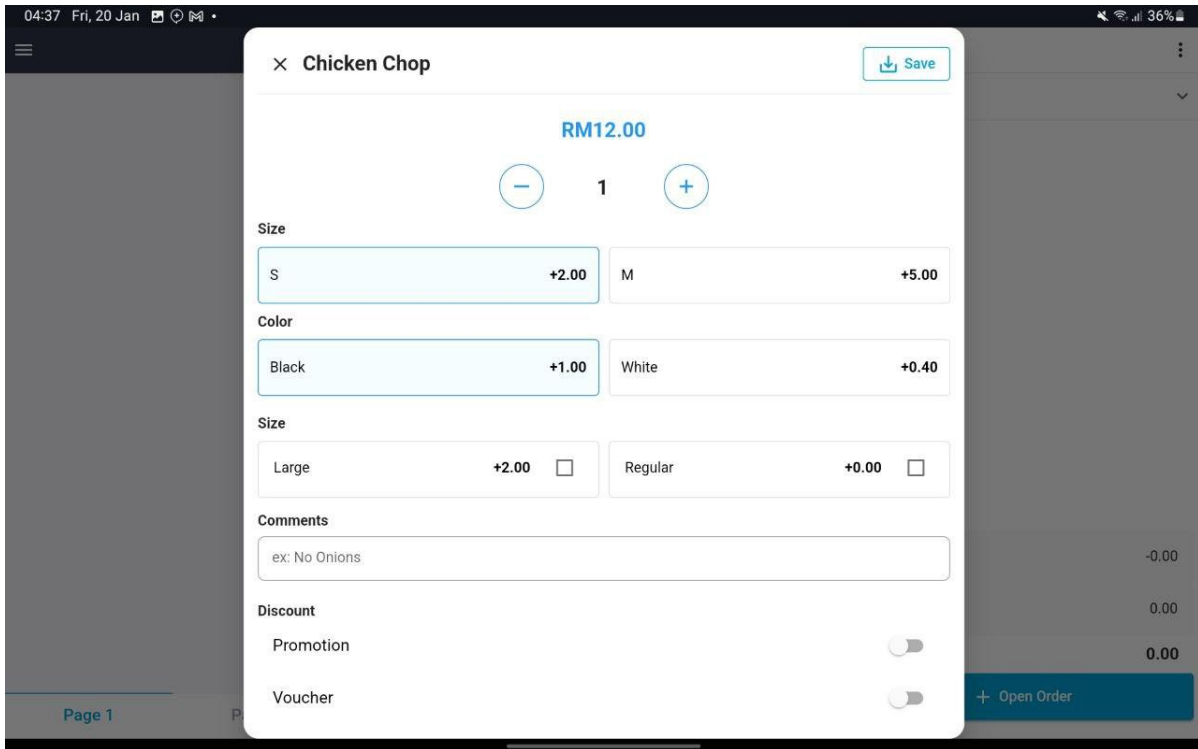


Figure 23 Item Details Popup UI

Figure 23 Item Details Popup UI shows the popup dialog that will be shown to the cashier once the cashier presses any of the menu. In this popup, the cashier needs to enter the quantity, choose the variants, choose the modifiers, enter comments, enable or disable discount and taxes. Once done key in all the details, the cashier needs to press *Save* button to add this item to the list of items on the right side.

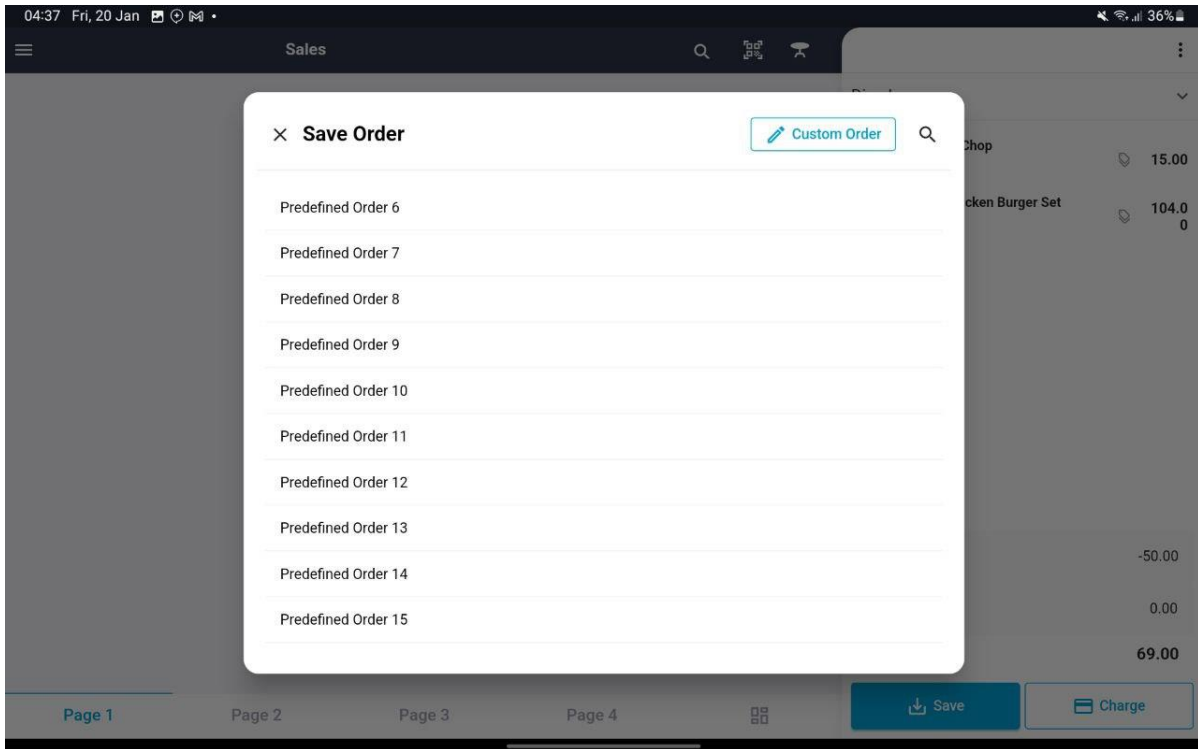


Figure 24 Popup when Press Save button UI

Figure 24 Popup when Press Save button UI shows the list of orders in the system. The cashier can choose an existing predefined order or create a new one. The predefined order can be a table or any name.

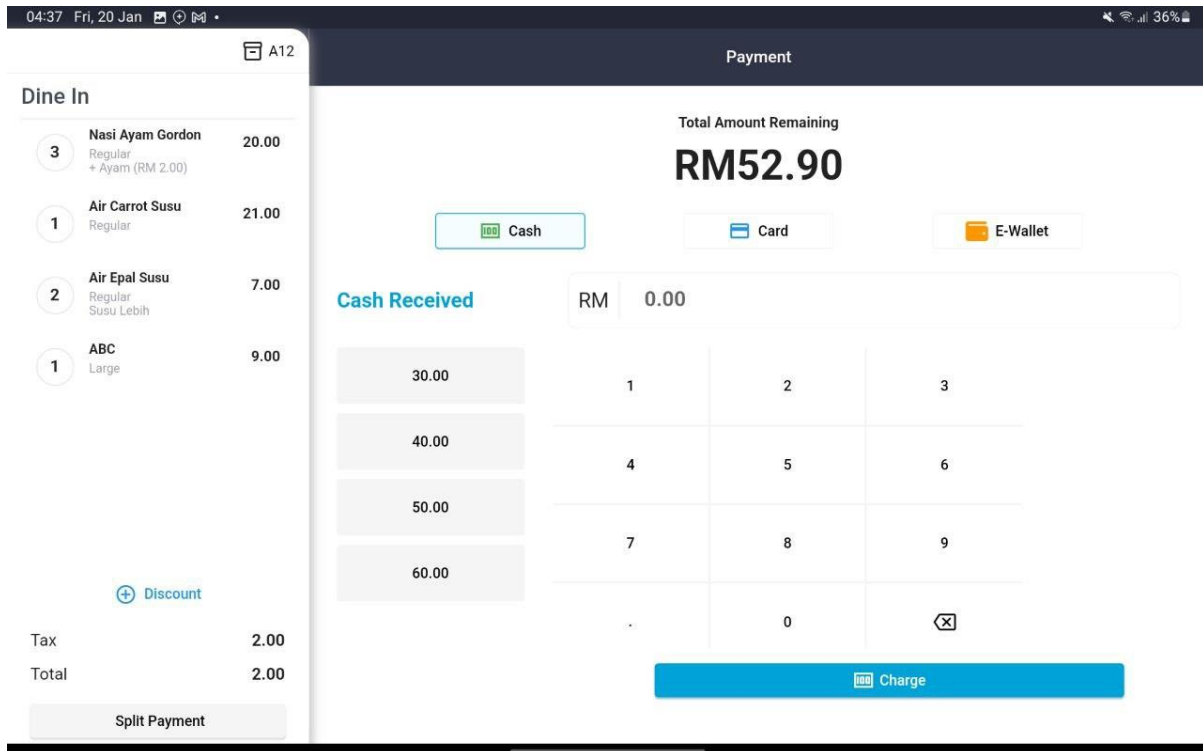


Figure 25 Order Details UI

Figure 25 Order Details UI shows the order details page. At this page, the cashier can enter the amount paid the customer either using e-wallet, cash or card. At the left side, the cashier will see the list of items for this order along with calculation and total.

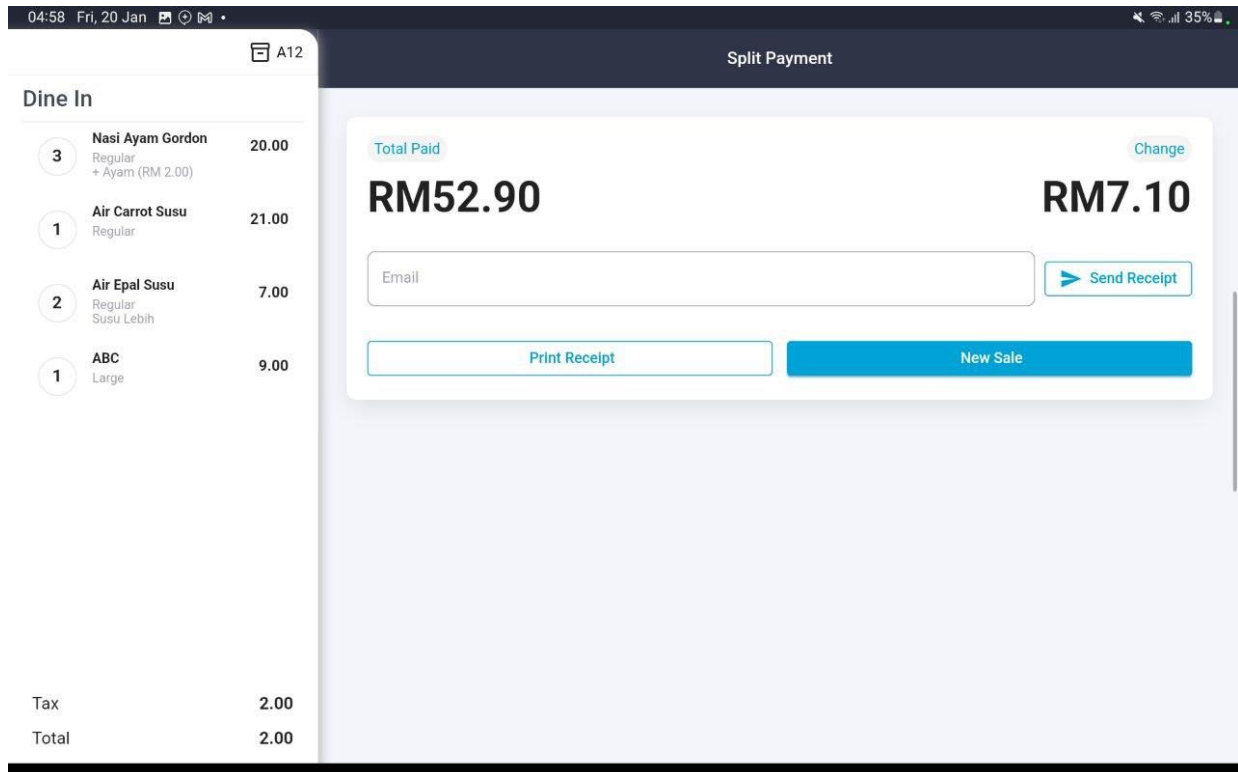


Figure 26 Payment Details UI

Figure 26 Payment Details UI shows the payment details interface along with the total paid and change need to be paid to the customer. The cashier can press print receipt to print the receipt using thermal printer or press *New Sale* button to make a new sale.

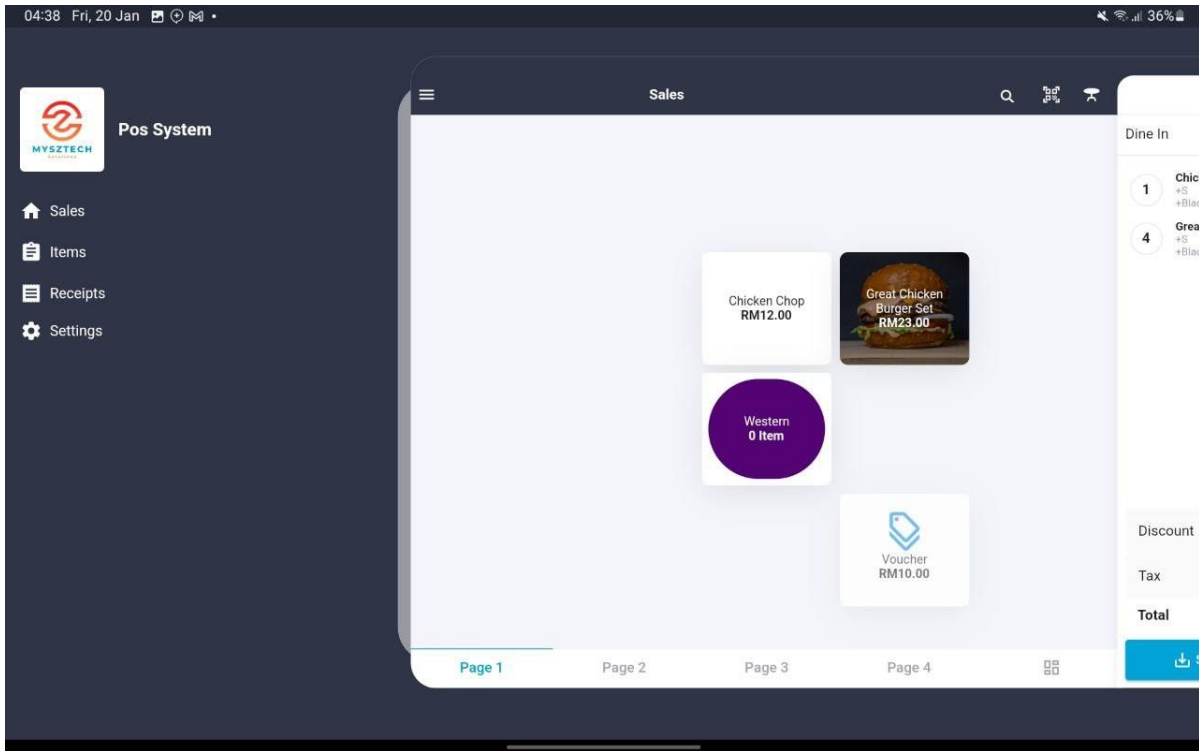


Figure 27 Menu in the Side Navigation

Figure 27 Menu in the Side Navigation shows the navigation tabs which are Sales Module, Items Module, Receipts Module and Settings Module. To access this navigation, the cashier needs to press the hamburger logo at the top left.

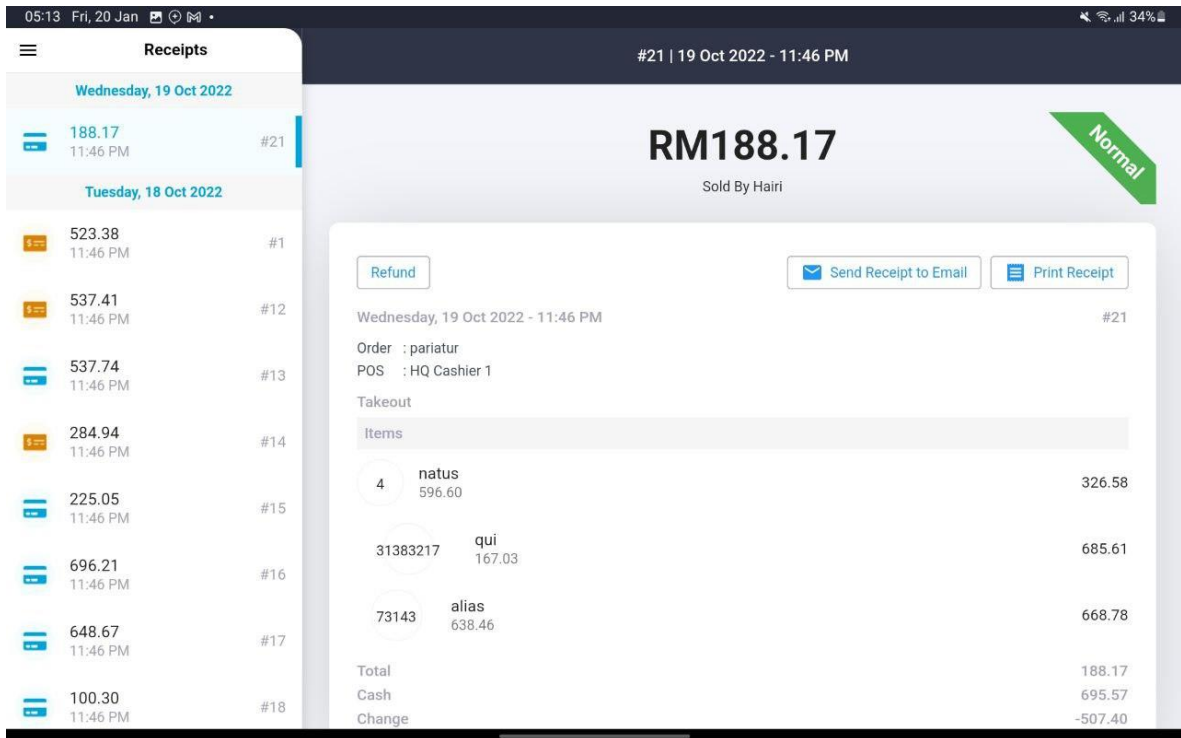


Figure 28 List of Receipt UI

Figure 28 List of Receipt UI shows the list of receipts with the details and status. At this interface, the cashier can view the list of previous receipt, the time and the details of the receipt.

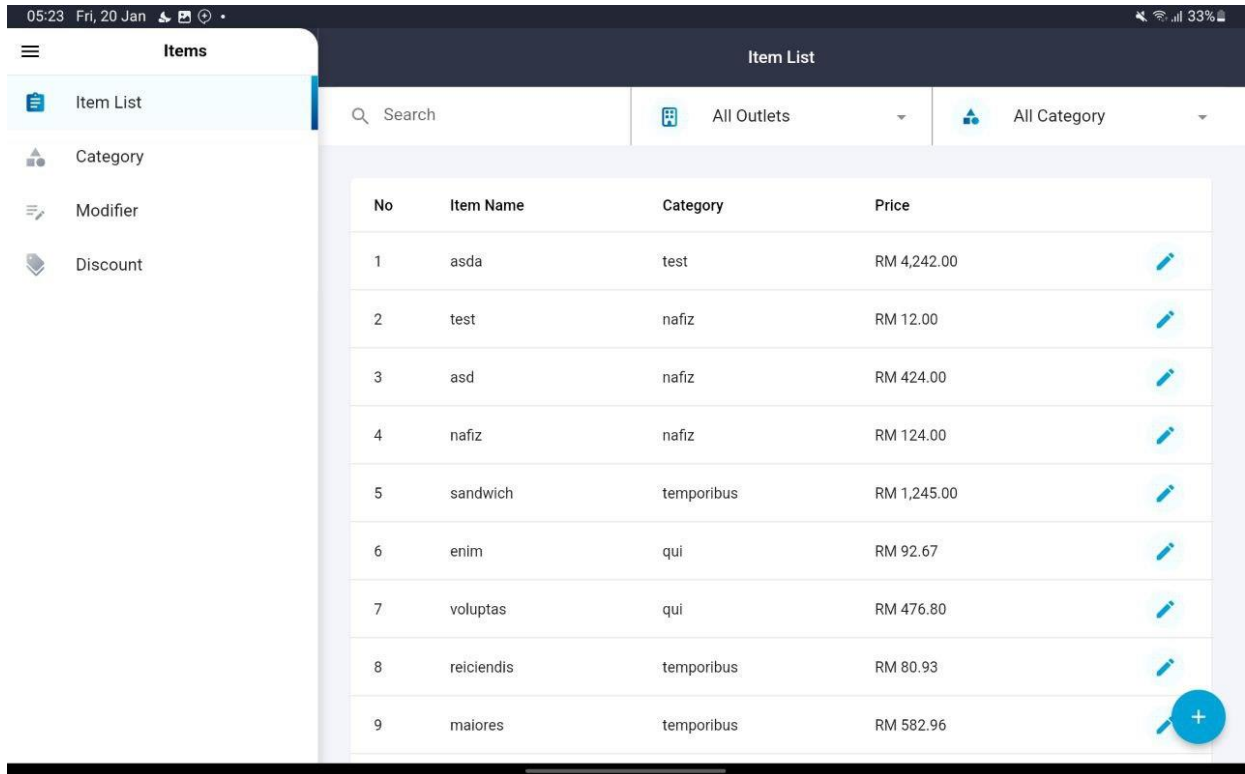


Figure 29 Item List UI

Figure 29 Item List UI shows the list of items or menu. This interface will allow the cashier to view all the items with the item name, category and price.

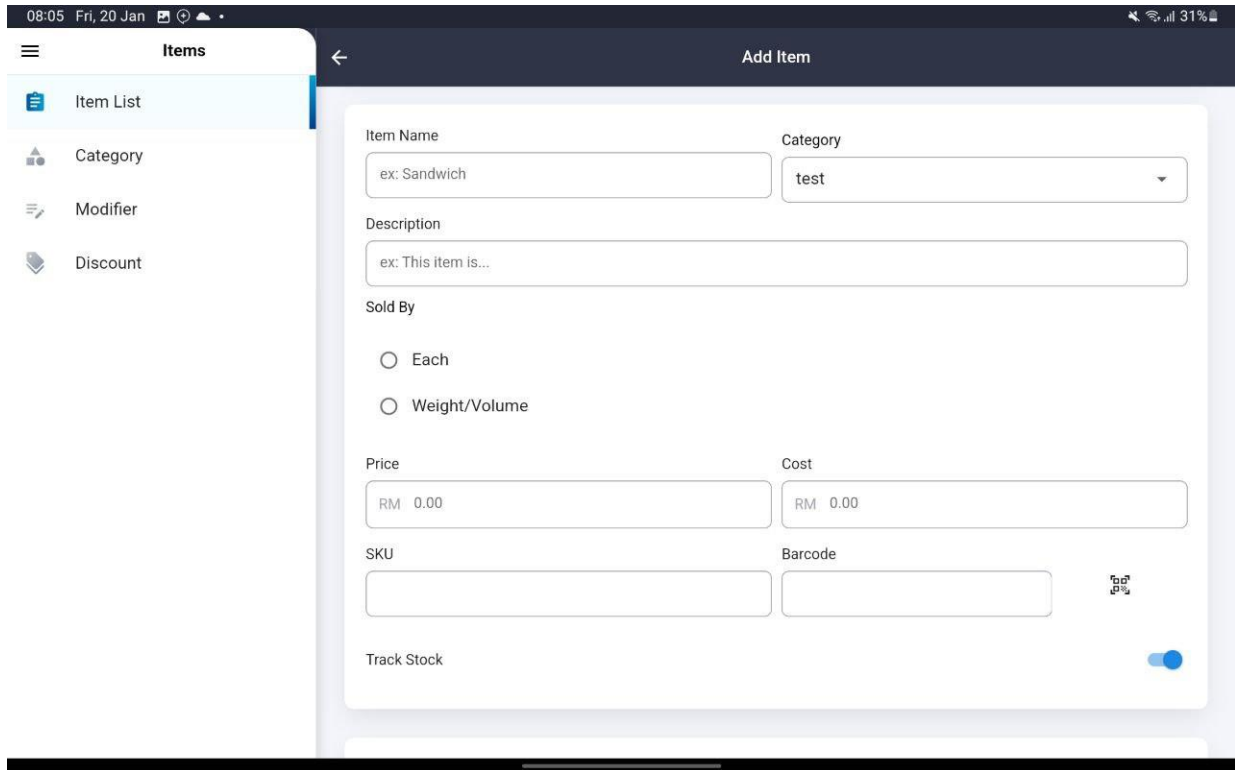


Figure 30 Add Item UI

Figure 30 Add Item UI shows the add item interface. The cashier needs to enter all the details required.

