UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

W O R K S H O P 1

R E P O R T

Name : CHOOI JUN XIANG

Matric Number : B032310599

Program : FTMK BITI

Project Title : Sushi Restaurant Menu Recommendation Order System

Supervisor Name : Ts. Dr. Siti Azirah Asmai

Supervisor Signature : _____

Evaluator Name : Prof. Madya Dr. Asmala Ahmad

EXECUTIVE SUMMARY

The project entitled "Sushi Restaurant Menu Suggestions System" is designed to make a revolution in the restaurant industry by applying and integrating the advanced technological solutions to the traditional culture of dining. It provides an efficient system for ordering, authentication, analysis, and personalized menu recommendations for customers, thus solving many issues involved in the operational inefficiencies of a conventional restaurant. With the programming language C++, the system is designed to perform CRUD operations efficiently to allow the smooth and integrity data management of the system. It made a provision for sales tracking features that create daily, monthly, and yearly reports regarding revenue towards business management to understand much at the performance of the business. The menu recommendation feature of the system is unique because it makes an analysis of the sales data, so it recommends popular items to the customers to enhance their dining experience and probably increase sales. Furthermore, from the modular point of view, the system can conveniently be extended for features like real-time inventory tracking and multi-language support. This project will not only facilitate the operation of a restaurant with good efficiency but also help improve the environment in terms of service due to its data-driven insight, leading to a more personalized and efficient service that can consequently improve customer satisfaction and business performance.

TABLES OF CONTENTS

PAGE

CHAPTER 1: INTRODUCTION

1.1 Introduction

Within the current and modern dining world, it is all sushi restaurants. They offer customers extensive and diverse menus full of culinary variety, but then again, they come along with decision fatigue as part of the dining experience bliss. The many options have created a situation with more exasperating decisions that lead one to have decision fatigue during the dining.

Traditional restaurant systems are often not your advanced systems, where features such as personalized recommendations and real-time order tracking are built. The lack of these can cause operational inefficiencies and a lackluster experience from a customer perspective. Besides, intertwining AI with different digital tools has been known to improve service delivery and satisfaction levels in the food industry.

The need for digital transformation in the food service industry is most urgent than ever to assert efficiency and convenience in businesses. The audience has evolved to expect convenience and personalization. In addition, AI and the Internet of Things (IoT) have developed the industry's future through their effects on decision making, sustainability, and traceability.

Thus, in response to such challenges and opportunities, the Sushi Restaurant Menu Suggestions System will be a holistic solution for the overall dining experience enhancement through intelligent menu recommendations and efficient order management. The solution offers a systematic approach in solving challenges related to menu selection and operation limitations from being a traditional system.

1.2 Problem Statement

Traditional restaurant management systems often lack robust recommendation features and effective analytics tools. This can lead to inefficiencies in sales tracking, inventory management, and customer satisfaction.

1. Customers face decision fatigue due to a large variety of options.

2. Existing order systems do not provide insights into popular menu items .

3. Manual calculations for sales reports can lead to inefficiencies.

1.3 Objectives

There are 3 objectives that this project will be achieved:

i. To create and design a sushi restaurant menu recommendation order system with complete ordering function for customer.

ii. Improve customer's dining experience in the restaurant with a good flexible order system.

iii. To develop a sales generated report system of sushi restaurant for data analysis purpose.

1.4 Scope

1.4.1 Module to be developed

The scope of the "Sushi Restaurant Menu Suggestions System" encompasses the development and integration of several key modules, each designed to enhance the operational efficiency of the restaurant and improve the customer dining experience.

Modules to be developed:

1) Customer Login and Registration:

Description: This module will facilitate secure authentication processes, allowing customers to create accounts and log in to the system.

Purpose: By maintaining individual customer profiles, the system can personalize the dining experience, track order history, and manage preferences effectively.

2) Menu Management (Add, Update, Delete Items):

Description: This feature enables administrative users to manage the menu dynamically, including adding new items, updating existing ones, or removing obsolete dishes.

Purpose: It ensures that the menu remains current and reflects the restaurant's offerings accurately, allowing for flexibility in response to inventory changes or seasonal specialties.

3) Sales Tracking and Analytics (Daily/Monthly/Yearly Sales):

Description: This module will record and analyze sales data over various time frames, providing comprehensive reports on revenue and sales trends.

Purpose: Access to detailed analytics supports informed decision-making, helping management identify popular items, peak sales periods, and areas needing improvement.

4) Cart and Order Management:

Description: Customers can select items to add to their virtual cart, modify quantities, and place orders through this interface.