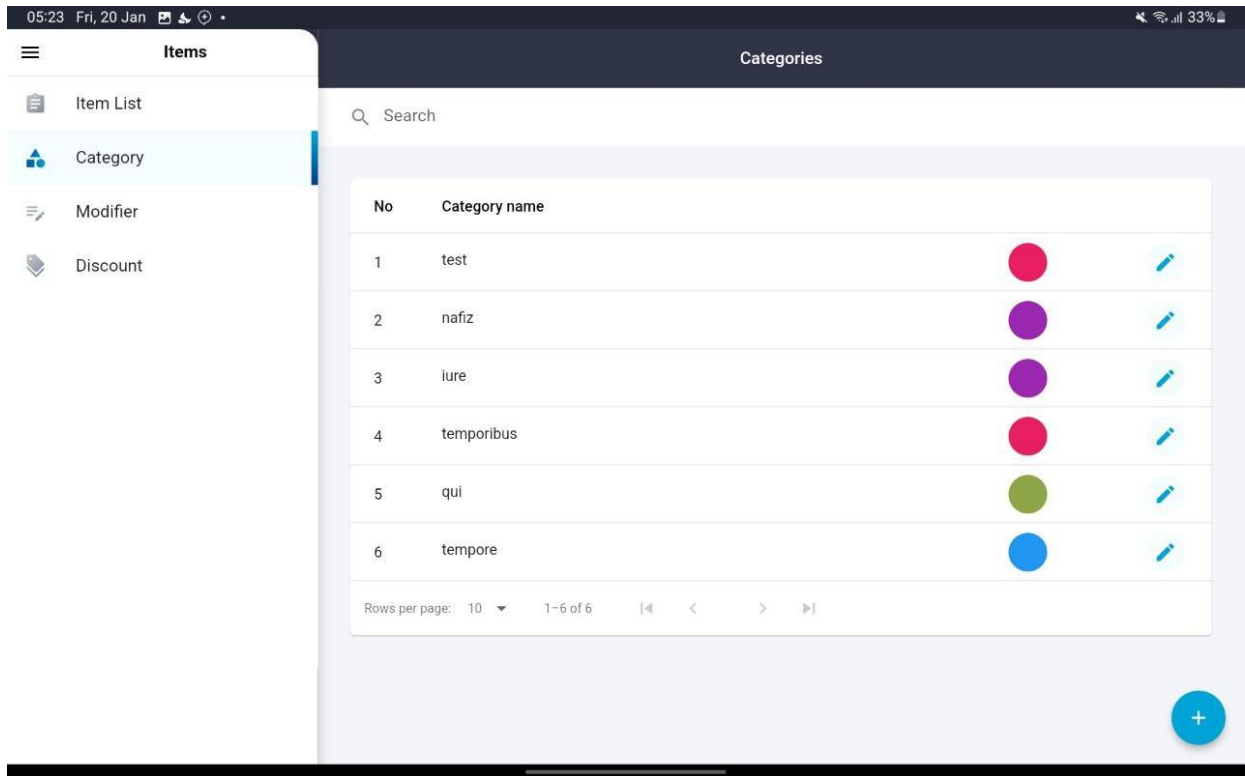


Figure 31 List Category UI

Figure 31 List Category UI shows the list of categories. This interface shows the list of categories that will be used to categorize all of the menu.

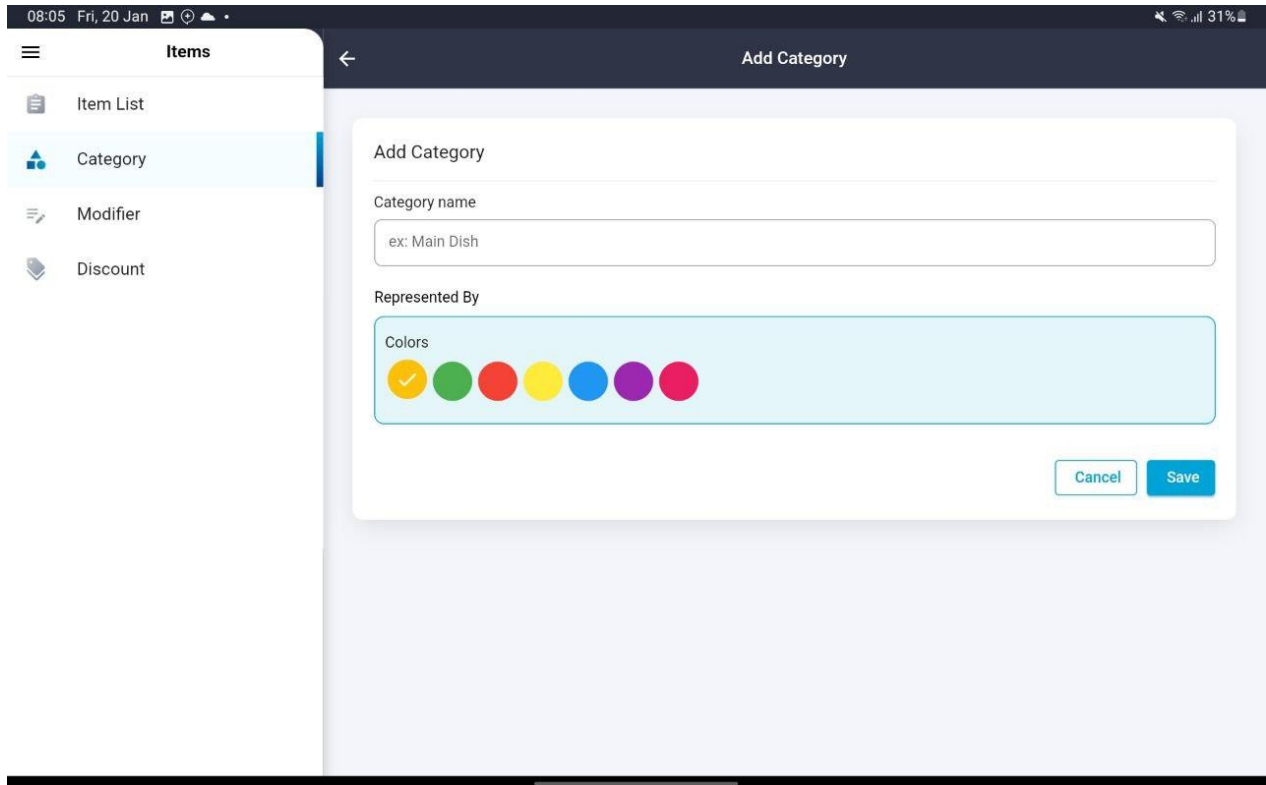


Figure 32 Add Category UI

Figure 32 Add Category UI shows the add category interface. The cashier needs to enter the category name and color.

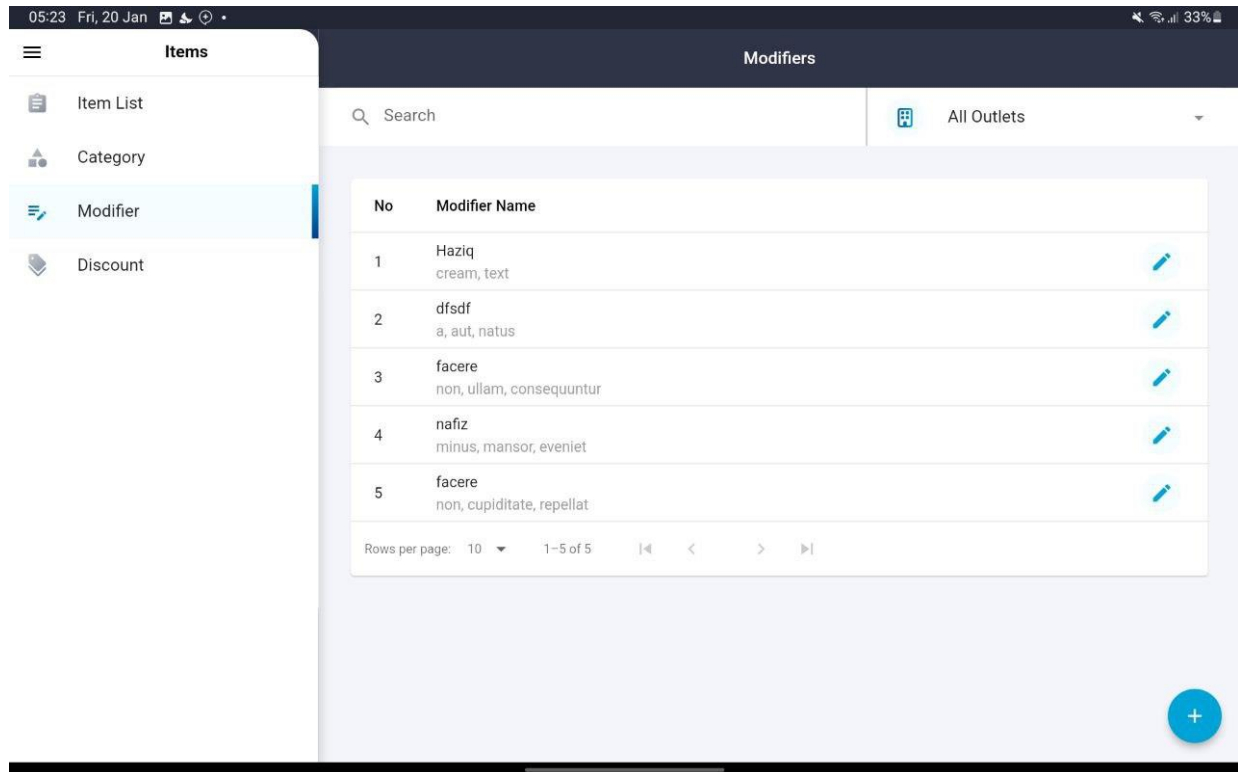


Figure 33 List Modifiers UI

Figure 33 List Modifiers UI shows the list of modifiers. It shows the name of the modifier, and the list of options. The cashier also can search for the modifiers if needed.

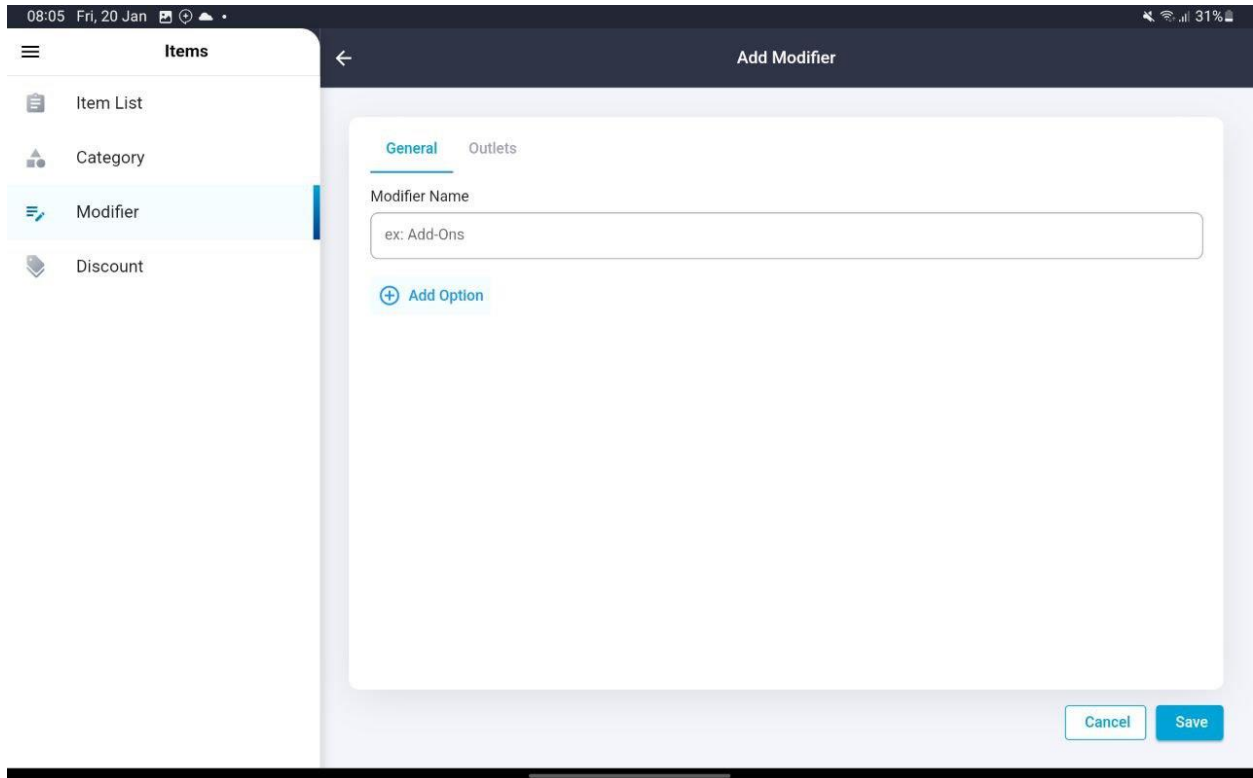


Figure 34 Add Modifier UI

Figure 34 Add Modifier UI shows the add modifier interface. The cashier needs to enter the modifier name and add an option to that modifier.

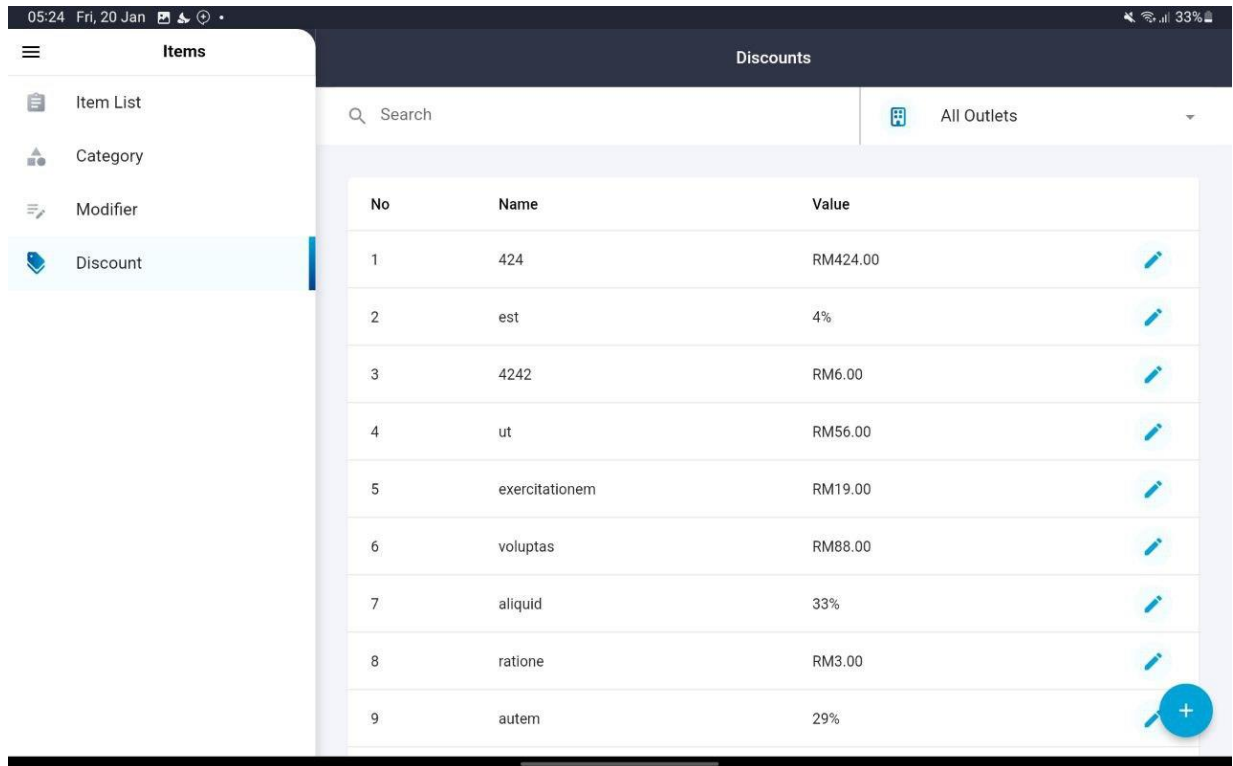


Figure 35 List Discounts UI

Figure 35 List Discounts UI shows the list of discounts. This interface shows the list of discounts that can be applied by the cashier when making a sale. The discount will be either a percentage or fixed amount.

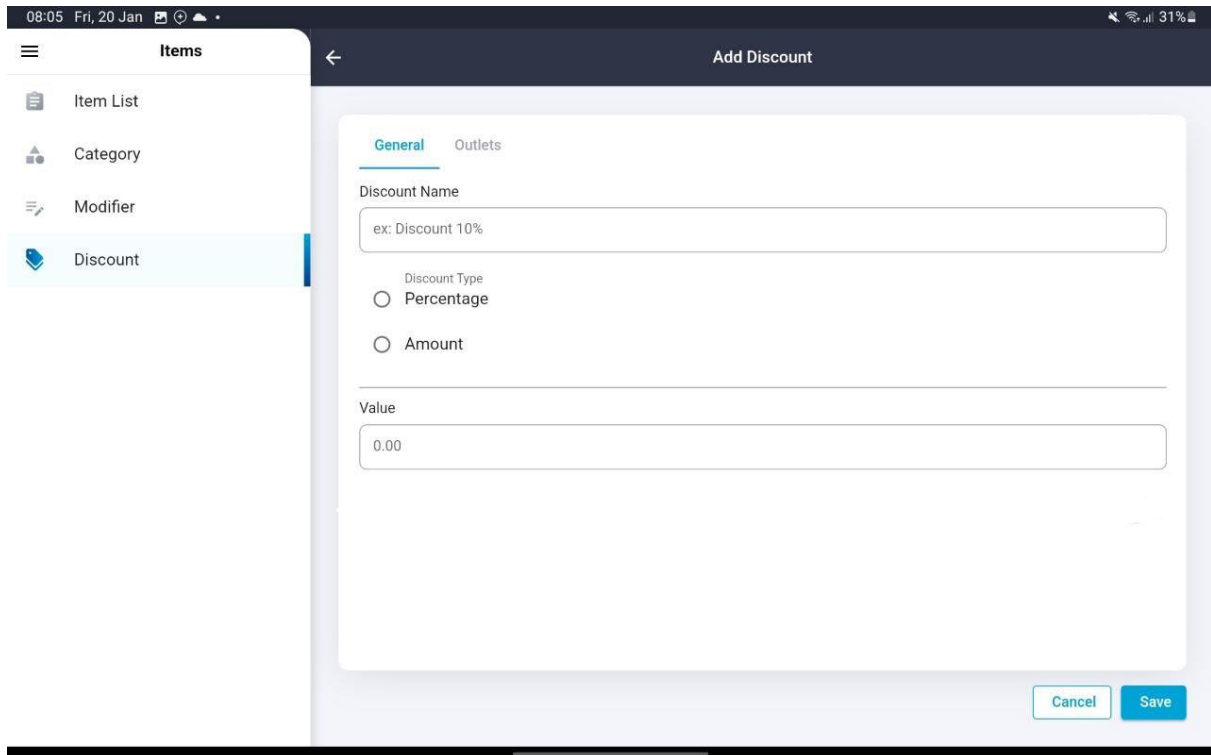


Figure 36 Add Discount UI

Figure 36 Add Discount UI shows the add discount interface. The cashier needs to enter the discount name, percentage or amount, and the value of the discount.

3.5 Data Design

3.5.1.1 Entity Relationship Diagram (ERD)

The ERD shows all the tables with column names and data types along with the relationships between the tables according to Laravel relationships. Please refer **Appendix A** for the ERD.

3.5.1.2 Data Dictionary

Adhering to the naming conventions and relationships outlined in Laravel's rules for database design is crucial for utilizing Laravel's built-in features and tools effectively. The conventions have been established to promote organization, readability, and consistency within the codebase. By following these conventions, we are making it easier for everyone to understand and maintain the code (“Laravel Naming Conventions - Web dev etc - my software development blog,” 2022).

Additionally, following the conventions can reduce the likelihood of bugs and errors, as they help to ensure that the database is structured in a way that is compatible with Laravel's built-in features and tools. These tools are designed to work seamlessly with the conventions, so by adhering to them, we will be able to take full advantage of Laravel's capabilities and make the most out of the development efforts. Please refer **Appendix B** for the data dictionary.

3.6 Proof of Initial Concept

A proof of concept for a POS (Point-of-Sale) system with table layout management would involve a prototype or a simulation of the system to demonstrate its functionality and usability. The following steps could be taken to create a proof of concept for the system:

1. Define the requirements for the system: Understand the specific needs and requirements of the restaurant or business that will be using the system. This includes determining features and functionality that are required, such as table layout management, sales management and reporting capabilities.
2. Develop a prototype: Use a software development tool to create a working prototype of the system. This prototype should include all the features and functionality specified in the requirement.

3. Test the prototype: Use test cases to evaluate and ensure that it meets the requirements and is easy to manage. This can include testing the table layout management, inventory management and reporting capabilities.
4. Evaluate the results: Analyze the results of the testing and use them to identify any issues or areas for improvement in the prototype. Make any necessary changes and retest the system until it meets all the requirements.
5. Present the proof of concept: Once the prototype is finalized and meets all of the requirements, it can be presented to stakeholders as proof of concept for the POS system with the features of table layout.

3.7 Validation and Testing Plan

A validation and testing plan for the MYSZTECH POS System would two types of testing plans which are User Acceptance Test and Usability Test (“12 Usability Testing Templates/Examples [+Checklist] | Hotjar,” 2022).

3.7.1 User Acceptance Test

User acceptance test will be performed to ensure that the system meets the end-users' requirements and is usable for them. The questions will cover the overall system satisfaction, table layout satisfaction and the functionality of the system.

Table 3.13 Functionality Feedback Form

Module	Activities	Status		Comment
		Yes	No	
Login	Login using correct credentials			
Sales Module	Add items to sales			
	Edit items in sale			
	Remove items from sale			

	Add discount to the sale			
	Remove discount from sale			
	Add taxes to the sale			
	Remove taxes from sale			
	Charge the customer			
	Store the sale in predefined order			
	Create the order using the table layout			
Table Layout Module	View correct table orders			
	Edit the table layout			
	Assign name to the table			
	Assign predefined order to the table			
	Delete table			
	Open order from the table			
	Add section			
	Edit section name			

Manage Items	View all items			
	Add items			
	Edit items			
	Delete Items			
Manage Categories	View all categories			
	Add categories			
	Edit categories			
	Delete categories			
Manage Discounts	View all discounts			
	Add discounts			
	Edit discounts			
	Delete discounts			
Manage Taxes	View all taxes			
	Add taxes			
	Edit taxes			
	Delete taxes			
Manage Outlets	View all outlets			
	Add outlets			
	Edit outlets			
	Delete outlets			

3.7.2 Usability Feedback Form

Table 14 Usability Feedback Form

No	Activities	Strongly Disagree				Strongly Agree
		1	2	3	4	5
1.	The system works effectively for me.					
2.	The system is user friendly (easy to use and to interact with).					
3.	The system has high validation and clearly provides a guide to fulfill the requirement.					
4.	The interface of the system is consistent and attractive.					
5.	The system provides a clear information (e.g., menu information, order information)					
6.	The system has great information organization.					
7.	The system fulfills all the requirements as a basic POS system.					
8.	How likely are you to recommend the MTS POS System to other businesses in your industry?					
9.	From 1 to 5 what is your rating to this POS system?					

3.7.3 Table Layout Feedback Form

Table 3.15 Table Layout Feedback Form

No	Activities	Strongly Disagree				Strongly Agree
		1	2	3	4	5
6.	How intuitive and user-friendly is the table layout in terms of navigating and managing tables?					
7.	How visually appealing and aesthetically pleasing is the table layout design					
8.	How responsive is the table layout in terms of updating and reflecting real-time changes?					
9.	How satisfied are you with the level of customization and flexibility offered by the table layout in terms of arranging and organizing tables based on your specific business needs?					

3.8 Potential Used of Proposed Solution

The MYSZTECH POS System has a number of potential uses in the restaurant and food service industry. Some of the key uses of the proposed solution include:

1. Efficiency: The POS system allows for fast and efficient processing of orders and payments, which can help to improve the overall efficiency of the restaurant.
2. Integration: The MYSZTECH POS System can be integrated with other software or hardware such as payment gateways, accounting software, and kitchen printers, which can streamline operations and improve overall efficiency.
3. Streamlined Table-Based Services: For businesses offering table-based services, the system's table layout management module facilitates efficient table assignment, tracking

occupancy, and managing orders. This ensures a seamless flow of orders and enhances the overall dining experience.

4. **Discount and Promotion Management:** The MTS POS System provides businesses with the ability to apply discounts and manage promotions effectively. This empowers businesses to attract customers, increase sales, and implement targeted marketing strategies.
5. **Tax Compliance:** The MTS POS System includes a tax management module that assists businesses in ensuring compliance with tax regulations. It simplifies the calculation and application of taxes, generates tax reports, and facilitates seamless tax submissions.

3.9 Gantt Chart

No	Item	Durations (Weeks)	Implementation																	
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	
1	Development Documentation Preparation	2	█	█																
2	Mockup Preparation	2			█	█														
3	Environment and project setup	2				█	█													
4	QR Ordering UI development	2					█	█												
5	Manage Sales for Cashier UI development	2						█	█											
6	Manage Items for Cashier UI development	1								█										
7	Manage Outlets for Cashier UI development	1									█									
8	Manage Categories for Cashier UI development	1										█								
9	Manage Discounts for Cashier UI development	1											█							
10	Manage Taxes for Cashier UI development	1												█						
11	Basic functions API development	2											█	█						
12	QR Ordering API development	2												█	█					
13	Manage Sales for Cashier API development	1													█					
14	Manage Items for Cashier API development	1														█				
15	Manage Outlets for Cashier API development	1															█			
16	Manage Categories for Cashier API development	1																█		
17	Manage Discounts for Cashier API development	1																	█	
18	Manage Taxes for Cashier API development	1																		█
19	Server installation & configuration	1																		█
20	System testing	1																		█
21	User Manual and Update Documentation	1																		█

Figure 37 Gantt Chart

CHAPTER 4

4 IMPLEMENTATION, RESULT AND DISCUSSION

4.1 Introduction

This chapter will discuss the development, implementation, and testing of MTS POS System. This mobile app is specifically designed to streamline and optimize point-of-sale operations, enabling businesses to increase efficiency, productivity, and customer satisfaction. The MTS POS mobile app has been meticulously crafted and developed utilizing top-of-the-line tools and technologies, which are the powerful Flutter Framework for the mobile application, Laravel Framework for the API endpoint, Visual Studio Code as the IDE, MySQL for the database, and Figma for the design. These tools ensure that the app is efficient, secure, and user-friendly. Moreover, the testing phase of the MTS POS mobile app has been performed with the utmost diligence and attention to detail, to identify and address any potential issues or bugs.

4.2 Implementation Process

Implementation is a process to develop the system from the requirements to the fully functional system that can be used by the end users.

4.2.1 Setup Version Control

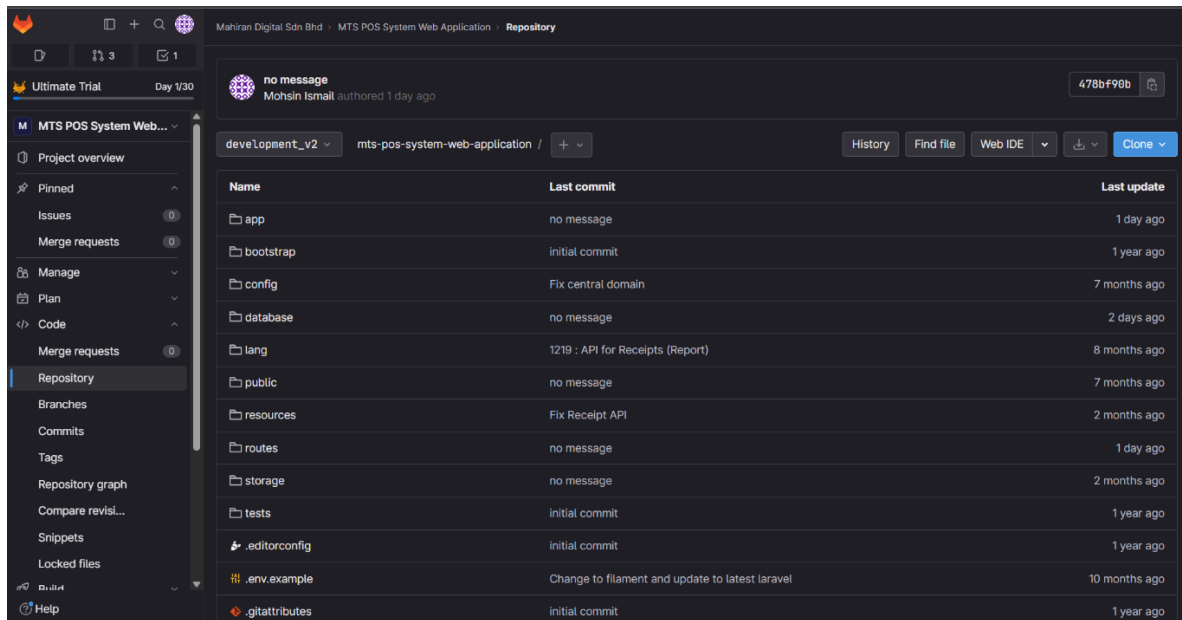


Figure 38 Gitlab Setup

Based on the Figure 38 Gitlab Setup, I have chosen GitLab as my preferred version control system. The implementation process begins by creating two repository on GitLab to store the project's source code and relevant files where one for API development using Laravel framework and the other is for mobile app development using Flutter framework . I clone this repository onto my local machine and start committing my changes. To manage different features, bug fixes, and releases, I establish threes branches named "development_v1", "staging_v1" and "production_v1". These branches allow me to work on specific features independently and merge them when ready. I leverage collaborative features such as pull requests and code reviews to ensure code quality and maintain a clean codebase. Version control with GitLab offers several benefits, including a centralized and organized history of code changes, seamless collaboration, the ability to rollback to previous versions if needed, and support for concurrent development efforts. Overall, GitLab serves as a valuable tool for enhancing productivity and facilitating effective code management throughout the development lifecycle of the MTS POS System.

4.2.2 Design Database Structure

The development of MTS POS System starts with outlining the database of the system. Based on Entity Relationship Diagram (ERD), database blueprint has been designed following the Laravel naming conventions and the correct ways in creating the database structure which is by creating the migration files in Laravel Project. Below is the migration files that has been developed and database generated in MySQL based on the migration files.

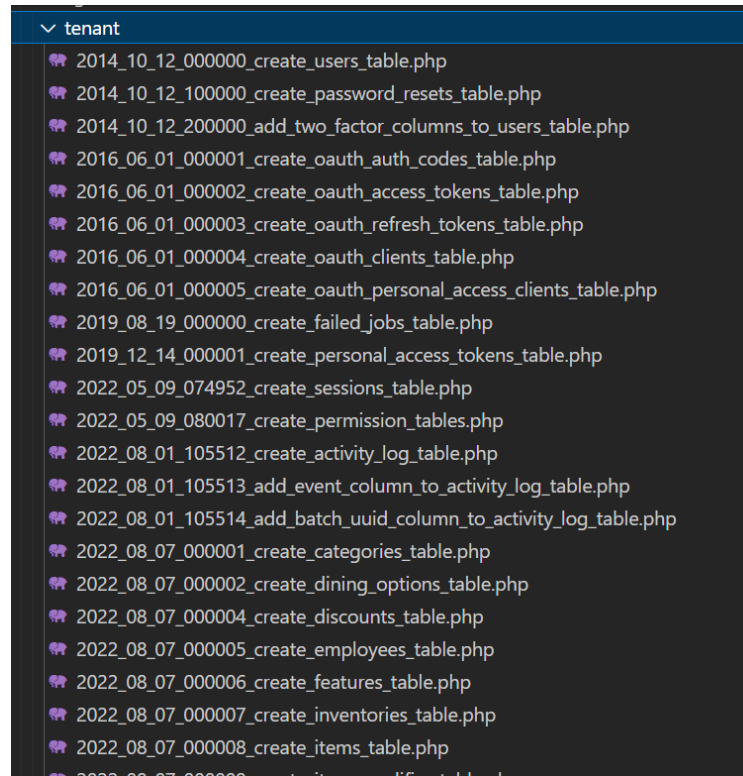
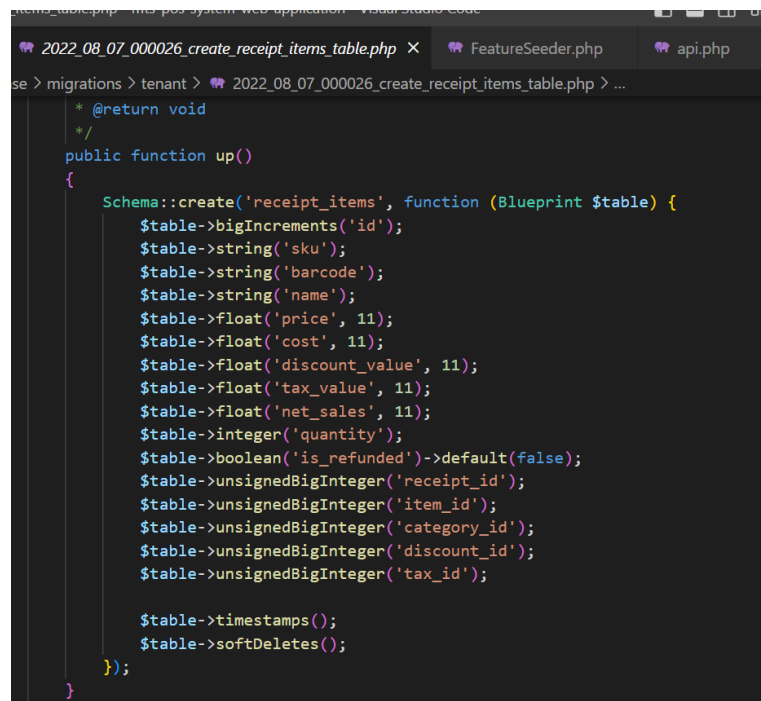


Figure 39 List of migration files

Figure 39 List of migration files shows a migration file. This migration file is a crucial component in Laravel development. It defines the database schema and allows for smooth management of database changes over time. The migration file contains a set of instructions that Laravel follows to create or modify database tables, columns, indexes, and relationships. By utilizing migrations, developers can easily version control their database structure and collaborate seamlessly. It provides a standardized and organized approach to handle database modifications, making it easier to deploy and maintain the application across different environments. Migrations also allow for efficient rollbacks, enabling developers to revert changes if needed. This powerful

feature of Laravel simplifies the process of database management and ensures consistency and reliability in the application's data structure.



```
2022_08_07_000026_create_receipt_items_table.php × FeatureSeeder.php api.php
se > migrations > tenant > 2022_08_07_000026_create_receipt_items_table.php > ...

    * @return void
    */
    public function up()
    {
        Schema::create('receipt_items', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('sku');
            $table->string('barcode');
            $table->string('name');
            $table->float('price', 11);
            $table->float('cost', 11);
            $table->float('discount_value', 11);
            $table->float('tax_value', 11);
            $table->float('net_sales', 11);
            $table->integer('quantity');
            $table->boolean('is_refunded')->default(false);
            $table->unsignedBigInteger('receipt_id');
            $table->unsignedBigInteger('item_id');
            $table->unsignedBigInteger('category_id');
            $table->unsignedBigInteger('discount_id');
            $table->unsignedBigInteger('tax_id');

            $table->timestamps();
            $table->softDeletes();
        });
    }
}
```

Figure 40 Receipt Items table migration file snippet

Figure 40 Receipt Items table migration file snippet represents a migration file's `up` method in Laravel, which is responsible for creating a database table called `receipt_items`. Here's a breakdown of the table's columns and their definitions:

1. `id`: A primary key column with auto-incrementing big integers.
2. `sku`, `barcode`, `name`: Strings representing the SKU, barcode, and name of the receipt item.
3. `price`, `cost`, `discount_value`, `tax_value`, `net_sales`: Float columns representing the price, cost, discount value, tax value, and net sales of the receipt item.
4. `quantity`: An integer column indicating the quantity of the receipt item.
5. `is_refunded`: A boolean column with a default value of false, indicating whether the item has been refunded.
6. `receipt_id`, `item_id`, `category_id`, `discount_id`, `tax_id`: Unsigned big integer columns representing foreign keys referencing other related tables.

7. ``timestamps()``: Automatically manages the `created_at` and `updated_at` columns for tracking the item's creation and last update timestamps.
8. ``softDeletes()``: Enables soft deletion, adding a `deleted_at` column to the table to track when an item is deleted (the actual row is not removed from the database, but marked as deleted).

When this migration is run, it will create the ``receipt_items`` table with the specified columns and the appropriate indexing.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	sku	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	barcode	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
5	price	double(11,2)			No	None			Change Drop More
6	cost	double(11,2)			No	None			Change Drop More
7	discount_value	double(11,2)			No	None			Change Drop More
8	tax_value	double(11,2)			No	None			Change Drop More
9	net_sales	double(11,2)			No	None			Change Drop More
10	quantity	int(11)			No	None			Change Drop More
11	is_refunded	tinyint(1)			No	0			Change Drop More
12	receipt_id	bigint(20)		UNSIGNED	No	None			Change Drop More
13	item_id	bigint(20)		UNSIGNED	No	None			Change Drop More

Figure 41 MTS POS System database

Figure 41 MTS POS System database shows the table `receipt_items` structure in the database.

4.2.3 Development of the API

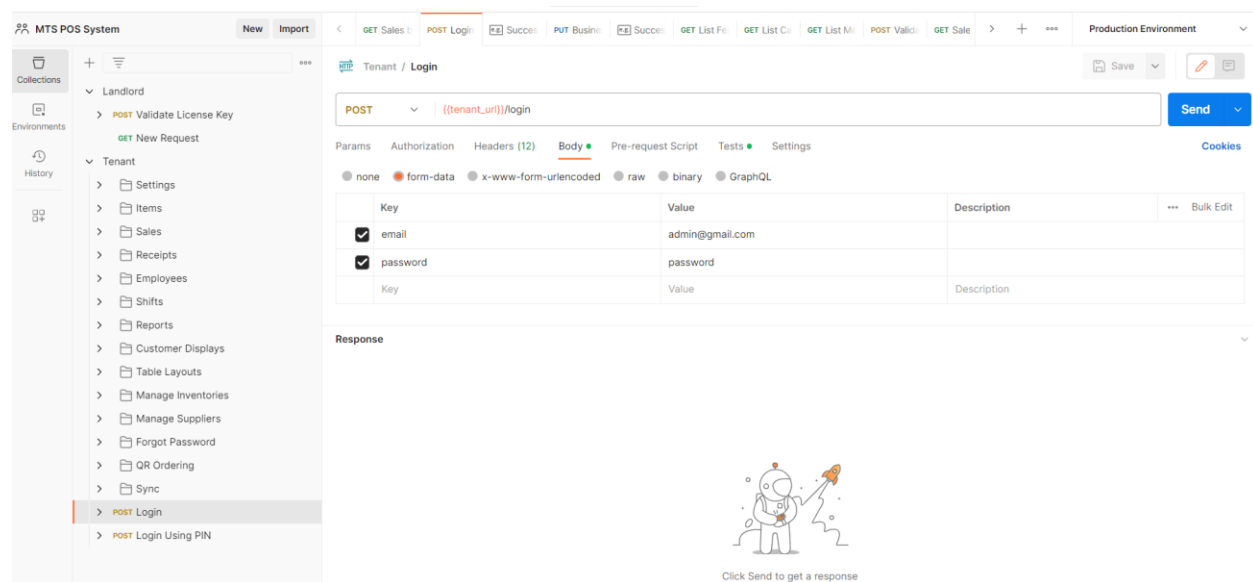


Figure 42 Postman API Testing UI

The process of API development for the MTS POS System using the Laravel framework involves several steps.

1. **Designing API Endpoints:** Start by designing the API endpoints that will be responsible for handling various functionalities of the POS system. Determine the required endpoints for tasks such as managing sales, inventory, customers, and authentication.
2. **Creating Routes:** In Laravel, define routes that correspond to each API endpoint. These routes will map the incoming requests to the appropriate controllers and methods that will handle the logic and data processing.
3. **Building Controllers:** Create controllers that will receive the requests from the API routes and contain the necessary code to fetch, manipulate, and respond with the required data. Implement the logic for validating inputs, authenticating requests, and executing the necessary business operations.
4. **Implementing Models:** Define models in Laravel to represent the various data entities in the POS system, such as sales, inventory items, and customers. These models will interact with the database and provide an abstraction layer for data operations.
5. **Validating and Sanitizing Input:** Implement input validation and sanitization mechanisms to ensure that the data received from API requests is valid and secure.

Laravel provides convenient validation and sanitization features to enforce rules and sanitize user input.

6. **Handling Authentication:** Implement authentication mechanisms to secure the API endpoints. Laravel offers various authentication methods such as token-based authentication (using Laravel Passport).
7. **Testing and Debugging:** Thoroughly test the API endpoints using tools like Laravel's built-in testing framework or third-party tools which is Postman. Test various scenarios, including success cases, edge cases, and error conditions. Debug and fix any issues or inconsistencies.
8. **Documentation:** Document the API endpoints, their expected inputs, and the responses they provide. Generate API documentation using Postman.
9. **Deployment and Monitoring:** Deploy the API to a suitable server environment and set up monitoring and logging mechanisms to track performance, errors, and usage patterns.

4.2.4 API Development Coding

The language that has been use to develop the API is PHP programming language. The API was develop using visual studio code.

A screenshot of a code editor window with a dark background and light-colored text. The code is PHP and is numbered from 1 to 20. It defines a public function named 'verify' that takes a 'Request \$request' as an argument. The function first retrieves a user from the database based on the email provided in the request. It then checks if the user's email has already been verified. If not, it validates the OTP provided in the request. If the OTP is valid, it marks the user's email as verified and returns a successful response. If the OTP is invalid, it returns an error response. If the email is already verified, it returns a response with a translation key for 'auth.email_already_verified'.

```
1 // Verify user email
2 public function verify(Request $request)
3 {
4
5     $user = User::where('email', $request->email)->first();
6
7     if (!$user->hasVerifiedEmail()) {
8         $verify = Otp::validate($user->id, $request->otp);
9         if ($verify->status) {
10             // Valid
11             $user->markEmailAsVerified();
12             return $this->return_api(true, Response::HTTP_OK, null, null, null);
13         } else {
14             // Invalid
15             return $this->return_api(false, Response::HTTP_UNPROCESSABLE_ENTITY, null, null, null);
16         }
17     }
18
19     return $this->return_api(false, Response::HTTP_OK, trans('auth.email_already_verified'), null, null);
20 }
```

Figure 43 Auth Controller–Verify User Email

This function is responsible for verifying the user's email address. It takes the user's email and OTP (One-Time Password) as inputs. First, it retrieves the user based on the provided email. If the user's email is not already verified, it validates the OTP entered by the user. If the OTP is valid, the function marks the user's email as verified and returns a successful response. If the OTP is invalid, it returns an error response.


```

1 // Login
2 public function login(Request $request)
3 {
4     // Validate email and password
5     $validator = Validator::make($request->all(), [
6         'email' => 'required|string',
7         'password' => 'required|string',
8     ]);
9
10    // Return errors if validation failed
11    if ($validator->fails()) {
12        return $this->return_api(false, Response::HTTP_UNPROCESSABLE_ENTITY, null, null, $validator->errors());
13    }
14
15    // If login successful
16    if (Auth::attempt($validator->validated(), true)) {
17        // Get user using auth
18        $user = Auth::user();
19        // If user email already verified
20        if ($user->hasVerifiedEmail()) {
21            // Create access token
22            $user->accessToken = $user->createToken('authToken')->accessToken;
23            // Return response with user resource model
24            return $this->return_api(true, Response::HTTP_OK, null, new UserResource($user), null);
25        }
26        $userTemp = new User();
27        $userTemp->email = $user->email;
28        // Email not verified
29        return $this->return_api(false, Response::HTTP_FORBIDDEN, trans("auth.email_not_verified"), $userTemp, null);
30    }
31    return $this->return_api(false, Response::HTTP_UNAUTHORIZED, trans("auth.failed"), null, null);
32 }

```

Figure 44 Auth Controller-Login

The login function handles user authentication. It takes the user's email and password as input. First, it validates the email and password using the provided validation rules. If the validation fails, it returns an error response with the validation errors. If the login attempt is successful, it creates an access token for the user and returns a response with the authenticated user's details in the UserResource model. If the user's email is not verified, it returns a response indicating that the email is not verified.

```

1 public function logout()
2 {
3     $user = Auth::user()->token();
4
5     $user->revoke();
6     return $this->return_api(true, Response::HTTP_OK, null, null, null);
7 }

```

Figure 45 Auth Controller-Logout

This function logs out the authenticated user by revoking the user's access token. It retrieves the authenticated user's token and revokes it. It returns a successful response upon successful logout.

```

1 // Resend email verification
2 public function resend(Request $request)
3 {
4     $user = User::where('username', $request->username)->first();
5     if ($user->hasVerifiedEmail()) {
6         return $this->return_api(false, Response::HTTP_BAD_REQUEST, trans("auth.email_already_verified"), null, null);
7     }
8
9     try {
10        $this->buildEmailVerificationMessage($user);
11        return $this->return_api(true, Response::HTTP_OK, trans("auth.email_send"), null, null);
12    } catch (Exception $e) {
13        return $this->return_api(false, Response::HTTP_INTERNAL_SERVER_ERROR, $e->getMessage(), null, null);
14    }
15 }

```

Figure 46 Auth Controller-Resend Verification Email

This function allows the user to request a resend of the email verification OTP. It takes the user's username as input. It retrieves the user based on the provided username. If the user's email is already verified, it returns an error response. If the email is not verified, it generates a new email verification code and sends it to the user's email address. It returns a successful response upon successful email sending and an error response if there is an issue with the email sending process.

```

1 // Send Email Verification Code
2 protected function buildEmailVerificationMessage($user)
3 {
4     // Generate Code for current user
5     $otp = Otp::generate($user->id);
6
7     if ($otp->status) {
8         $data = ['title' => trans("auth.verify_title"), 'email' => $user->email, 'otp' => $otp];
9         SendEmailVerificationCode::dispatch($data);
10    } else {
11        return $this->return_api(false, Response::HTTP_UNPROCESSABLE_ENTITY, $otp->message, null, null);
12    }
13 }

```

Figure 47 Auth Controller-Send Email Verification Code

This function is responsible for generating an email verification code and sending it to the user. It takes the user as input. First, it generates a one-time password (OTP) for the user. If the OTP generation is successful, it constructs an email message with the OTP and dispatches it for sending. If there is an error in the OTP generation process, it returns an error response.

```

1 public function store(CategoryStoreRequest $request)
2 {
3     // Get Validate Request
4     $validated = $request->validated();
5
6     return DB::transaction(function () use ($validated) {
7         // Create category
8         $category = Category::create($validated);
9
10        return $this->return_api(true, Response::HTTP_CREATED, null, null, null);
11    });
12 }
13

```

Figure 48 Category Item Controller - Store

This function allows a user to store or create a new category in a database and it ensures that the request is validated, and database changes are performed within a transaction for data integrity.

```

1 public function update(CategoryUpdateRequest $request, Category $category)
2 {
3     // Get Validate Request
4     $validated = $request->validated();
5     $category->update($validated);
6
7     return $this->return_api(true, Response::HTTP_ACCEPTED, null, null, null);
8 }
9

```

Figure 49 Category Item Controller - Update

This function allows a user to send a request to update an existing category in the database. It validates the request data, updates the category, and returns an API response indicating the success of the update operation.

```

1 public function store(DiningOptionStoreRequest $request){
2     // Get Validate Request
3     $validated = $request->validated();
4
5     return DB::transaction(function () use ($validated) {
6         // Create dining
7         $dining = DiningOption::create($validated);
8
9         foreach ($validated['outlets'] as $outlet) {
10            // Attach selected outlet to this dining
11            $dining->outlets()->syncWithoutDetaching($outlet['id']);
12        }
13
14        return $this->return_api(true, Response::HTTP_CREATED, null, null, null);
15    });
16 }
17

```

Figure 50 Dining Setting Controller - Store

This function allows a user to send a request to store a new dining option in the database, along with attaching selected outlets to the dining option. It ensures that the request is validated, and the database changes are performed within a transaction for data integrity.

```

1 public function update(DiningOptionUpdateRequest $request, DiningOption $diningOption)
2 {
3     $validated = $request->validated();
4
5     foreach ($validated['outlets'] as $outlet) {
6         $diningOption->outlets()->detach();
7     }
8
9     foreach ($validated['outlets'] as $outlet) {
10        // Attach selected outlet to this tax
11        $diningOption->outlets()->syncWithoutDetaching($outlet['id']);
12    }
13
14    $diningOption->update($validated);
15
16    return $this->return_api(true, Response::HTTP_ACCEPTED, null, null, null);
17 }
18

```

Figure 51 Dining Setting Controller - Update

This function allows a user to send a request to update an existing dining option in the database. It validates the request data, updates the dining option (including modifying associated outlets), and returns an API response indicating the success of the update operation.

```

1 public function store(DiscountStoreRequest $request)
2 {
3     // Get Validate Request
4     $validated = $request->validated();
5
6     return DB::transaction(function () use ($validated) {
7         // Create discount
8         $discount = Discount::create($validated);
9
10        foreach ($validated['outlets'] as $outlet) {
11            // Attach selected outlet to this discount
12            $discount->outlets()->syncWithoutDetaching($outlet['id']);
13        }
14
15        return $this->return_api(true, Response::HTTP_CREATED, null, null, null);
16    });
17 }
18

```

Figure 52 Discount Item Controller - Store

This function allows a user to send a request to store a new discount in the database. It ensures that the request is validated, and the database changes are performed within a transaction for data integrity.

```

1 public function update(DiscountUpdateRequest $request, Discount $discount)
2 {
3     $validated = $request->validated();
4
5     foreach ($validated['outlets'] as $outlet) {
6         // Detach selected outlet to this discount
7         $discount->outlets()->detach();
8     }
9     foreach ($validated['outlets'] as $outlet) {
10        // Reattach selected outlet to this discount
11        $discount->outlets()->syncWithoutDetaching($outlet['id']);
12    }
13
14    $discount->update($validated);
15
16    return $this->return_api(true, Response::HTTP_ACCEPTED, null, null, null);
17 }
18

```

Figure 53 Discount Item Controller - Update

This function allows a user to send a request to update an existing discount in the database. It validates the request data, updates the discount (including modifying associated outlets), and returns an API response indicating the success of the update operation.

```

1 public function store(PredefinedOrderStoreRequest $request)
2 {
3     // Get Validate Request
4     $validated = $request->validated();
5     $validated['outlet_id'] = $validated['outlet']['id'];
6
7     // Create predefined_order
8     PredefinedOrder::create($validated);
9
10    return $this->return_api(true, Response::HTTP_CREATED, null, null, null);
11 }
12

```

Figure 54 Order Setting Controller- Store

This function allows a user to send a request to store a new predefined order in the database. It validates the request data, creates the predefined order, and returns an API response indicating the success of the creation operation.

```

1 public function update(PredefinedOrderUpdateRequest $request, PredefinedOrder $openOrder)
2 {
3     $validated = $request->validated();
4
5     $validated['outlet_id'] = $validated['outlet']['id'];
6     $openOrder->update($validated);
7
8     return $this->return_api(true, Response::HTTP_ACCEPTED, null, null, null);
9 }
10

```

Figure 55 Order Setting Controller - Update

This function allows a user to send a request to update an existing predefined order in the database. It validates the request data, updates the predefined order, and returns an API response indicating the success of the update operation

```

1 public function sendReceipt(Request $request, Receipt $receipt)
2 {
3     // Get the authenticated user's email
4     //$email = $request->user()->email;
5     //$user = User::where('email', $email)->first();
6
7     // Find receipt id
8     $receipt = Receipt::findOrFail($receipt->id);
9
10    // Get cashier based on shift id
11    $shift = Shift::findOrFail($receipt->shift_id);
12    $cashier = User::findOrFail($shift->user_id);
13
14    // Get all receipt item based on receipt id
15    $receiptItems = ReceiptItem::where('receipt_id', $receipt->id)->get();
16
17    // Send email to recipient
18    Mail::to($request->user()->email)->send(new SendReceipt($receipt, $receiptItems, $cashier));
19
20    return $this->return_api(true, Response::HTTP_OK, null, null, null);
21 }
22

```

Figure 56 Receipt Controller – Send Receipt

This function allows a user to send a receipt via email. It retrieves the necessary information related to the receipt, such as the receipt items and cashier details, and sends an email containing the receipt details to the user who made the request.

```
1 public function store(SaleStoreRequest $request)
2 {
3     $this->authorize('create', Sale::class);
4
5     $validated = $request->validated();
6
7     $sale = Sale::create($validated);
8
9     return new SaleResource($sale);
10 }
11
```

Figure 57 Sale Controller - Store

This function allows a user to send a request to create a new sale. It performs an authorization check to ensure the user has the necessary permissions, validates the request data, creates a new sale record in the database, and returns the created sale data in the desired format.

4.2.5 Mobile App Development Coding

The language that has been used to develop the mobile app is Dart programming language using visual studio code.

```
1 class ItemBloc {
2   //sync
3   Future<ItemsSyncResponseModel> syncItemList() async {
4     return await Webservice.get(ItemsSyncResource.getItemList());
5   }
6
7   // Get list of item
8   Future<ItemListResponseModel> getListItems(
9     {ItemFilterModel? itemFilterModel}) async {
10    return await Webservice.get(
11      ItemResource.getItemsList(itemFilterModel ?? ItemFilterModel());
12    }
13
14    // Store Item and tab
15    Future<ItemStoreResponseModel> storeItem(ItemModel itemModel) async {
16      return await Webservice.post(ItemResource.storeItem(itemModel));
17    }
18
19    // Update Item and tab
20    Future<ItemStoreResponseModel> updateItem(ItemModel itemModel) async {
21      return await Webservice.put(ItemResource.updateItem(itemModel));
22    }
23
24    // Delete Item and tab
25    Future<ItemStoreResponseModel> deleteItem(int id) async {
26      return await Webservice.delete(ItemResource.deleteItem(id));
27    }
28 }
```

Figure 58 ItemBloc

1. `syncItemList()`: This asynchronous function is responsible for syncing the list of items with a remote server. It uses the `Webservice.get()` method to make a `GET` request to the server's `ItemsSyncResource.getItemList()` API endpoint. The function returns a `Future` that resolves to an `ItemsSyncResponseModel`, representing the response received from the server.
2. `getListItems({ItemFilterModel? itemFilterModel})`: This function retrieves a list of items from the server. It accepts an optional `itemFilterModel` parameter, which can be used to apply filters to the list of items. The function uses the `Webservice.get()` method

to make a `GET` request to the server's `ItemResource.getItemsList()` API endpoint, passing the `itemFilterModel` as a parameter. It returns a `Future` that resolves to an `ItemListResponseModel`, representing the response containing the list of items.

3. `storeItem(ItemModel itemModel)`: This function is responsible for storing a new item on the server. It takes an `itemModel` parameter, which represents the item to be stored. The function uses the `Webservice.post()` method to make a `POST` request to the server's `ItemResource.storeItem()` API endpoint, passing the `itemModel` as the request payload. It returns a `Future` that resolves to an `ItemStoreResponseModel`, representing the response received from the server.
4. `updateItem(ItemModel itemModel)`: This function updates an existing item on the server. It takes an `itemModel` parameter, which represents the updated item data. The function uses the `Webservice.put()` method to make a `PUT` request to the server's `ItemResource.updateItem()` API endpoint, passing the `itemModel` as the request payload. It returns a `Future` that resolves to an `ItemStoreResponseModel`, representing the response received from the server.
5. `deleteItem(int id)`: This function deletes an item with the specified ID from the server. It takes an `id` parameter, representing the ID of the item to be deleted. The function uses the `Webservice.delete()` method to make a `DELETE` request to the server's `ItemResource.deleteItem()` API endpoint, passing the `id` as a parameter. It returns a `Future` that resolves to an `ItemStoreResponseModel`, representing the response received from the server.

```

1 class LoginFormBloc extends FormBuilder<String, String> {
2   final UserBloc userBloc = UserBloc();
3
4   final email = TextFieldBloc(
5     initialValue: "admin@gmail.com",
6     validators: [
7       InputValidator.required,
8       InputValidator.emailChar,
9     ],
10  );
11  final password = TextFieldBloc(
12    initialValue: "password",
13    validators: [
14      InputValidator.required,
15    ],
16  );
17
18  LoginFormBloc() {
19    addFieldBlocs(fieldBlocs: [
20      email,
21      password,
22    ]);
23  }
24
25  @override
26  Future<void> onSubmitting() async {
27    try {
28      // Call API to Login
29      UserResponseModel userResponseModel =
30        await userBloc.login(email.value.trim(), password.value);
31
32      if (userResponseModel.isSuccess &&
33          userResponseModel.data!.accessToken != null) {
34        emitSuccess(canSubmitAgain: true);
35      } else {
36        email.addFieldError(userResponseModel.message);
37        emitFailure();
38      }
39    } catch (e) {
40      emitFailure(failureResponse: e.toString());
41    }
42  }
43 }

```

Figure 59 LoginFormBloc

1. `LoginFormBloc` has two fields: `email` and `password`, both of which are instances of `TextFieldBloc`. Each field is initialized with an initial value and a list of validators to perform input validation.
2. In the constructor of `LoginFormBloc`, the `email` and `password` fields are added to the form by calling `addFieldBlocs()`.
3. The `onSubmitting()` method is overridden from the `FormBloc` class. This method is called when the form is submitted. Within this method, the code tries to log in the user by

calling the `login()` method of the `userBloc` instance. It passes the values of the `email` and `password` fields as parameters.

4. If the login is successful (indicated by `userResponseModel.isSuccess` being true and a non-null `accessToken`), the form emits a success state by calling `emitSuccess(canSubmitAgain: true)`.
5. If the login fails, the `email` field is updated with an error message obtained from `userResponseModel.message`, and the form emits a failure state by calling `emitFailure()`.
6. If an exception occurs during the login process, the form emits a failure state with the error message obtained from `e.toString()`.

Overall, this code represents a form bloc that handles the login functionality. It validates the email and password inputs, performs the login operation using the `userBloc`, and updates the form state based on the login result.

```
1 class UserModel {
2     int? id;
3     String? name;
4     String? email;
5     String? phoneNo;
6     String? icNo;
7     String? profilePhoto;
8     String? thumbnail;
9     String? accessToken;
10    String? password;
11
12    UserModel(
13        {this.id,
14        this.name,
15        this.email,
16        this.phoneNo,
17        this.icNo,
18        this.profilePhoto,
19        this.thumbnail,
20        this.accessToken,
21        this.password});
22
23    UserModel.fromJson(Map<String, dynamic> json) {
24        id = json['id'];
25        name = json['name'];
26        email = json['email'];
27        phoneNo = json['phone_no'];
28        icNo = json['ic_no'];
29        profilePhoto = json['profile_photo'];
30        thumbnail = json['thumbnail'];
31        accessToken = json['access_token'];
32        password = json['password'];
33    }
34
35    Map<String, dynamic> toJson() {
36        final Map<String, dynamic> data = <String, dynamic>{};
37        data['id'] = id;
38        data['name'] = name;
39        data['email'] = email;
40        data['phone_no'] = phoneNo;
41        data['ic_no'] = icNo;
42        data['profile_photo'] = profilePhoto;
43        data['thumbnail'] = thumbnail;
44        data['access_token'] = accessToken;
45        data['password'] = password;
46        return data;
47    }
48 }
```

Figure 60 User Model

1. The class has several properties that represent different attributes of a user, such as `id`, `name`, `email`, `phoneNo`, `icNo`, `profilePhoto`, `thumbnail`, `accessToken`, and `password`. These properties are nullable (`Type?`) to accommodate cases where the values may not be present.
2. The class has a constructor that allows initializing the properties with values. It accepts named parameters corresponding to the properties.

3. There is also a named constructor `fromJson` that takes a `Map<String, dynamic>` as input. This constructor is used to create a `UserModel` object from a JSON object by extracting the values from the map and assigning them to the corresponding properties.
4. The class includes a `toJson` method that converts the `UserModel` object to a `Map<String, dynamic>`. This method is used when serializing the object to JSON. It creates a map, assigns the property values to the corresponding keys, and returns the map.

In summary, the `UserModel` class represents a user entity and provides methods to convert the object to and from JSON format. It allows convenient handling of user data within the application.



```
1 class _LoginFormState extends State<LoginForm> {
2   bool isChecked = false;
3   late Future<String> companyName;
4
5   @override
6   void initState() {
7     companyName = getCompanyName();
8     super.initState();
9   }
10
11  @override
12  Widget build(BuildContext context) {
13    double screenWidth = MediaQuery.of(context).size.width;
14    return BlocProvider(
15      create: (context) => LoginFormBloc(),
16      child: Builder(builder: (context) {
17        final loginFormBloc = context.read<LoginFormBloc>();
18        return FormBuilder<LoginFormBloc, String, String>({
19          onSubmitting: ((context, state) {
20            FocusScope.of(context).unfocus();
21            LoadingDialog.show(context);
22          }),
23          onSuccess: (context, state) {
24            LoadingDialog.hide(context);
25            // // Check if the device type is null or not
26            // String deviceType = await SecureStorageApi.read(key: "device_type");
27
28            // if (deviceType == "") {
29            //   // Means not choose device type yet
30
31            Navigator.pushAndRemoveUntil(
32              context,
33              MaterialPageRoute(
34                builder: ((context) => const ChooseDeviceTypeScreen()),
35            ),
36            (Route<dynamic> route) => false);
```

Figure 61 Login Form UI Code Snippet

The snippet code is a Flutter login form implemented as a stateful widget. It includes form validation and submission handling. The form is wrapped in a BlocProvider to provide an instance of LoginFormBloc to handle the form logic. The UI consists of input fields for email and password, a forgot password link, and a sign-in button. Upon successful submission, a loading dialog is displayed, and the user is navigated to the ChooseDeviceTypeScreen. In case of failure, an error message is shown, and the loading dialog is hidden.

4.2.6 Development of the MTS POS System

4.2.6.1 User Interface for Cashier

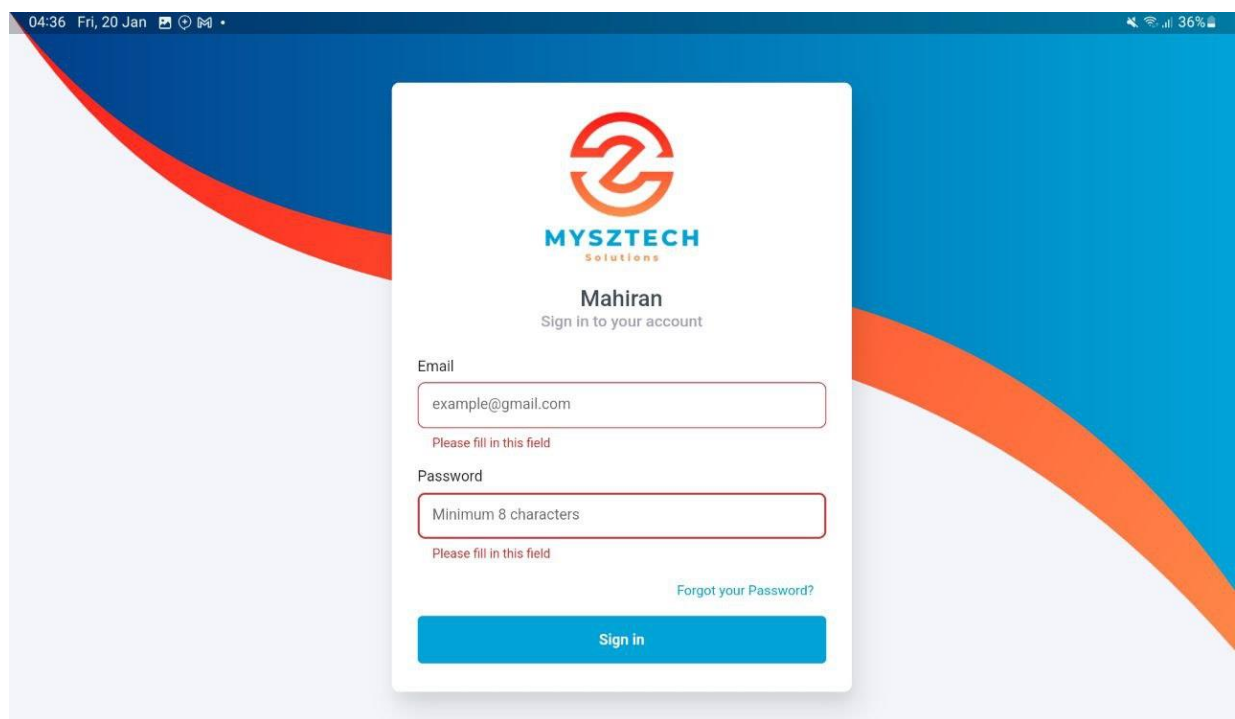


Figure 62 Login UI for Cashier

Figure 62 Login UI for Cashier shows the login interface for the cashier. The cashier need to enter their email and password to login. If they forgot password, they can press the *Forgot your Password?* button. If the cashier doesn't enter the correct details, the system will show an invalid message.

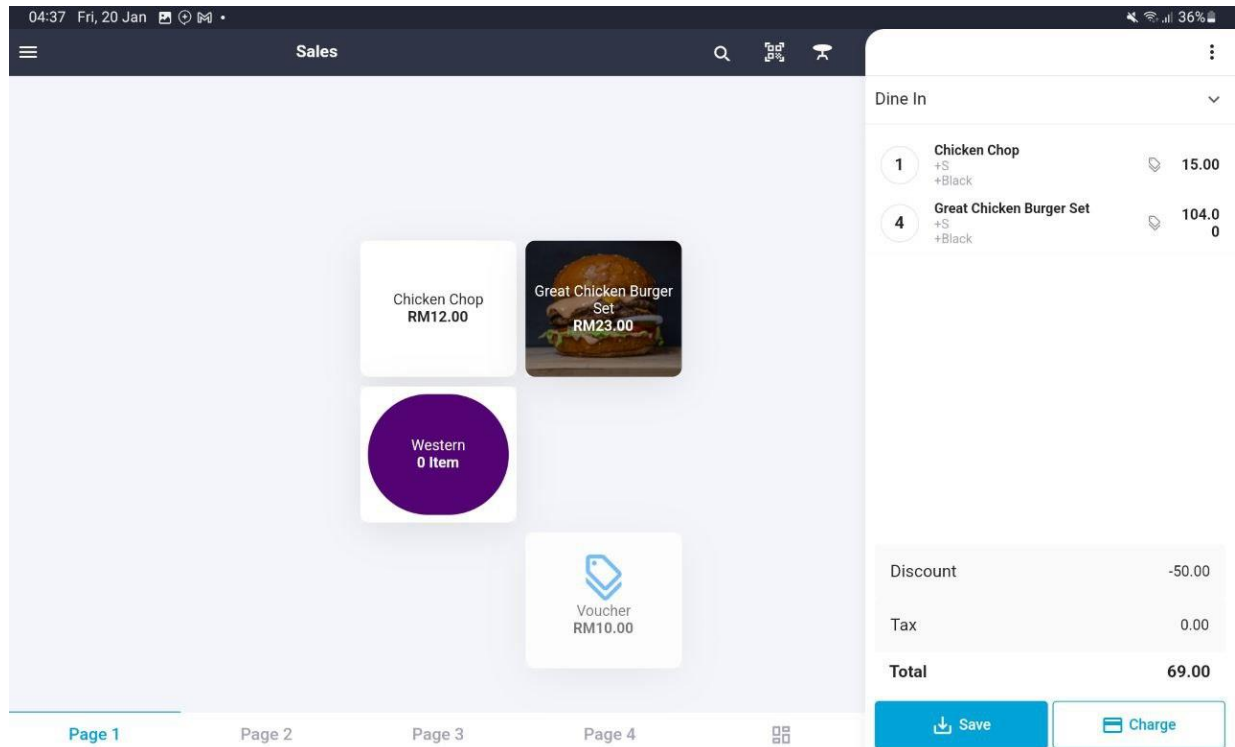


Figure 63 Main Sales UI

Figure 63 Main Sales UI shows the interface for the cashier to key in the sales. From this interface, the cashier can choose the menu requested by the customer along with the discount and tax if needed. The cashier can save the order first or directly charge the customer to make a payment. From the list at the right side, the cashier can see all the requested items by the customer and the total of the order.

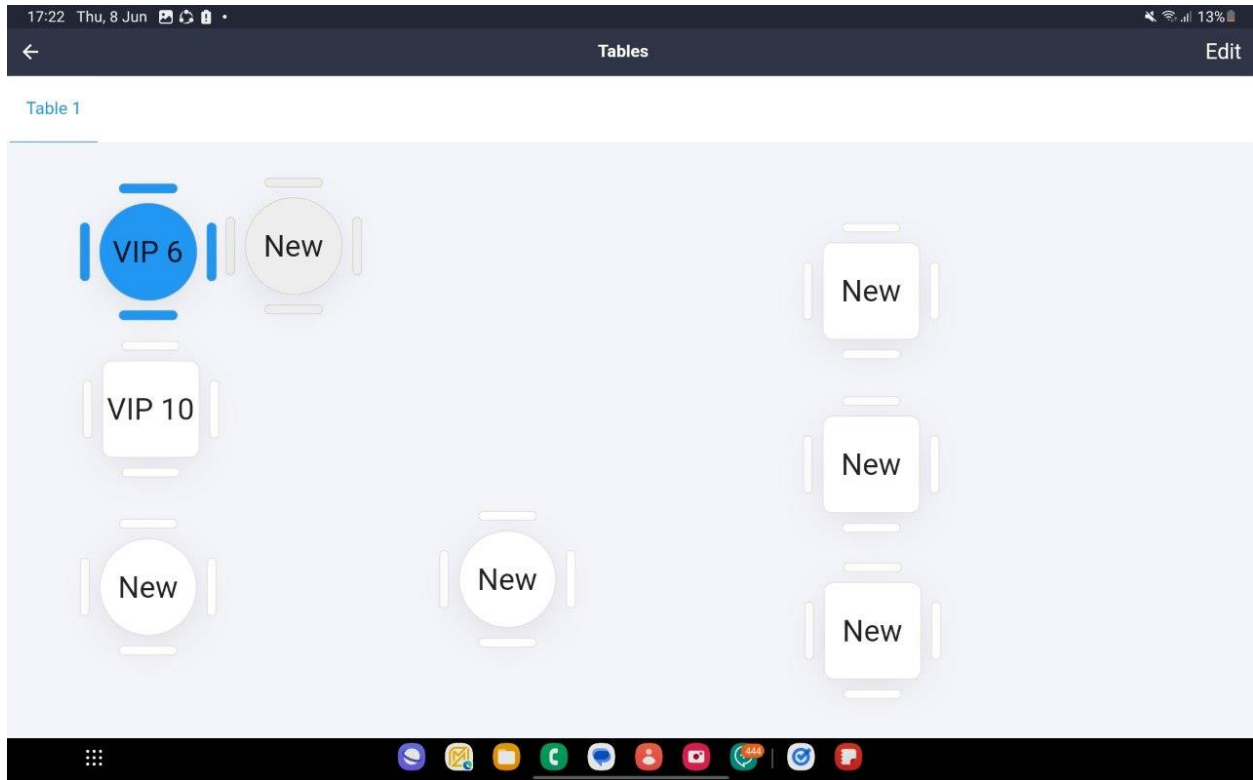


Figure 64 Table Layout UI

Figure 64 Table Layout UI shows the table layout that the cashier can use to make and order. There are an edit button on the top right. The table in blue color is the table that already has an order while, the gray table is the table that is unavailable, and the white table is the table is ready to be occupied.

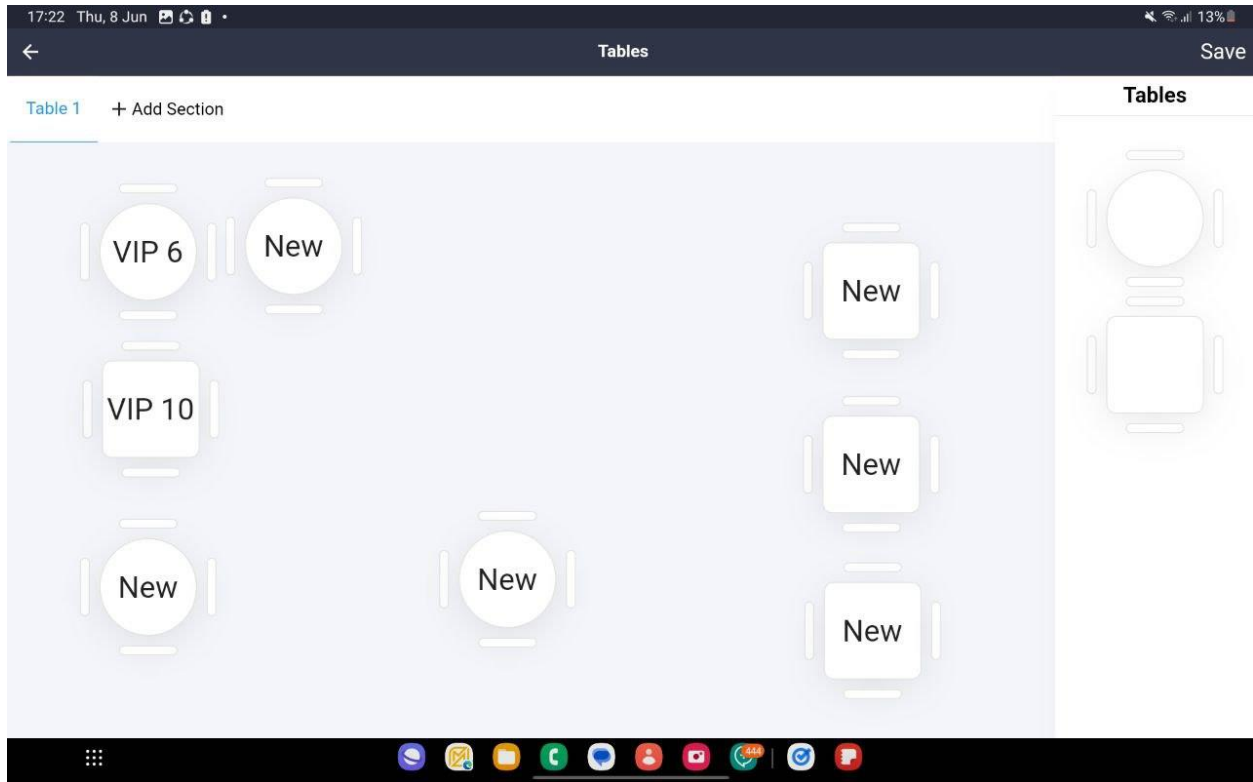


Figure 65 Edit Table Layout UI

Figure 65 Edit Table Layout UI shows the edit table layout UI where the cashier can drag and drop the tables from the side to the layout. After that, the cashier also can rename the table, assign predefined order or remove any table by press on the table. The cashier also can add more section if necessary such as Main Dining Room and VIP Dining Room by press the *Add Section* button at the above. The cashier also can the section name later. After done edit the layout, the cashier can press the *Save* button to save the layouts.

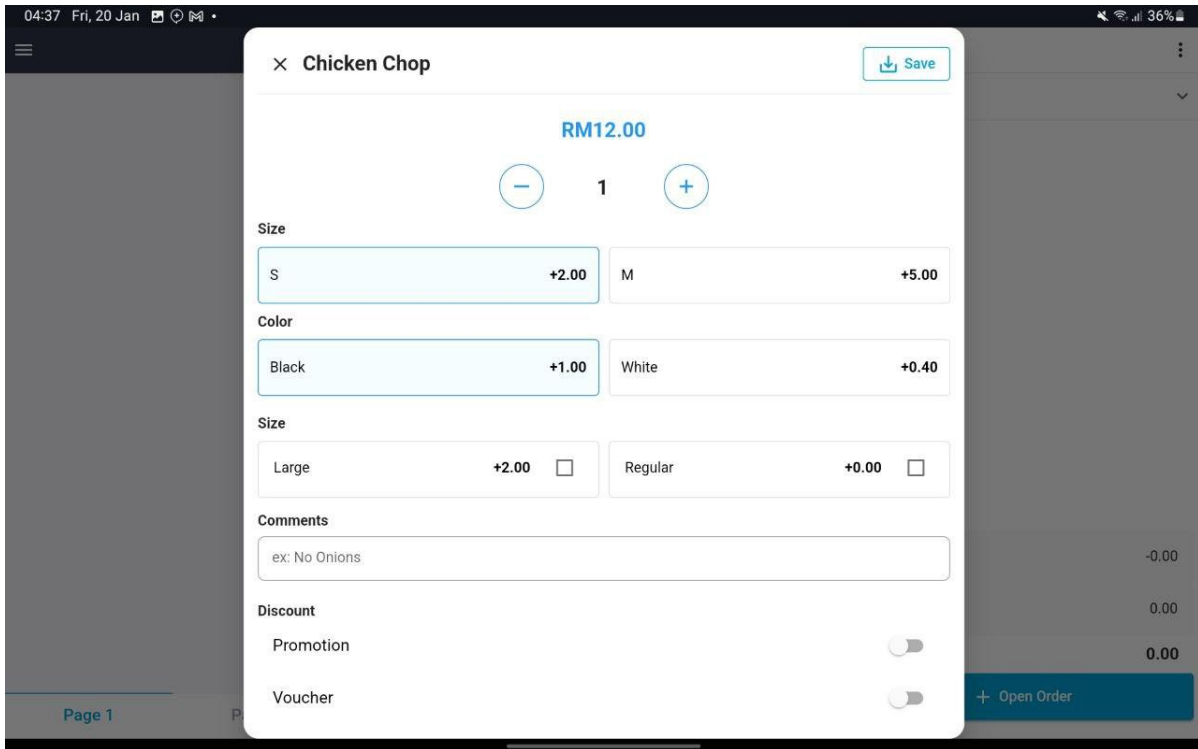


Figure 66 Item Details Popup UI

Figure 66 Item Details Popup UI shows the popup dialog that will be shown to the cashier once the cashier presses any of the menu. In this popup, the cashier needs to enter the quantity, choose the variants, choose the modifiers, enter comments, enable or disable discount and taxes. Once done key in all the details, the cashier needs to press *Save* button to add this item to the list of items at the right side.

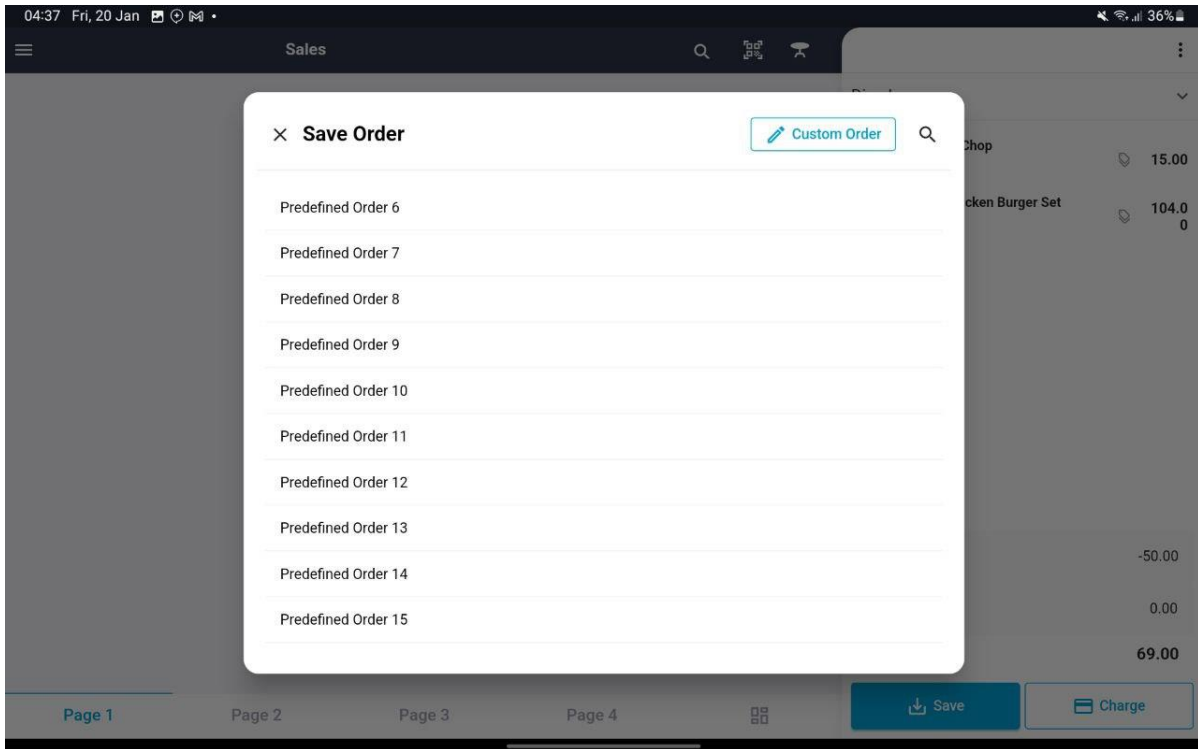


Figure 67 Popup when Press Save button UI

Figure 67 Popup when Press Save button UI shows the list of orders in the system. The cashier can choose an existing predefined order or create a new one. The predefined order can be a table or any name.

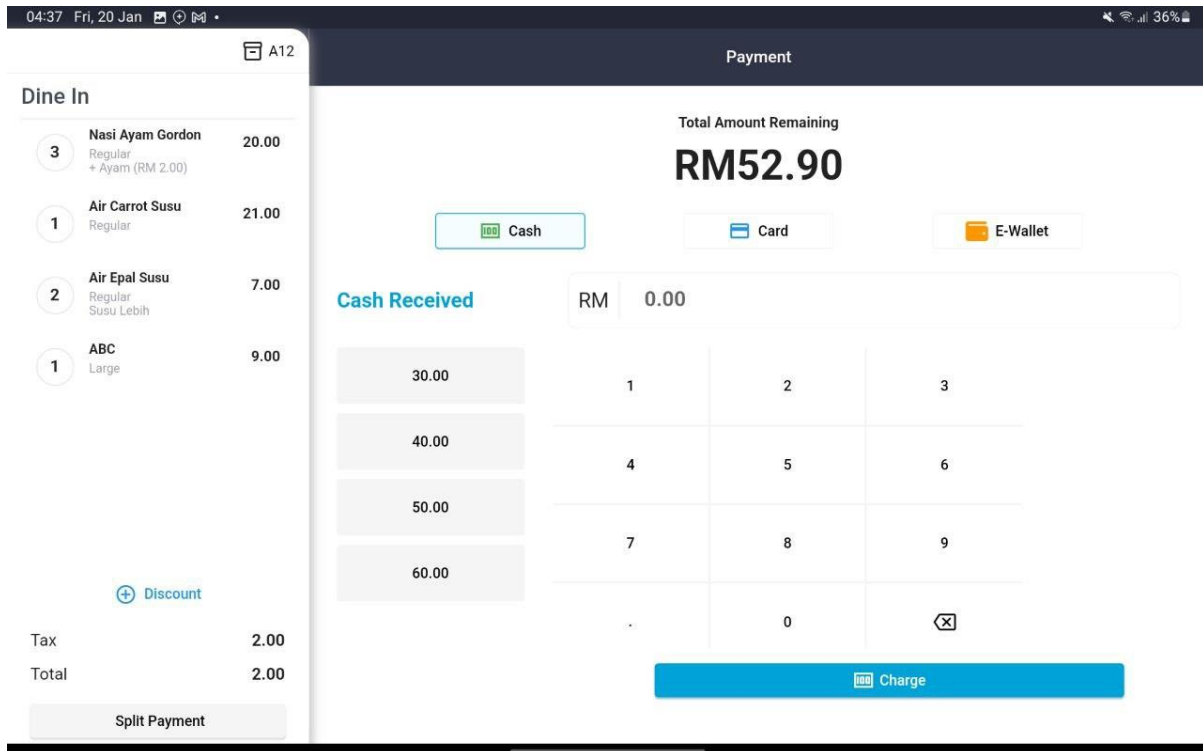


Figure 68 Order Details UI

Figure 68 Order Details UI shows the order details page. At this page, the cashier can enter the amount paid the customer either using e-wallet, cash or card. At the left side, the cashier will see the list of items for this order along with calculation and total.

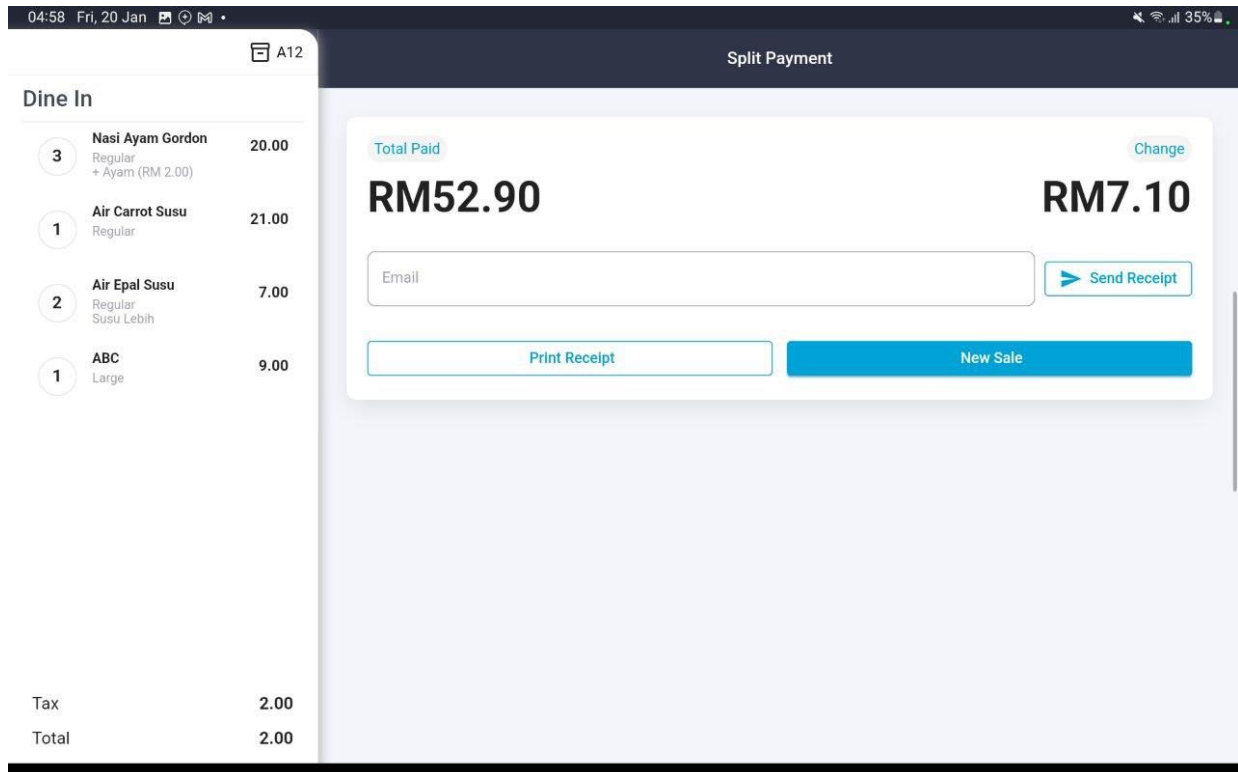


Figure 69 Payment Details UI

Figure 69 Payment Details UI shows the payment details interface along with the total paid and change need to be paid to the customer. The cashier can press print receipt to print the receipt using thermal printer or press *New Sale* button to make a new sale.

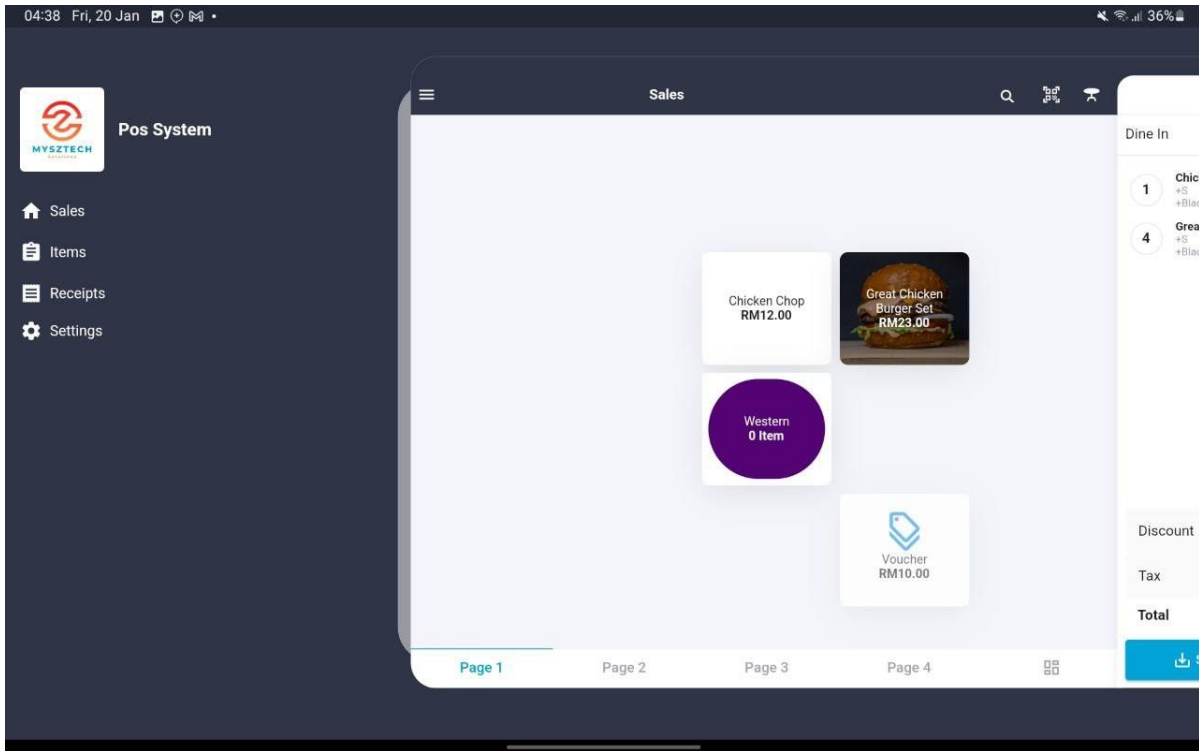


Figure 70 Menu in the Side Navigation

Figure 70 Menu in the Side Navigation shows the navigation tabs which are Sales Module, Items Module, Receipts Module and Settings Module. To access this navigation, the cashier needs to press the hamburger logo at the top left.

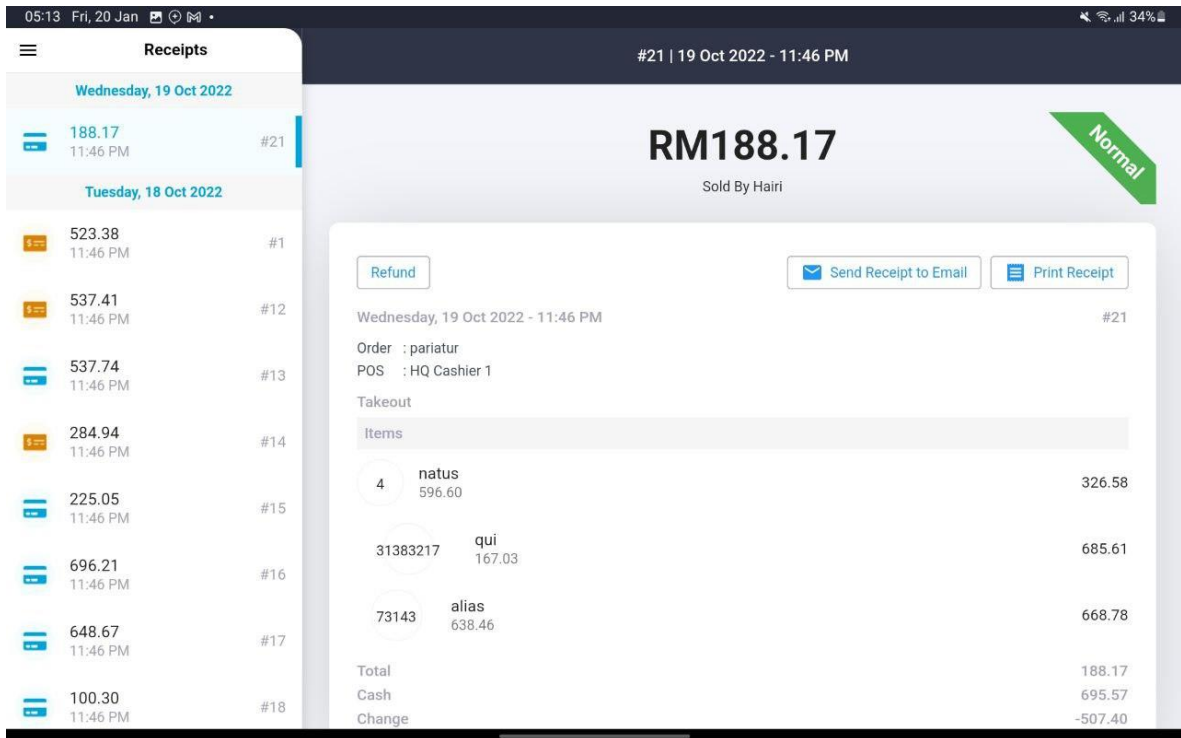


Figure 71 List of Receipt UI

Figure 71 List of Receipt UI shows the list of receipts with the details and status. At this interface, the cashier can view the list of previous receipt, the time and the details of the receipt.

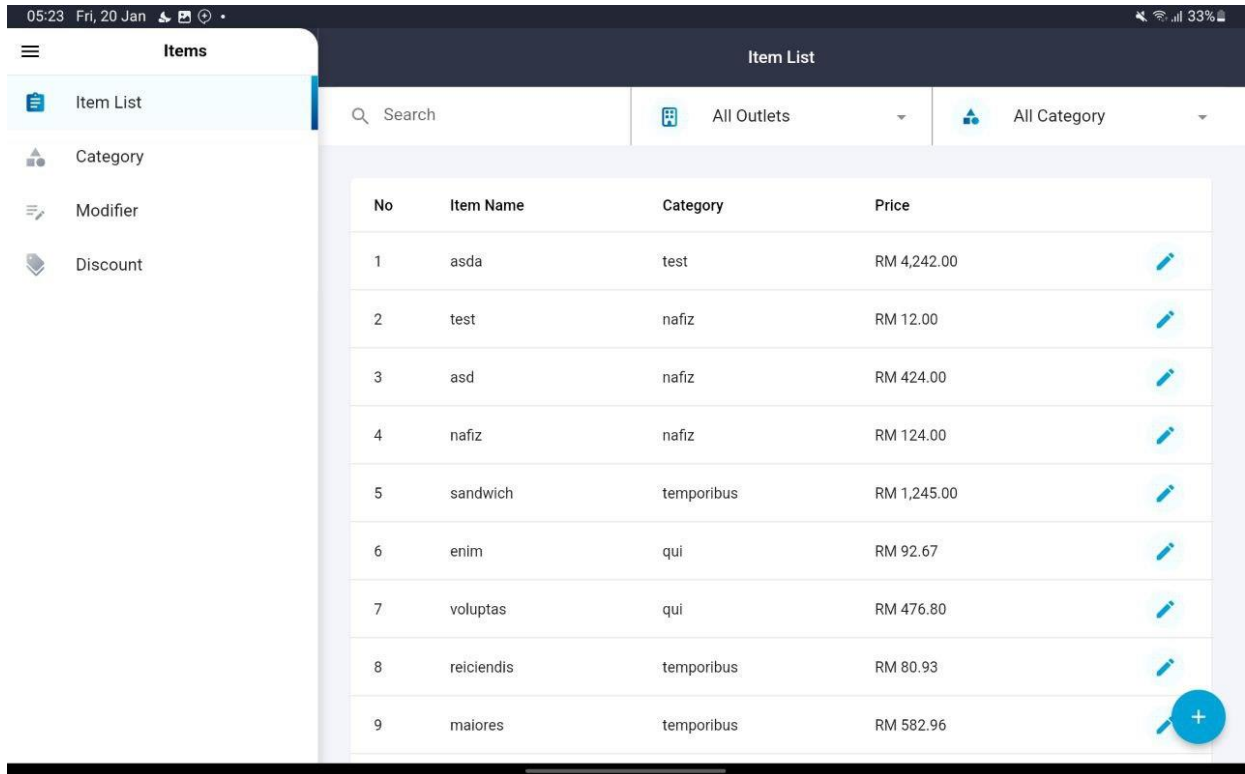


Figure 72 Item List UI

Figure 72 Item List UI shows the list of items or menu. This interface will allow the cashier to view all the items with the item name, category and price.

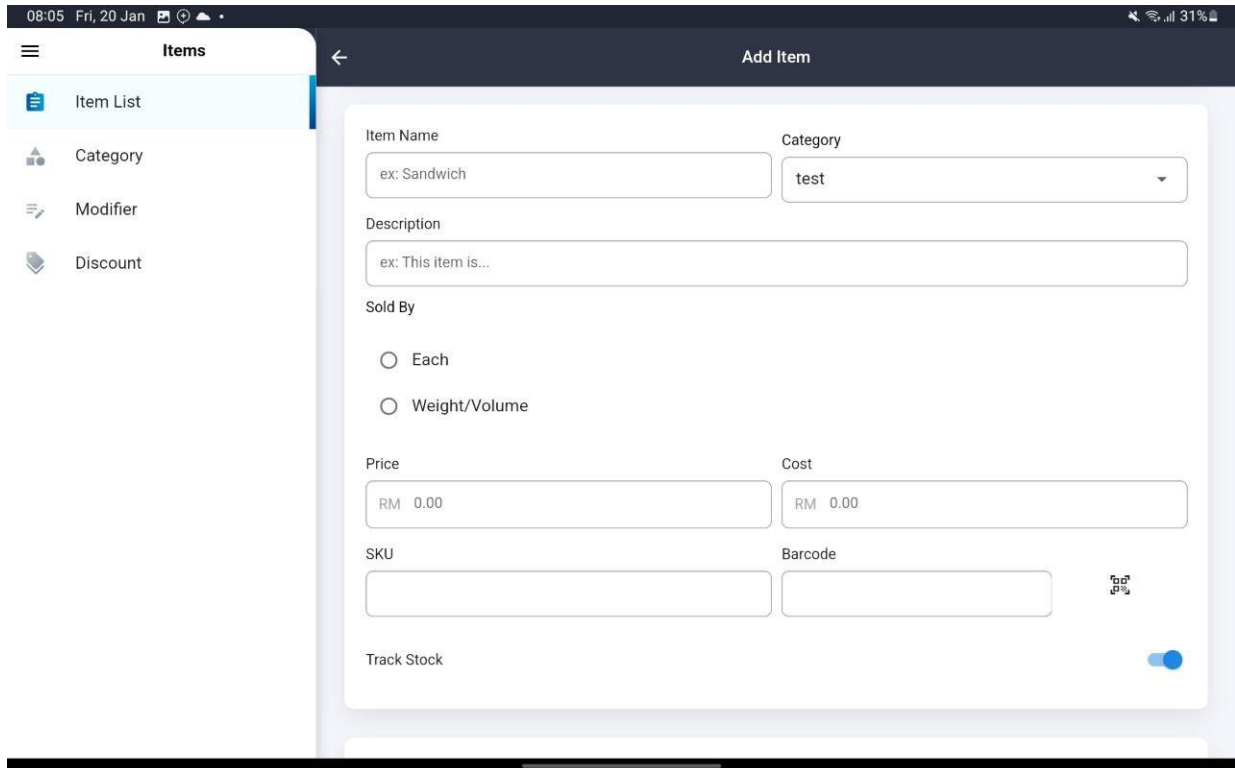


Figure 73 Add Item UI

Figure 73 Add Item UI shows the add item interface. The cashier needs to enter all the details required.

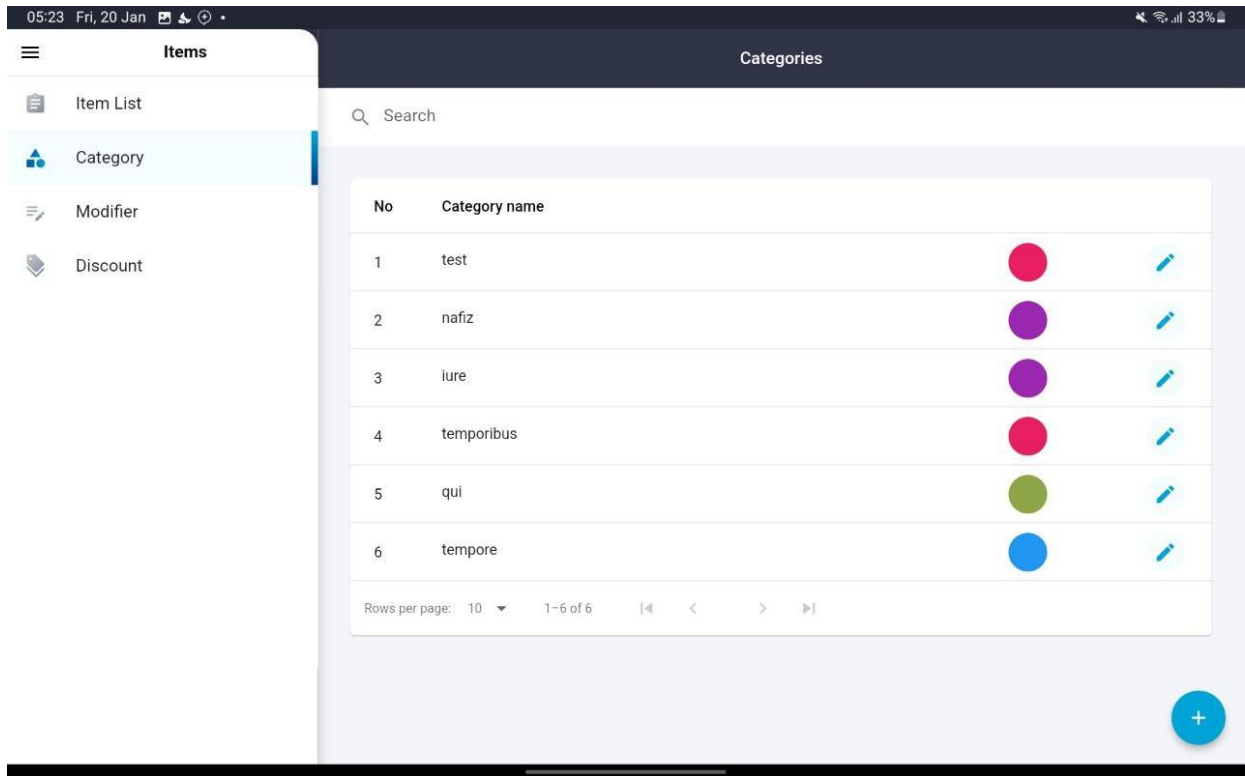


Figure 74 List Category UI

Figure 74 List Category UI shows the list of category. This interface shows the list of category that will be used to categorize all the menu.

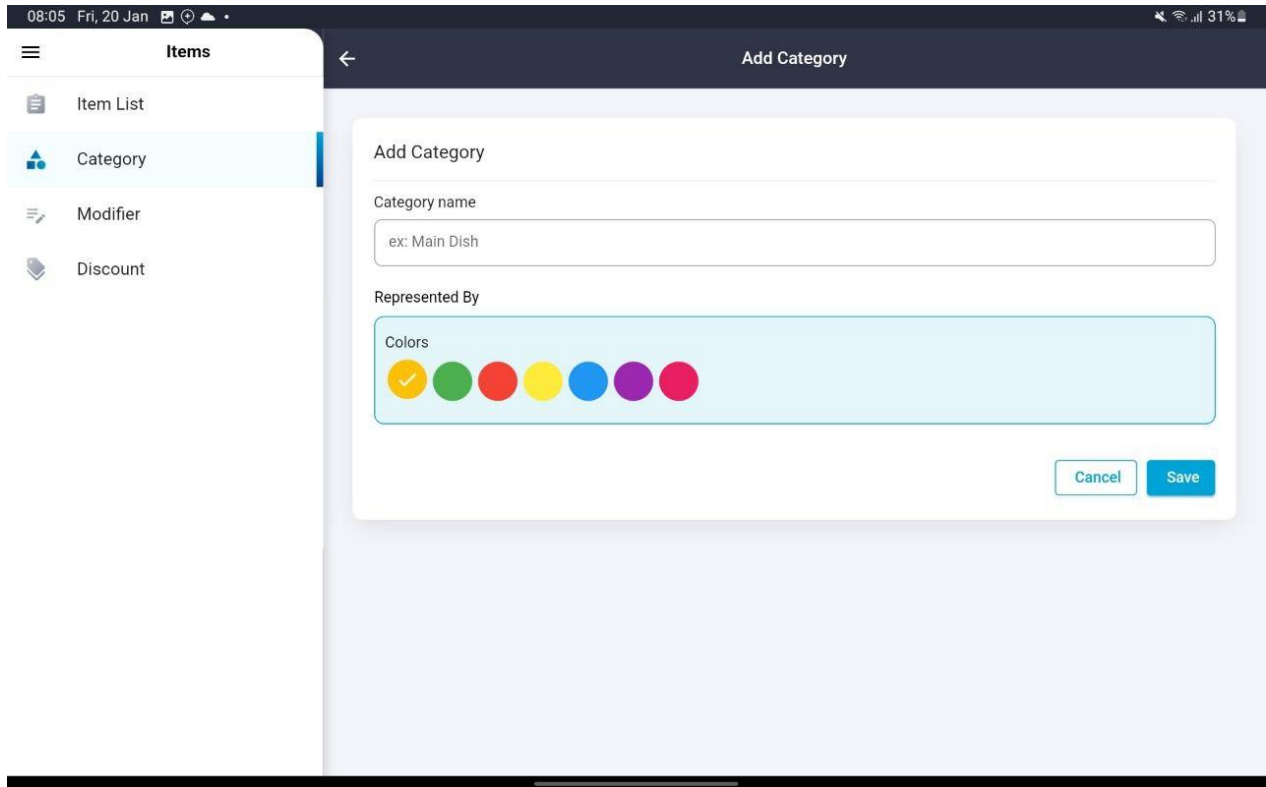


Figure 75 Add Category UI

Figure 75 Add Category UI shows the add category interface. The cashier need to enter the category name and color.

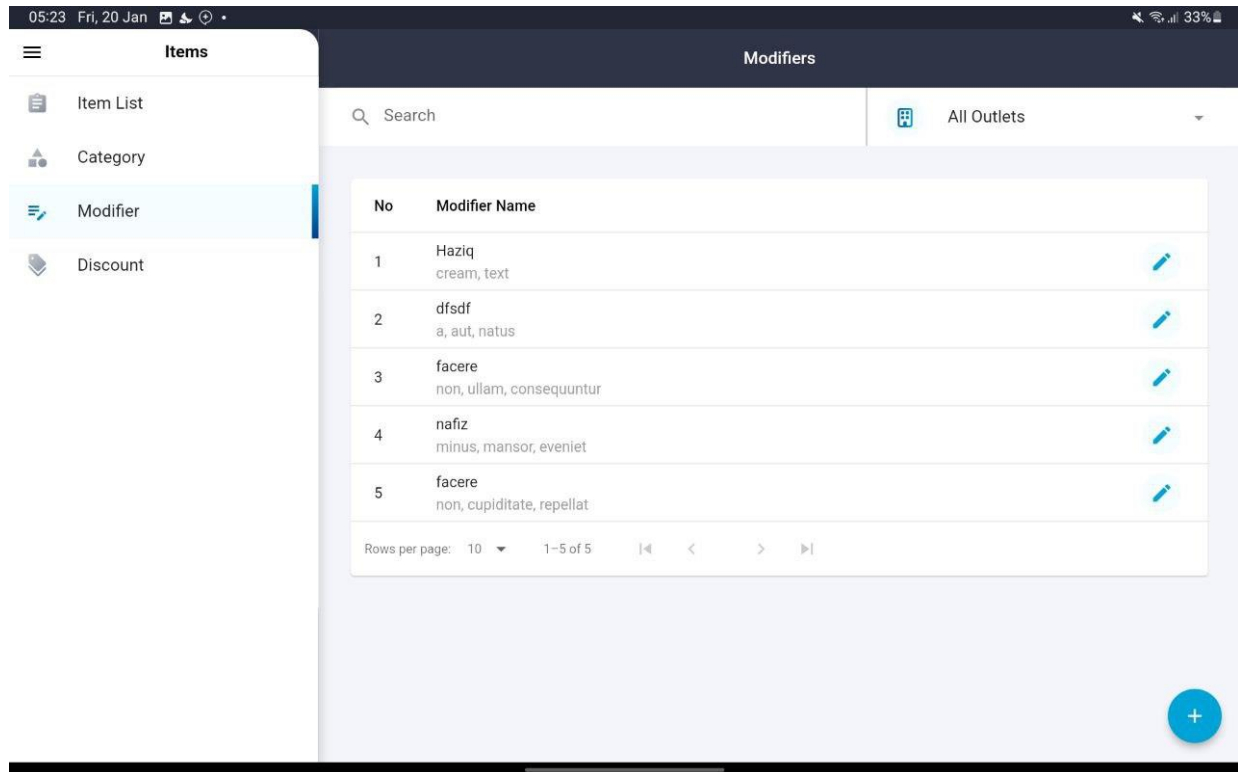


Figure 76 List Modifiers UI

Figure 76 List Modifiers UI shows the list of modifiers. It shows the name of the modifier, and the list of options. The cashier also can search for the modifiers if needed.

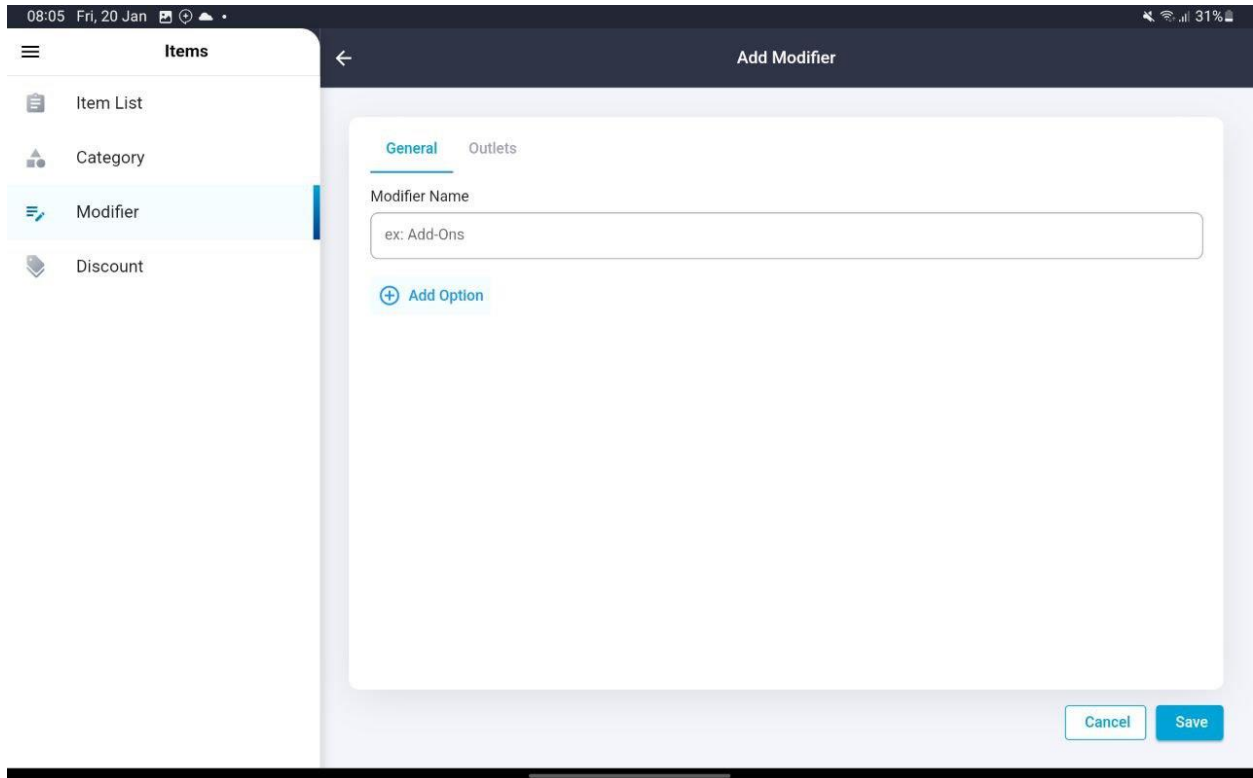


Figure 77 Add Modifier UI

Figure 77 Add Modifier UI shows the add modifier interface. The cashier needs to enter the modifier name and add an option to that modifier.

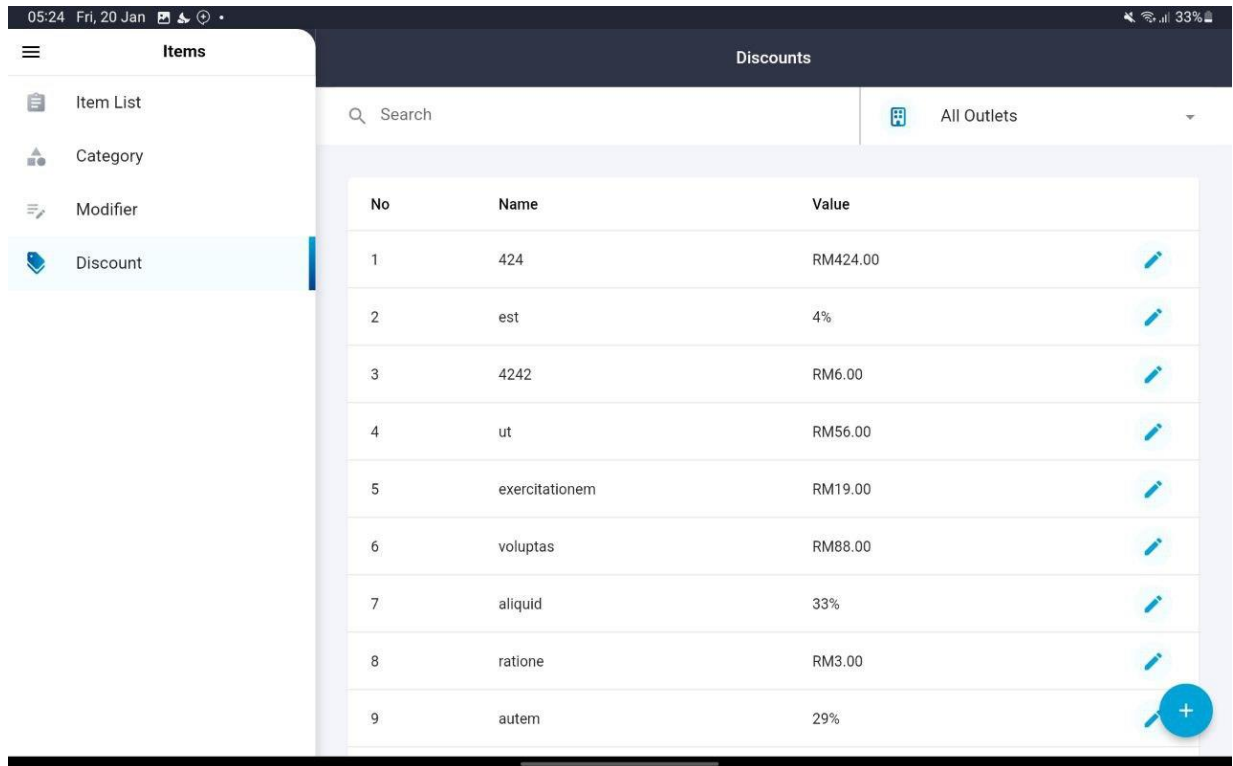


Figure 78 List Discounts UI

Figure 78 List Discounts UI shows the list of discounts. This interface shows the list of discounts that can be applied by the cashier when making a sale. The discount will be either percentage or fixed amount.

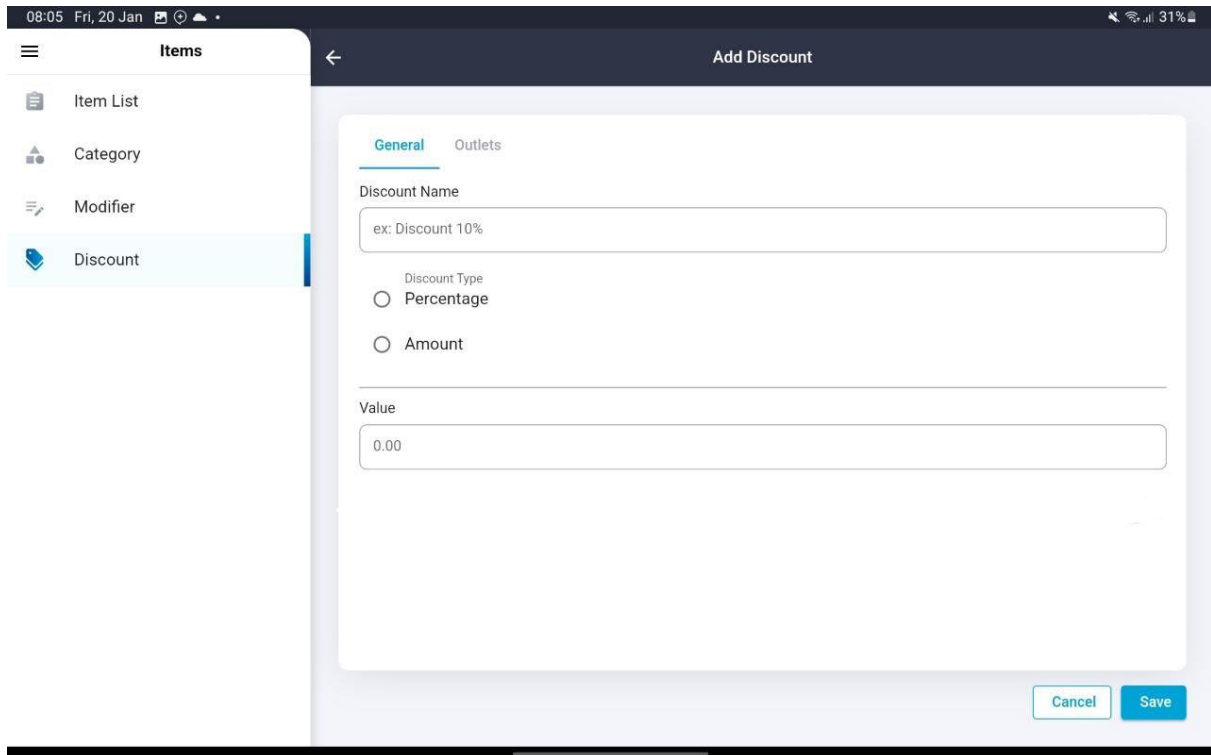


Figure 79 Add Discount UI

Figure 79 Add Discount UI shows the add discount interface. The cashier needs to enter the discount name, percentage or amount, and the value of the discount.

4.3 Testing and Result Discussion

Once the development of the MTS POS System was complete, the team moved on to the rigorous testing phase to evaluate the mobile application's functionality and usability. One crucial part of the testing process was the User Acceptance test (UAT), which involved the POS system consultant company and the cashier. During UAT, the users were given access to the mobile application and were asked to evaluate the available features. The aim of the UAT was to gather user feedback and identify any bugs or errors that needed to be addressed before the launch of the system.

Apart from UAT, a Google Form was created to assess the usability of the MTS POS System. The questionnaire in the form was designed to gather feedback from users about their satisfaction while using the system. The questions covered various aspects of usability, including ease of use, navigation, and overall satisfaction. The responses gathered from the form helped the development team to identify areas where the system needed improvement and ensure that the final product was as user-friendly as possible. The combination of UAT and Google Form feedback allowed the team to fine-tune the MTS POS System and ensure that it met the needs of businesses and their customers alike.

5 CONCLUSION

5.1 Introduction

This chapter presents a comprehensive summary of the development of the MTS POS System for the F&B industry, showcasing its successful achievement of objectives and resolution of previously identified problems. It is important to note that existing POS systems predominantly cater to the cafe industry, limiting their applicability to other types of F&B businesses. In contrast, the MTS POS System was specifically tailored to address the unique requirements of various F&B establishments, including cafes and restaurants.

One standout feature of the MTS POS System is its intuitive table layout functionality, which sets it apart from other systems. This feature enables seamless management of orders for individual tables, alleviating the challenges associated with identifying or memorizing customer table assignments. By streamlining the order management process, this feature significantly enhances operational efficiency and improves customer service.

While some existing POS systems offer table layout features, they often come at a high cost for businesses adopting them. In contrast, the MTS POS System aims to deliver industry-level features while remaining cost-friendly for small businesses in Malaysia. This affordability factor makes it an attractive and accessible choice for F&B establishments of all sizes, empowering them to streamline operations without incurring significant financial burdens.

The MTS POS System was developed using the Flutter framework in conjunction with Visual Studio Code, following the Waterfall methodology. To assess the system's effectiveness and functionality, the software was evaluated by the client who is the POS system consultant and provided valuable feedback. The evaluation process yielded positive feedback from these owners, further validating the system's efficiency.

Overall, the development of the MTS POS System showcases its ability to cater to the specific needs of the F&B industry while offering a cost-effective solution for businesses. By incorporating industry-leading features and receiving positive feedback during the evaluation process, the MTS POS System demonstrates its value as a comprehensive and user-friendly solution for F&B establishments in Malaysia.

5.2 Discussion on User Acceptance

After the development phase of the MTS POS System was completed, an implementation and evaluation process were conducted to assess the system's functionality, usability, and effectiveness. To test the system's functionalities, a User Acceptance Test (UAT) was carried out using Google Form, where users were tasked with testing the main features of the system, including sales module, items module, receipts module, and table layout to ensure their proper functioning. The detailed results of the UAT can be found in **APPENDIX C**.

The results of the UAT for the MTS POS System were highly positive, indicating a successful and well-functioning system. The client, MYSZTECH SOLUTION, and three additional cashiers participated in the testing process and provided valuable feedback.

During the UAT, the client and cashiers thoroughly tested the various modules of the POS system, including table layout, order management, items management, and hardware integration. They reported that all modules operated smoothly, without any major glitches or errors. The system effectively managed table reservations, allowed for easy order customization, and accurately processed payments..

Additionally, the UAT participants found the user interface to be intuitive and user-friendly. They were able to navigate through the system effortlessly, perform tasks efficiently, and quickly adapt to the functionalities. The system's responsiveness and performance were commendable, ensuring smooth and seamless operations during peak hours.

Based on the feedback and responses obtained, it can be concluded that the MTS POS System successfully met the client's business requirements. The UAT results provided confidence in the system's reliability and demonstrated its capability to streamline operations, enhance efficiency, and improve the overall customer experience. The positive UAT outcomes laid a solid foundation for the implementation of the system within the client's business, ensuring its readiness for full-scale deployment.

5.3 Limitation and Constraint

As mentioned in Chapter 3, this project encountered several constraints that impacted its development and implementation. In this chapter, we will delve into the limitations and constraints faced in detail, highlighting the challenges and their implications on the project's outcome.

5.3.1 Cost

The decision to utilize the Laravel Framework for system development necessitated hosting the application on a VPS SSD server, as shared hosting lacked the required terminal access. This architectural choice increased the overall development cost significantly. Furthermore, the integration of suitable hardware components, such as thermal printers and cash drawers, also incurred additional expenses.

Throughout the project, careful consideration was given to managing the costs associated with hosting and hardware integration. We analyzed different hosting providers and hardware options to strike a balance between functionality, compatibility, and affordability. By carefully evaluating available choices and negotiating competitive pricing, we were able to mitigate the impact of the cost constraint on the overall project budget.

5.3.2 Timeline

The project involved working with both the Laravel framework and Flutter, requiring a substantial investment of time to gain proficiency in these technologies. As a result, the allocated development timeline posed a limitation in achieving the desired system performance within the given timeframe.

Despite the challenges posed by the timeline constraint, efforts were made to optimize the development process. We focused on prioritizing essential features and functionalities, streamlining the development workflow, and employing agile methodologies to ensure efficient progress. Although some compromises had to be made in terms of the desired performance targets, we aimed to strike a balance between functionality and timely delivery.

5.3.3 Hardware Integration

The project involved working with both the Laravel framework and Flutter, requiring a substantial investment of time to gain proficiency in these technologies. As a result, the allocated development timeline posed a limitation in achieving the desired system performance within the given timeframe.

Despite the challenges posed by the timeline constraint, efforts were made to optimize the development process. We focused on prioritizing essential features and functionalities, streamlining the development workflow, and employing waterfall methodology to ensure efficient progress. Although some compromises had to be made in terms of the desired performance targets, we aimed to strike a balance between functionality and timely delivery.

5.4 Future Work

There are several features that can be added or improved for the future of MTS POS System for the F&B industry:

5.4.1 Generating Invoices for Clients

In order to cater to the needs of clients who require invoices instead of receipts, an enhanced feature can be implemented in the future of the MTS POS System for the F&B industry. This feature would allow users to generate detailed invoices that include relevant billing information such as client details, itemized purchases, taxes, and payment terms. By providing this functionality, the POS system would be able to meet the specific requirements of clients who prefer invoices for their financial record-keeping purposes, enhancing the system's versatility and accommodating a wider range of customer preferences.

5.4.2 Credit Card Machines and Online Payment Gateways

To improve payment convenience and flexibility, integrating the MTS POS System with credit card machines and online payment gateway services would be a valuable addition. This integration would enable seamless processing of credit card payments directly through the POS system, eliminating the need for manual input or separate payment terminals. Additionally, by integrating with popular online payment gateway services, such as PayPal or Stripe, the POS

system would empower businesses to accept digital payments securely and efficiently, enhancing the overall customer experience and streamlining the payment process.

5.4.3 Zoom In/ Out Capability for Table Layout

To enhance the user experience and improve usability, the inclusion of a zoom in/out feature for the table layout within the MTS POS System would be beneficial. This feature would allow users to adjust the scale of the table layout interface, providing the flexibility to view a comprehensive overview of the floor plan or zoom in for a more detailed view of individual tables. By incorporating this capability, users can easily manage table assignments, track occupied and available tables, and efficiently allocate resources during busy periods. The zoom in/out feature adds a layer of convenience and customization to the table management functionality, enhancing the system's usability and adaptability to various operational scenarios.

5.4.4 Enhance Reporting Module

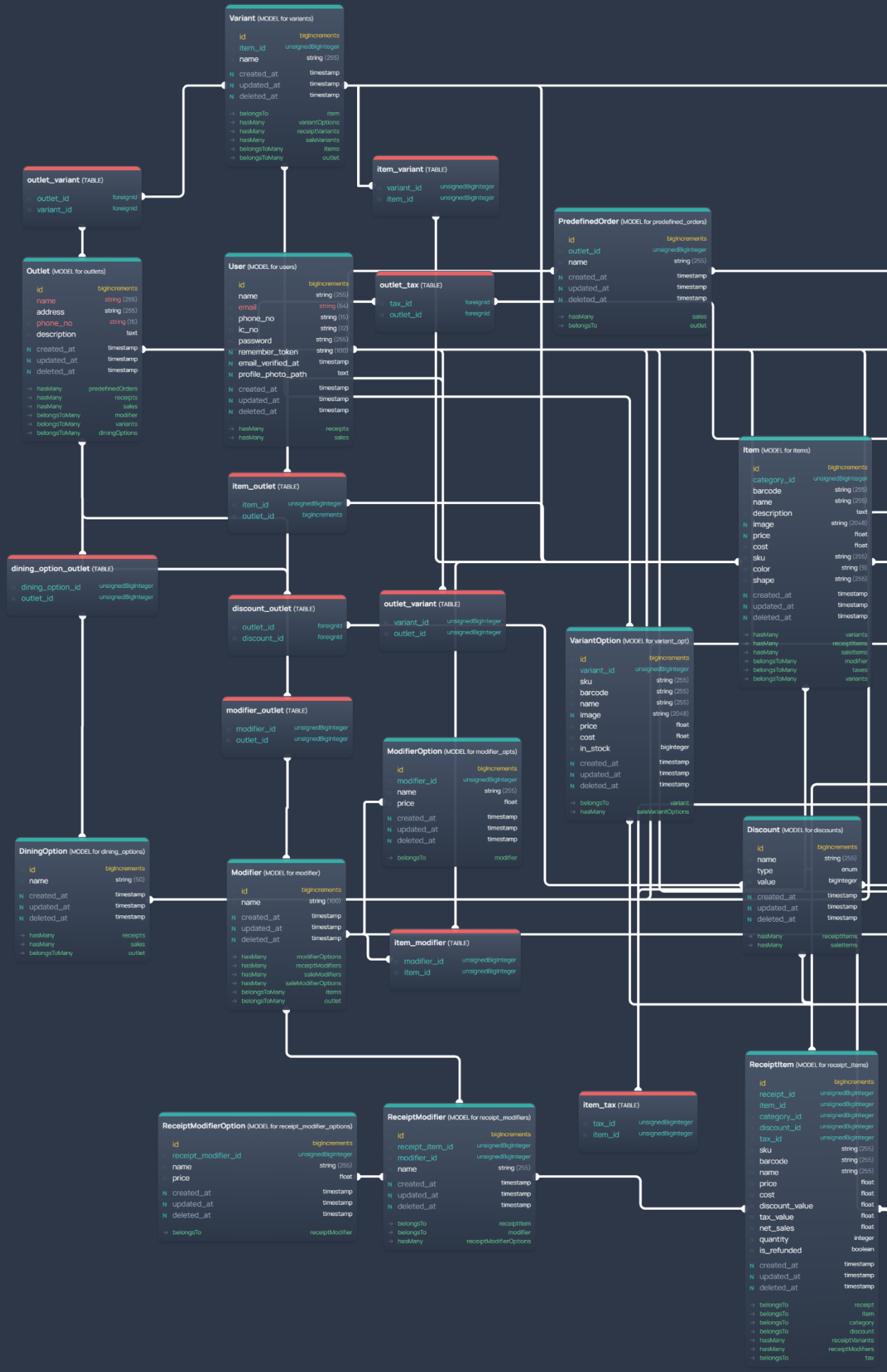
As part of the UAT for the MTS POS System, it was identified that the current system lacks sufficient reporting capabilities, specifically the ability to export data to Excel and provide in-depth insights into sales. To address this limitation, a future work recommendation is to enhance the reporting module of the system. This would involve developing functionalities that allow users to export sales data and other relevant information to Excel spreadsheets for further analysis and reporting purposes. Additionally, implementing advanced reporting features such as graphical representations, charts, and customizable dashboards would enable users to gain valuable insights into sales trends, top-selling items, and performance metrics. By incorporating these enhancements, the MTS POS System would provide users with comprehensive reporting capabilities, facilitating data-driven decision-making and enabling businesses to optimize their operations and drive growth. (Fuscaldo, 2022)

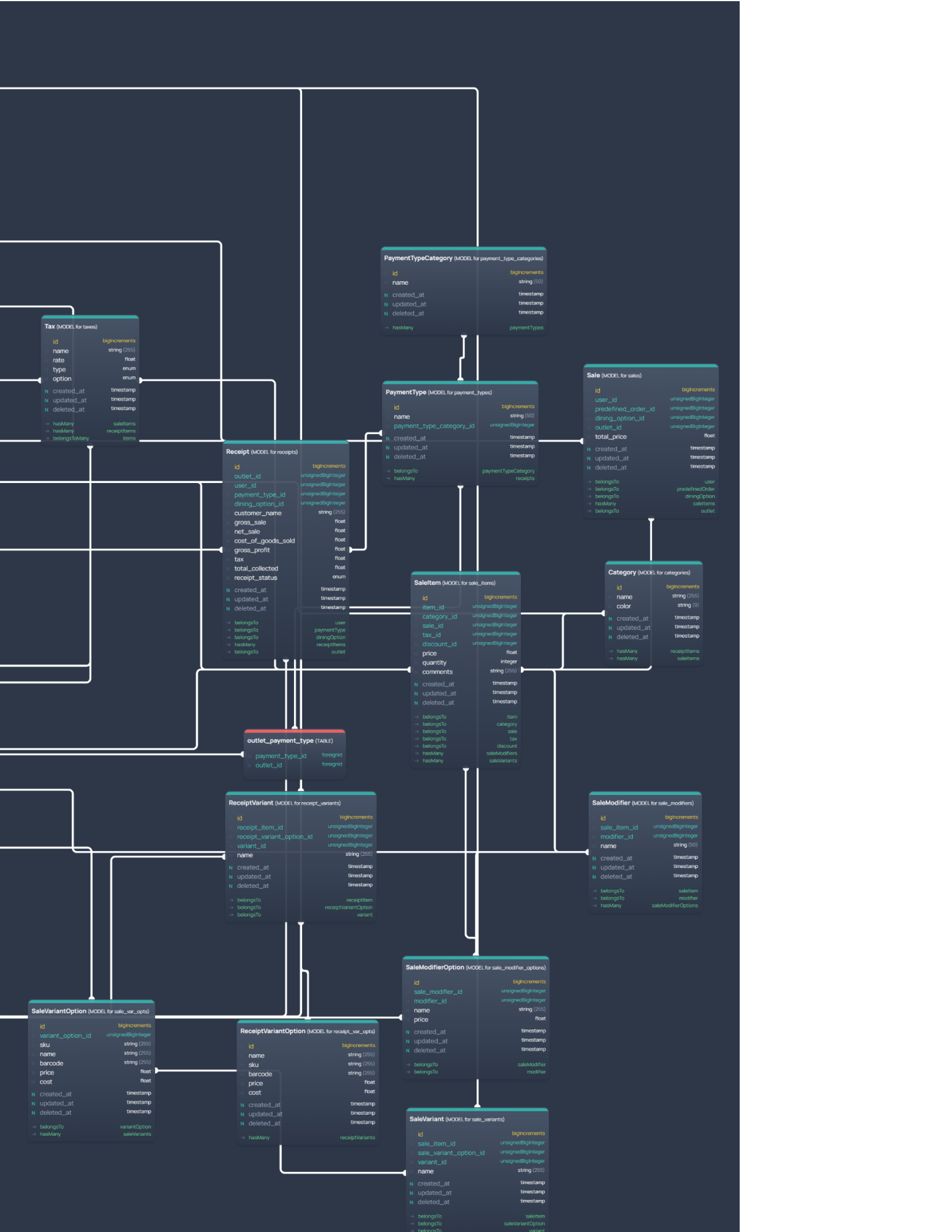
6 REFERENCES

- 12 Usability Testing Templates/Examples [+Checklist] | Hotjar. (2022). Retrieved February 17, 2023, from hotjar.com website: <https://www.hotjar.com/usability-testing/template-checklist/>
- Bakery & Coffee Shop POS System | Lightspeed Restaurant Point of Sale. (2023). Retrieved June 12, 2023, from Lightspeedhq website: <https://www.lightspeedhq.com/pos/restaurant/bakery-and-cafe-pos-system/>
- Forsyth, A. (1999). The Information Age: Economy, Society and Culture. *Journal of Planning Education and Research*, Vol. 19, pp. 211–213. <https://doi.org/10.1177/0739456X9901900212>
- Fuscaldo, D. (2022). How POS Reports Can Help You Run Your Business Better - businessnewsdaily.com. Retrieved June 12, 2023, from businessnewsdaily website: <https://www.businessnewsdaily.com/10449-time-saving-pos-reports.html>
- Imam, A. (2018). Software Development Life Cycle: The phases of SDLC - TestLodge Blog. Retrieved November 12, 2019, from Testlodge website: <https://blog.testlodge.com/software-development-life-cycle/>
- Laravel Naming Conventions - Web dev etc - my software development blog. (2022). Retrieved February 17, 2023, from webdevetc.com website: <https://webdevetc.com/blog/laravel-naming-conventions/>
- POS (Point of Sale) Software for your business | Square. (2023). Retrieved June 12, 2023, from Squareup website: <https://squareup.com/us/en/point-of-sale/software>
- Sinha, A., & Das, P. (2021). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. *2021 5th International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech 2021*. <https://doi.org/10.1109/IEMENTech53263.2021.9614779>
- Toast. (2019). Toast POS. Retrieved June 12, 2023, from 2019 website: <https://pos.toasttab.com/products>

What is Waterfall model- Examples, advantages, disadvantages & when to use it? (2022). Retrieved December 6, 2022, from <http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/>

APPENDIX A





APPENDIX B



MYSZTECH
Solutions

MTS POS SYSTEM DATA DICTIONARY

Prepared By Mohamad Mohsin Bin Ismail

TABLE OF CONTENTS

1	Data Dictionary MTS POS System.....	4
1.1	users	4
1.2	outlets.....	4
1.3	items	5
1.4	modifiers.....	5
1.5	modifier_opts	5
1.6	item_modifier.....	6
1.7	modifier_outlet	6
1.8	variants.....	6
1.9	variant_opt	6
1.10	outlet_variant.....	7
1.11	categories	7
1.12	taxes	8
1.13	item_tax.....	8
1.14	discounts.....	8
1.15	predefined_orders.....	9
1.16	receipts.....	9
1.17	receipt_items.....	10
1.18	receipt_variants.....	11
1.19	receipt_var_opts	11
1.20	receipt_modifiers	11
1.21	receipt_modifier_options.....	12
1.22	dining_options.....	12
1.23	dining_option_outlet.....	12
1.24	payment_types.....	13
1.25	sales	13
1.26	sale_items.....	13
1.27	sale_modifiers	14
1.28	sale_modifier_options.....	14
1.29	sale_variants.....	15
1.30	sale_var_opts	15
1.31	item_outlet.....	16
1.32	outlet_payment_type.....	16
1.33	outlet_tax	16

1.34	outlet_variant.....	16
1.35	discount_outlet	16
1.36	Laravel Generated Database	17
1.36.1	personal_access_tokens (Generated).....	17
1.36.2	password_resets (Generated).....	17
1.36.3	jobs (Generated).....	17
1.36.4	migrations (Generated)	17
1.36.5	oauth_clients (Generated)	17
1.36.6	oauth_auth_codes (Generated).....	18
1.36.7	oauth_refresh_tokens (Generated)	18
1.36.8	oauth_personal_access_clients (Generated).....	18
1.36.9	sessions (Generated).....	18
1.36.10	failed_jobs (Generated).....	19
1.36.11	oauth_access_tokens (Generated).....	19
1.36.12	api_keys (Generated)	19
1.36.13	api_key_access_evants (Generated).....	19
1.36.14	api_key_admin_events (Generated).....	20
1.36.15	links (Generated)	20
1.36.16	groups (Generated)	20
1.37	Package khsing/world.....	21
1.37.1	world_divisions (Generated)	21
1.37.2	world_cities (Generated).....	21
1.37.3	world_countries_locale (Generated)	21
1.37.4	world_divisions_locale (Generated).....	21
1.37.5	world_cities_locale (Generated)	22
1.37.6	world_continents_locale (Generated)	22
1.37.7	world_countries (Generated).....	22
1.38	Package spatie/Laravel-permission.....	23
1.38.1	roles (Generated)	23
1.38.2	permissions (Generated).....	23
1.38.3	role_has_permissions (Generated)	23
1.38.4	model_has_roles (Generated).....	23
1.38.5	model_has_permissions (Generated)	23
1.39	Package spatie/Laravel-activitylog.....	24
1.39.1	activity_log (Generated).....	24
1.40	Package seshac/otp-generator.....	24

1.40.1	otps (Generated)	24
--------	------------------------	----

1 Data Dictionary MTS POS System

1.1 users

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Users' id
name	varchar	255		Users' name
email	varchar	64		Users' email
phone_no	varchar	15		Users' contact number
ic_no	varchar	12		Users's identity card number
password	varchar	255		Users' account password
remember_token	varchar	100	NULL	
email_verified_at	timestamp		NULL	Time and date the email get verified
profile_photo_path	text		NULL	
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.2 outlets

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Outlets id
name	varchar	255		Outlets' name
address	varchar	255		Outlets' address
phone_no	varchar	15		Outlets's contact number
description	text			Description about outlets
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.3 items

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Item id
category_id (FK)	bigint	20		Category id for the item
barcode	varchar	255		Item barcode
name	varchar	255		Item name
description	text		NULL	Item description
image	varchar	2048	NULL	Item image
price	float	11,2	NULL	Item price
cost	float	11,2		Item cost
sku	varchar	255		Item stock keeping unit
color	varchar	9	NULL	Item color
shape	varchar	255	NULL	Item shape
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.4 modifiers

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Modifier id
name	varchar	100		Modifier name
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.5 modifier_opts

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Modifier option id
modifier_id (FK)	bigint	20		Modifier id
name	varchar	255		Modifier option name

price	float	11,2	NULL	Modifier option price
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.6 item_modifier

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
item_id (FK)	bigint	20		Foreign Key (pivot table)
modifier_id (FK)	bigint	20		Foreign Key (pivot table)

1.7 modifier_outlet

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
modifier_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.8 variants

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Variant id
item_id(FK)	bigint	20		Item id
name	varchar	255		Variant name
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.9 variant_opt

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Variant option id

variant_id (FK)	bigint	20		Variant id
sku	varchar	255		Variant option stock keeping unit
barcode	varchar	255		Variant option barcode
name	varchar	255		Variant option name
image	varchar	2048	NULL	Variant option image
price	float	11,2		Variant option price
cost	float	11,2		Variant option cost
in_stock	bigint	20	NULL	How many variant options available in stock
deleted_at	timestamp			Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.10 outlet_variant

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
outlet_id (FK)	bigint	20		Foreign Key (pivot table)
variant_id (FK)	bigint	20		Foreign Key (pivot table)

1.11 categories

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Category id
name	varchar	255		Category name for item
color	varchar	9		Category color
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.12 taxes

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Tax id
name	varchar	255		Tax name
rate	float	11,2		Value rate for tax
type	enum		<ol style="list-style-type: none"> 1. to_new_items 2. to_existing_items 3. to_new_and_existing_items 4. dont_apply 	Tax type
option	enum		<ol style="list-style-type: none"> 1. included 2. added 	Tax option
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.13 item_tax

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
item_id (FK)	bigint	20		Foreign Key (pivot table)
tax_id (FK)	bigint	20		Foreign Key (pivot table)

1.14 discounts

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Discount id
name	varchar	255		Discount name
type	enum		<ol style="list-style-type: none"> 1. percentage 2. amount 	Discount unit
value	bigint			Discount value

deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.15 predefined_orders

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Predefined order id
outlet_id(FK)	bigint	20		Outlet id
name	varchar	255		Prdefined order name
deleted_at	timestamp		NULL	Time and date the data deleted
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

1.16 receipts

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Receipt id
outlet_id (FK)	bigint	20		Outlet id
user_id (FK)	bigint	20		User id
payment_type_id (FK)	bigint	20		Payment type id
dining_option_id (FK)	bigint	20		Dining option id
customer_name	varchar	255		Customer name for the receipt
gross_sale	float	11,2		Total unit * Original sale price
net_sale	float	11,2		gross sales - discount
cost_of_goods_sold	float	11,2		Beginning inventory + purchases and other costs - ending inventory

gross_profit	float	11,2		net sales - cogs
tax	float	11,2		Tax amount
total_collected	float	11,2		gross profit - tax
receipt_status	enum		0 = Cancelled 1 = Refunded 2 = Normal	Status for the receipt
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.17 receipt_items

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Receipt item id
receipt_id (FK)	bigint	20		Receipt id
item_id (FK)	bigint	20		Item id
category_id (FK)	bigint	20		Category id for the item
discount_id (FK)	bigint	20		Discount id
tax_id (FK)	bigint	20		Tax id
sku	varchar	255		Receipt item stock keeping unit
barcode	varchar	255		Receipt item barcode
name	varchar	255		Receipt item name
price	float	11,2		Item price in the receipt
cost	float	11,2		Item cost in the receipt
discount_value	float	11,2		Item discount value
tax_value	float	11,2		Item tax value
net_sales	float	11,2		Item net sales
quantity	int			Quantity item in the receipt
is_refunded	bool		FALSE	
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.18 receipt_variants

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Receipt variant id
receipt_item_id (FK)	bigint	20		Receipt item id
receipt_variant_option_id (FK)	bigint	20		Receipt variant option id
variant_id (FK)	bigint	20		Variant id
name	varchar	255		Variant name in the receipt
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.19 receipt_var_opts

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
sku	varchar	255		
barcode	varchar	255		
name	varchar	255		
price	float	11,2		
cost	float	11,2		
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.20 receipt_modifiers

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
receipt_item_id (FK)	bigint	20		

modifier_id (FK)	bigint	20		
name	varchar	255		
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.21 receipt_modifier_options

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
receipt_modifier_id (FK)	bigint	20		
name	varchar	255		
price	float	11,2		
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp		NULL	Time and date the data deleted

1.22 dining_options

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Dining option id
name	varchar	50		Dining option name
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.23 dining_option_outlet

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
outlet_id (FK)	bigint	20		Foreign Key (pivot table)
dining_option_id (FK)	varchar	50		Foreign Key (pivot table)

1.24 payment_types

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Payment type id
payment_type_category	enum		1. card 2. check 3. other	Payment type category
name	varchar	50		Payment type name
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.25 sales

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sales id
user_id (FK)	bigint	20		User id
predefined_order_id (FK)	bigint	20		Predefined order id
dining_option_id (FK)	bigint	20		Dining option id
outlet_id (FK)	bigint	20		Outlet id
total_price	float	11,2		Total of price for the sales
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.26 sale_items

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sale by item id
item_id (FK)	bigint	20		Item id

category_id (FK)	bigint	20		Item Category
sale_id (FK)	bigint	20		Sale id
tax_id (FK)	bigint	20		Tax id
discount_id (FK)	bigint	20		Discount id
price	float	11,2		Sale item price
quantity	int	20		Quantity of item that had been sale
comments	varchar	255		Comments for the sale item
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.27 sale_modifiers

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sale modifiers id
sale_item_id (FK)	bigint	20		Sale item id
modifier_id (FK)	bigint	20		Modifier id
name	varchar	50		Sale modifier name
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.28 sale_modifier_options

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sale modifier option id
sale_modifier_id (FK)	bigint	20		Sale modifier id
modifier_id (FK)	bigint	20		Modifier id
name	varchar	50		Sale modifier option name
price	float	11,2		Sale modifier option price
created_at	timestamp			Time and date the data created

updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.29 sale_variants

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sale variant id
sale_item_id (FK)	bigint	20		Sale item id
sale_variant_option_id (FK)	bigint	20		Sale variant option id
variant_id (FK)	bigint	20		Variant id
name	varchar	50		Sale variant name
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated
deleted_at	timestamp			Time and date the data deleted

1.30 sale_var_opts

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		Sale Variant Option id
variant_option_id (FK)	bigint	20		Variant option id
sku	varchar	255		Variant option stock keeping unit for the sale
barcode	varchar	255		Sale Variant Option barcode
name	varchar	255		Sale Variant Option name
price	float	11,2		Sale Variant Option price
cost	float	11,2		Sale Variant Option cost
created_at	timestamp			Time and date the data created
updated_at	timestamp			Time and date the data updated

deleted_at	timestamp		NULL	Time and date the data deleted
-------------------	-----------	--	------	--------------------------------

1.31 item_outlet

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
item_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.32 outlet_payment_type

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
payment_type_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.33 outlet_tax

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
tax_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.34 outlet_variant

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
variant_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.35 discount_outlet

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
discount_id (FK)	bigint	20		Foreign Key (pivot table)
outlet_id (FK)	bigint	20		Foreign Key (pivot table)

1.36 Laravel Generated Database

1.36.1 personal_access_tokens (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
tokenable	BIGINT	20		
name	VARCHAR	255		
token	VARCHAR	64	NULL	
abilities	TEXT		NULL	
last_used_at	TIMESTAMP		NULL	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.2 password_resets (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
email	VARCHAR	255		
token	VARCHAR	255		
created_at	TIMESTAMP		NULL	

1.36.3 jobs (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
queue	VARCHAR	255		
payload	LONGTEXT			
attempts	TINYINT	3		
reserved_at	INT	10		
available_at	INT	10		
created_at	INT	10		

1.36.4 migrations (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	INT	10		
migration	VARCHAR	255		
batch	INT	11		

1.36.5 oauth_clients (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
user_id	BIGINT	20	NULL	

name	VARCHAR	255		
secret	VARCHAR	100	NULL	
provider	VARCHAR	255	NULL	
redirect	TEXT			
personal_access_client	TINYINT	1		
password_client	TINYINT	1		
revoked	TINYINT	1		
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.6 oauth_auth_codes (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	VARCHAR	100		
user_id	BIGINT	20		
client_id	BIGINT	20		
scopes	TEXT		NULL	
revoked	TINYINT	1		
expires_at	DATETIME		NULL	

1.36.7 oauth_refresh_tokens (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	VARCHAR	100		
access_token_id	VARCHAR	100		
revoked	TINYINT	1		
expires_at	DATETIME		NULL	

1.36.8 oauth_personal_access_clients (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
client_id	BIGINT	20		
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.9 sessions (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
Id (PK)	VARCHAR	255		
user_id	BIGINT	20	NULL	
ip_address	VARCHAR	45	NULL	
user_agent	TEXT		NULL	
payload	TEXT			
last_activity	INT	11		

created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.10 failed_jobs (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	bigint	20		
uuid	varchar	255		
connection	text			
queue	text			
payload	longtext			
exception	longtext			
failed_at	timestamp			

1.36.11oauth_access_tokens (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	VARCHAR	100		
user_id	BIGINT	20	NULL	
client_id	BIGINT	20		
name	VARCHAR	255	NULL	
scopes	TEXT		NULL	
revoked	TINYINT	1		
created_at	TIMESTAMP			
updated_at	TIMESTAMP			
expires_at	DATETIME		NULL	

1.36.12api_keys (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	100		
name	VARCHAR	255		
key	VARCHAR	64		
active	BOOL	255	1	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			
deleted_at	TIMESTAMP			

1.36.13api_key_access_evants (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	100		
api_key_id	BIGINT	100		

(FK)				
ip_address	VARCHAR	64		
url	TEXT	255	1	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.14api_key_admin_events (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	100		
api_key_id (FK)	BIGINT	100		
ip_address	VARCHAR	64		
event	VARCHAR	255	1	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.15links (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	100		
group_id (FK)	BIGINT	100	NULL	
url	VARCHAR	255		
data	TEXT	255		
expiry	DATETIME	255	NULL	
click_limit	INT	100	NULL	
clicks	INT	100	0	
uuid	VARCHAR	255		
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.36.16groups (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	100		
expiry	DATETIME	255	NULL	
click_limit	INT	100	NULL	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

1.37 Package khsing/world

For world database.

1.37.1 world_divisions (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
country_id	BIGINT	20		
name	VARCHAR	255		
full_name	VARCHAR	255		
code	VARCHAR	64		
has_city	TINYINT	1		
total_vendor	INT	11		

1.37.2 world_cities (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
country_id	BIGINT	20		
division_id	BIGINT	20		
name	VARCHAR	255		
full_name	VARCHAR	255		
code	VARCHAR	64		

1.37.3 world_countries_locale (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
country_id	BIGINT	20		
name	VARCHAR	255		
alias	VARCHAR	255		
abbr	VARCHAR	16		
full_name	VARCHAR	255		
current_name	VARCHAR	255		
locale	VARCHAR	6		

1.37.4 world_divisions_locale (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
division_id	BIGINT	20		
name	VARCHAR	255		
abbr	VARCHAR	16		
alias	VARCHAR	255		

full_name	VARCHAR	255		
locale	VARCHAR	6		

1.37.5 world_cities_locale (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
city_id	BIGINT	20		
name	VARCHAR	255		
alias	VARCHAR	255		
full_name	VARCHAR	255		
locale	VARCHAR	6		

1.37.6 world_continents_locale (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	bigint	20		
continent_id	bigint	20		
name	varchar	255		
alias	varchar	255		
abbr	varchar	16		
full_name	varchar	255		
locale	varchar	6		

1.37.7 world_countries (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id	BIGINT	20		
continent_id	BIGINT	20		
name	VARCHAR	255		
full_name	VARCHAR	255		
capital	VARCHAR	255		
code	VARCHAR	4		
code_alpha3	VARCHAR	6		
emoji	VARCHAR	16		
has_division	TINYINT	1		
currency_code	VARCHAR	3		
currency_name	VARCHAR	128		
tld	VARCHAR	8		
callingcode	VARCHAR	8		
<i>serial_number_formula</i>	<i>VARCHAR</i>	<i>255</i>		

1.38 Package spatie/Laravel-permission

1.38.1 roles (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
name	varchar	255		
guard_name	varchar	255		
created_at	timestamp			
updated_at	timestamp			

1.38.2 permissions (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
name	varchar	255		
guard_name	varchar	255		
created_at	timestamp			
updated_at	timestamp			

1.38.3 role_has_permissions (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
permission_id (PK)	bigint	20		
role_id (PK)(FK)	bigint	20		

1.38.4 model_has_roles (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
role_id (PK)	bigint	20		
model_type (PK)(FK)	varchar	255		
model_id (PK)(FK)	bigint	20		

1.38.5 model_has_permissions (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
permission_id (PK)	bigint	20		
model_type (PK)(FK)	varchar	255		

1.39 Package spatie/Laravel-activitylog

1.39.1 activity_log (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
log_name (PK)(FK)	varchar	255	NULL	
description	text	2555		
subject	morphs			
causer	morphs			
properties	json			
created_at	timestamp			
updated_at	timestamp			

1.40 Package seshac/otp-generator

1.40.1 otps (Generated)

FIELD NAME	DATA TYPE	FIELD SIZE	DEFAULT VALUE	COMMENT
id (PK)	bigint	20		
identifier	varchar	255		
token	varchar	255		
validity	int			
expired	bool		FALSE	
no_times_generated	int		0	
no_times_attempted	int		0	
generated_at	timestamp			
created_at	timestamp			
updated_at	timestamp			

APPENDIX C

User Acceptance Test for MTS POS System

Hye, we would like to hear your thoughts or feedback on the system that has been developed.
Thank you for your time.

Email *

hairi@mysztech.com

Name *

Hairi

Usability Feedback

This section is to evaluate the overall system satisfaction.

The system works effectively for me. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system is user friendly (easy to use and to interact with). *

Strongly Disagree

1

2

3

4

5

Strongly Agree

It is easy to use the system to make an order. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has high validation and clearly provides a guide to fulfill the requirement. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The interface of the system is consistent and attractive. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system provides a clear information (e.g., menu information, order information) *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has great information organization. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system fulfills all the requirements as a basic POS system. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

How likely are you to recommend the MTS POS System to other businesses in your industry? *

Very Low

1

2

3

4

5

Very High

From 1 to 5 what is your rating to this POS system? *

Very Low

1

2

3

4

5

Very High

Table Layout Feedback

This section is to evaluate the effectiveness and satisfaction of the Table Layout Module

How intuitive and user-friendly is the table layout in terms of navigating and managing tables? *

Very difficult to navigate and manage tables

1

2

3

4

5

Extremely intuitive and user-friendly

How visually appealing and aesthetically pleasing is the table layout design *

Very unappealing and visually cluttered

1

2

3

4

5

Extremely attractive and visually stunning

How responsive is the table layout in terms of updating and reflecting real-time changes? *

Very slow and delayed updates

1

2

3

4

5

Instantaneous updates and highly responsive

How satisfied are you with the level of customization and flexibility offered by the table layout in terms of arranging and organizing tables based on your specific business needs? *

Extremely dissatisfied

1

2

3

4

5

Extremely Satisfied

Login Module

This section will be evaluating the effectiveness and satisfaction of the Login Module

Can you login into the system using correct credentials? *

Yes

No

Sales Module

This section will be evaluating the effectiveness and satisfaction of Sales Module

Can you add items to sale? *

Yes

No

Can you edit items in sale? *

Yes

No

Can you remove items from sale? *

Yes

No

Are you able to add discount to the sale? *

Yes

No

Are you able to remove discount from sale? *

Yes

No

Are you able to add taxes to the sale? *

Yes

No

Are you able remove taxes from sale? *

Yes

No

Are you able to charge the customer? *

Yes

No

Are you able to store the sale in predefined order? *

Yes

No

Are you able create the order using the table layout? *

Yes

No

Table Layout Module

This section will be evaluating the effectiveness and satisfaction of Table Layout Module

Can you view correct table orders? *

Yes

No

Can you edit the table layout? *

Yes

No

Can you assign name to the table? *

Yes

No

Can you assign predefined order to the table? *

Yes

No

Can you delete table? *

Yes

No

Can you open order from the table? *

Yes

No

Can you add section? *

Yes

No

Can you edit section name? *

Yes

No

Manage Items Module

This section will be evaluating the effectiveness and satisfaction of Manage Items

Are you able to view all items? *

Yes

No

Are you able to add items? *

Yes

No

Are you able to edit items? *

Yes

No

Are you able to delete items? *

Yes

No

Manage Categories Module

This section will be evaluating the effectiveness and satisfaction of Manage Categories

Can you view all categories? *

Yes

No

Can you add categories? *

Yes

No

Can you edit categories? *

Yes

No

Can you delete categories? *

Yes

No

Manage Discounts Module

This section will be evaluating the effectiveness and satisfaction of Manage Discounts

Are you able to view all discounts? *

Yes

No

Are you able to add discounts? *

Yes

No

Are you able to edit discounts? *

Yes

No

Are you able to delete discounts? *

Yes

No

Manage Taxes Module

This section will be evaluating the effectiveness and satisfaction of Manage

Taxes

Are you able to view all taxes? *

Yes

No

Are you able to add taxes? *

Yes

No

Are you able to edit taxes? *

Yes

No

Are you able to delete taxes? *

Yes

No

Manage Outlets Module

This section will be evaluating the effectiveness and satisfaction of Manage Outlets

Are you able to view all outlets? *

Yes

No

User Acceptance Test for MTS POS System

Hye, we would like to hear your thoughts or feedback on the system that has been developed.
Thank you for your time.

Email *

wanmuhamadhaziqirfan@gmail.com

Name *

Wan Muhamad Haziq Irfan Bin Hafiz Sayuti

Usability Feedback

This section is to evaluate the overall system satisfaction.

The system works effectively for me. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system is user friendly (easy to use and to interact with). *

Strongly Disagree

1

2

3

4

5

Strongly Agree

It is easy to use the system to make an order. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has high validation and clearly provides a guide to fulfill the requirement. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The interface of the system is consistent and attractive. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system provides a clear information (e.g., menu information, order information) *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has great information organization. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system fulfills all the requirements as a basic POS system. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

How likely are you to recommend the MTS POS System to other businesses in your industry? *

Very Low

1

2

3

4

5

Very High

From 1 to 5 what is your rating to this POS system? *

Very Low

1

2

3

4

5

Very High

Table Layout Feedback

This section is to evaluate the effectiveness and satisfaction of the Table Layout Module

How intuitive and user-friendly is the table layout in terms of navigating and managing tables? *

Very difficult to navigate and manage tables

1

2

3

4

5

Extremely intuitive and user-friendly

How visually appealing and aesthetically pleasing is the table layout design *

Very unappealing and visually cluttered

1

2

3

4

5

Extremely attractive and visually stunning

How responsive is the table layout in terms of updating and reflecting real-time changes? *

Very slow and delayed updates

1

2

3

4

5

Instantaneous updates and highly responsive

How satisfied are you with the level of customization and flexibility offered by the table layout in terms of arranging and organizing tables based on your specific business needs? *

Extremely dissatisfied

1

2

3

4

5

Extremely Satisfied

Login Module

This section will be evaluating the effectiveness and satisfaction of the Login Module

Can you login into the system using correct credentials? *

Yes

No

Sales Module

This section will be evaluating the effectiveness and satisfaction of Sales Module

Can you add items to sale? *

Yes

No

Can you edit items in sale? *

Yes

No

Can you remove items from sale? *

Yes

No

Are you able to add discount to the sale? *

Yes

No

Are you able to remove discount from sale? *

Yes

No

Are you able to add taxes to the sale? *

Yes

No

Are you able remove taxes from sale? *

Yes

No

Are you able to charge the customer? *

Yes

No

Are you able to store the sale in predefined order? *

Yes

No

Are you able create the order using the table layout? *

Yes

No

Table Layout Module

This section will be evaluating the effectiveness and satisfaction of Table Layout Module

Can you view correct table orders? *

Yes

No

Can you edit the table layout? *

Yes

No

Can you assign name to the table? *

Yes

No

Can you assign predefined order to the table? *

Yes

No

Can you delete table? *

Yes

No

Can you open order from the table? *

Yes

No

Can you add section? *

Yes

No

Can you edit section name? *

Yes

No

Manage Items Module

This section will be evaluating the effectiveness and satisfaction of Manage Items

Are you able to view all items? *

Yes

No

Are you able to add items? *

Yes

No

Are you able to edit items? *

Yes

No

Are you able to delete items? *

Yes

No

Manage Categories Module

This section will be evaluating the effectiveness and satisfaction of Manage Categories

Can you view all categories? *

Yes

No

Can you add categories? *

Yes

No

Can you edit categories? *

Yes

No

Can you delete categories? *

Yes

No

Manage Discounts Module

This section will be evaluating the effectiveness and satisfaction of Manage Discounts

Are you able to view all discounts? *

Yes

No

Are you able to add discounts? *

Yes

No

Are you able to edit discounts? *

Yes

No

Are you able to delete discounts? *

Yes

No

Manage Taxes Module

This section will be evaluating the effectiveness and satisfaction of Manage

Taxes

Are you able to view all taxes? *

Yes

No

Are you able to add taxes? *

Yes

No

Are you able to edit taxes? *

Yes

No

Are you able to delete taxes? *

Yes

No

Manage Outlets Module

This section will be evaluating the effectiveness and satisfaction of Manage Outlets

Are you able to view all outlets? *

Yes

No

Are you able to add outlets? *

Yes

No

Are you able to edit outlets? *

Yes

No

Are you able to delete outlets? *

Yes

No

This content is neither created nor endorsed by Google.

Google Forms

User Acceptance Test for MTS POS System

Hye, we would like to hear your thoughts or feedback on the system that has been developed.
Thank you for your time.

Email *

sulaimanzurkifli@gmail.com

Name *

Ahmad Sulaiman Bin Zurkifli

Usability Feedback

This section is to evaluate the overall system satisfaction.

The system works effectively for me. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system is user friendly (easy to use and to interact with). *

Strongly Disagree

1

2

3

4

5

Strongly Agree

It is easy to use the system to make an order. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has high validation and clearly provides a guide to fulfill the requirement. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The interface of the system is consistent and attractive. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system provides a clear information (e.g., menu information, order information) *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has great information organization. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system fulfills all the requirements as a basic POS system. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

How likely are you to recommend the MTS POS System to other businesses in your industry? *

Very Low

1

2

3

4

5

Very High

From 1 to 5 what is your rating to this POS system? *

Very Low

1

2

3

4

5

Very High

Table Layout Feedback

This section is to evaluate the effectiveness and satisfaction of the Table Layout Module

How intuitive and user-friendly is the table layout in terms of navigating and managing tables? *

Very difficult to navigate and manage tables

1

2

3

4

5

Extremely intuitive and user-friendly

How visually appealing and aesthetically pleasing is the table layout design *

Very unappealing and visually cluttered

1

2

3

4

5

Extremely attractive and visually stunning

How responsive is the table layout in terms of updating and reflecting real-time changes? *

Very slow and delayed updates

1

2

3

4

5

Instantaneous updates and highly responsive

How satisfied are you with the level of customization and flexibility offered by the table layout in terms of arranging and organizing tables based on your specific business needs? *

Extremely dissatisfied

1

2

3

4

5

Extremely Satisfied

Login Module

This section will be evaluating the effectiveness and satisfaction of the Login Module

Can you login into the system using correct credentials? *

Yes

No

Sales Module

This section will be evaluating the effectiveness and satisfaction of Sales Module

Can you add items to sale? *

Yes

No

Can you edit items in sale? *

Yes

No

Can you remove items from sale? *

Yes

No

Are you able to add discount to the sale? *

Yes

No

Are you able to remove discount from sale? *

Yes

No

Are you able to add taxes to the sale? *

Yes

No

Are you able remove taxes from sale? *

Yes

No

Are you able to charge the customer? *

Yes

No

Are you able to store the sale in predefined order? *

Yes

No

Are you able create the order using the table layout? *

Yes

No

Table Layout Module

This section will be evaluating the effectiveness and satisfaction of Table Layout Module

Can you view correct table orders? *

Yes

No

Can you edit the table layout? *

Yes

No

Can you assign name to the table? *

Yes

No

Can you assign predefined order to the table? *

Yes

No

Can you delete table? *

Yes

No

Can you open order from the table? *

Yes

No

Can you add section? *

Yes

No

Can you edit section name? *

Yes

No

Manage Items Module

This section will be evaluating the effectiveness and satisfaction of Manage Items

Are you able to view all items? *

Yes

No

Are you able to add items? *

Yes

No

Are you able to edit items? *

Yes

No

Are you able to delete items? *

Yes

No

Manage Categories Module

This section will be evaluating the effectiveness and satisfaction of Manage Categories

Can you view all categories? *

Yes

No

Can you add categories? *

Yes

No

Can you edit categories? *

Yes

No

Can you delete categories? *

Yes

No

Manage Discounts Module

This section will be evaluating the effectiveness and satisfaction of Manage Discounts

Are you able to view all discounts? *

Yes

No

Are you able to add discounts? *

Yes

No

Are you able to edit discounts? *

Yes

No

Are you able to delete discounts? *

Yes

No

Manage Taxes Module

This section will be evaluating the effectiveness and satisfaction of Manage

Taxes

Are you able to view all taxes? *

Yes

No

Are you able to add taxes? *

Yes

No

Are you able to edit taxes? *

Yes

No

Are you able to delete taxes? *

Yes

No

Manage Outlets Module

This section will be evaluating the effectiveness and satisfaction of Manage Outlets

Are you able to view all outlets? *

Yes

No

Are you able to add outlets? *

Yes

No

Are you able to edit outlets? *

Yes

No

Are you able to delete outlets? *

Yes

No

This content is neither created nor endorsed by Google.

Google Forms

Are you able to add outlets? *

Yes

No

Are you able to edit outlets? *

Yes

No

Are you able to delete outlets? *

Yes

No

This content is neither created nor endorsed by Google.

Google Forms

User Acceptance Test for MTS POS System

Hye, we would like to hear your thoughts or feedback on the system that has been developed.
Thank you for your time.

Email *

aslammatasri@gmail.com

Name *

MUHAMMAD ASLAM BIN MAT ASRI

Usability Feedback

This section is to evaluate the overall system satisfaction.

The system works effectively for me. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system is user friendly (easy to use and to interact with). *

Strongly Disagree

1

2

3

4

5

Strongly Agree

It is easy to use the system to make an order. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has high validation and clearly provides a guide to fulfill the requirement. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The interface of the system is consistent and attractive. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system provides a clear information (e.g., menu information, order information) *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system has great information organization. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

The system fulfills all the requirements as a basic POS system. *

Strongly Disagree

1

2

3

4

5

Strongly Agree

How likely are you to recommend the MTS POS System to other businesses in your industry? *

Very Low

1

2

3

4

5

Very High

From 1 to 5 what is your rating to this POS system? *

Very Low

1

2

3

4

5

Very High

Table Layout Feedback

This section is to evaluate the effectiveness and satisfaction of the Table Layout Module

How intuitive and user-friendly is the table layout in terms of navigating and managing tables? *

Very difficult to navigate and manage tables

1

2

3

4

5

Extremely intuitive and user-friendly

How visually appealing and aesthetically pleasing is the table layout design *

Very unappealing and visually cluttered

1

2

3

4

5

Extremely attractive and visually stunning

How responsive is the table layout in terms of updating and reflecting real-time changes? *

Very slow and delayed updates

1

2

3

4

5

Instantaneous updates and highly responsive

How satisfied are you with the level of customization and flexibility offered by the table layout in terms of arranging and organizing tables based on your specific business needs? *

Extremely dissatisfied

1

2

3

4

5

Extremely Satisfied

Login Module

This section will be evaluating the effectiveness and satisfaction of the Login Module

Can you login into the system using correct credentials? *

Yes

No

Sales Module

This section will be evaluating the effectiveness and satisfaction of Sales Module

Can you add items to sale? *

Yes

No

Can you edit items in sale? *

Yes

No

Can you remove items from sale? *

Yes

No

Are you able to add discount to the sale? *

Yes

No

Are you able to remove discount from sale? *

Yes

No

Are you able to add taxes to the sale? *

Yes

No

Are you able remove taxes from sale? *

Yes

No

Are you able to charge the customer? *

Yes

No

Are you able to store the sale in predefined order? *

Yes

No

Are you able create the order using the table layout? *

Yes

No

Table Layout Module

This section will be evaluating the effectiveness and satisfaction of Table Layout Module

Can you view correct table orders? *

Yes

No

Can you edit the table layout? *

Yes

No

Can you assign name to the table? *

Yes

No

Can you assign predefined order to the table? *

Yes

No

Can you delete table? *

Yes

No

Can you open order from the table? *

Yes

No

Can you add section? *

Yes

No

Can you edit section name? *

Yes

No

Manage Items Module

This section will be evaluating the effectiveness and satisfaction of Manage Items

Are you able to view all items? *

Yes

No

Are you able to add items? *

Yes

No

Are you able to edit items? *

Yes

No

Are you able to delete items? *

Yes

No

Manage Categories Module

This section will be evaluating the effectiveness and satisfaction of Manage Categories

Can you view all categories? *

Yes

No

Can you add categories? *

Yes

No

Can you edit categories? *

Yes

No

Can you delete categories? *

Yes

No

Manage Discounts Module

This section will be evaluating the effectiveness and satisfaction of Manage Discounts

Are you able to view all discounts? *

Yes

No

Are you able to add discounts? *

Yes

No

Are you able to edit discounts? *

Yes

No

Are you able to delete discounts? *

Yes

No

Manage Taxes Module

This section will be evaluating the effectiveness and satisfaction of Manage

Taxes

Are you able to view all taxes? *

Yes

No

Are you able to add taxes? *

Yes

No

Are you able to edit taxes? *

Yes

No

Are you able to delete taxes? *

Yes

No

Manage Outlets Module

This section will be evaluating the effectiveness and satisfaction of Manage Outlets

Are you able to view all outlets? *

Yes

No

Are you able to add outlets? *

Yes

No

Are you able to edit outlets? *

Yes

No

Are you able to delete outlets? *

Yes

No

This content is neither created nor endorsed by Google.

Google Forms