Purpose: Streamlining the ordering process reduces wait times and minimizes errors, enhancing overall customer satisfaction.

5) Menu Recommendations:

Description: Utilizing sales data and customer preferences, this feature suggests dishes to customers, highlighting popular or complementary items.

Purpose: Personalized recommendations can enhance the dining experience, encourage exploration of the menu, and potentially increase average order value.

6) Inventory Management:

Description: Monitor and manage stock levels of ingredients and supplies.

Purpose: Ensure timely restocking, reduce waste, and maintain optimal inventory levels to meet customer demand.

7) Billing and Payment Processing:

Description: Generate bills and process payments through various methods, including cash, credit/debit cards, and digital wallets.

Purpose: Facilitate smooth and secure financial transactions, enhancing customer convenience and trust.

8) Feedback and Rating System:

Description: Allow customers to provide feedback and rate their dining experience.

Purpose: Gather valuable insights for continuous improvement and maintain high service standards.

1.4.2 Target User

1) Customers:
Access: They can browse the menu, add items to their cart, view cart, remove cart, view recommendations, provide feedback by rating food ordered , view order history, view profile, enjoy the discounts , register as a membership and confirm and place orders.

Benefit: A tailored and efficient ordering process enhances their dining experience.

2) Staff:
Access: Staff members can manage the menu (add , update and delete) , manager customer (customer membership (register or remove), update , view and delete customer information.), process order payment and view report and analytics.

Benefit: Streamlined order management facilitates smoother operations and reduces the likelihood of errors.

3) Managers:
Access: Managers have oversight of staff activities, can manage the menu (add , update and delete) , manager customer (customer membership (register or remove), update , view and delete customer information.), process order payment, view report and analytics and edit staff (add, update , view , delete).

Benefit: Comprehensive insights into operations support strategic planning and informed decision-making.

By defining and developing these modules, the system aims to create a cohesive platform that addresses the needs of all stakeholders, fostering an environment of efficiency, personalization, and data-driven management within the sushi restaurant.

1.5 Project Significance

The "Sushi Restaurant Menu Suggestions System" has everything in it to give the restaurant industry a flip by availing of advanced digital solutions to accommodate customer preferences and operationally efficient systems. In this age of technology, redefining the entire dining experience, this one introduces itself right at the cutting edge of innovation.

Increasing Customer Experience

Most importantly, personalization has been the hallmark of modern consumer expectations. Using average consumer behavior shown in preferences and order histories, the menu recommendations facilitate the treatment of diners as special people. It betters customer satisfaction and retention as guests are likely to return to places that usually cater to their tastes. Decision-making simplistically does wonder as well as making the overall experience as enjoyable as possible during feasts, mostly when one would have to choose from a myriad of options.

Operation Efficiency and Cost Management

For restaurant staff and management, the system is a way to automate the typical orders, sales tracking, and so on, thereby minimizing human error and streamlining operations so that they can focus on delivering great customer service. Additionally, integrating sales analytics provides managers with several real-time measures of performance metrics so that they can make data-driven decisions that have the potential to optimize inventory levels, waste reduction, and cost-effective management. Such efficiencies are very necessary for an industry that usually has thin margins and many operational challenges.

Driving Digital Transformation

It is quite a step towards digital transformation for the food service sector. Modernization is enabled by technology in operations of restaurants to help become competitive and meet changing demands from tech-savvy customers. Capabilities such as real-time order tracking and data analytics coincide with the industry's tendency in which digital tools are becoming staples for business success. Companies, for instance, like Deliveroo, have shown how such digital platforms can be used as "gateway tech" supporting their technology investments or internal improvements in operations.

Economic Resilience and Growth

Such advanced systems not only solve the current operational troubles, however, but they also prepare the restaurants for the future. Data and technology help restaurants to adapt speedily to changes in the environment, offer innovative services, and remain economically resilient. The analytics of the system could determine the trends and opportunities to refine business offerings proactively and strategies. This proactive method is important in a sector that is always under pressure from so many sides-the rising supply costs and changing behavior of consumers.

Thus, the "sushi restaurant menu suggestions system" isn't just an improvement but a total overhaul towards engagement with customers, a more refined efficiency in operations, and digital evolution within the restaurant space. Its implementation promises to deliver substantial benefits such that restaurants can survive in a continuously changing, competitive environment.

1.6 Gantt Chart of Project Activities

		Week														
No	Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Discussion of project proposal with supervisor.															
2	Project proposal submission after discussion and approval from supervisor.															
3	Analysis and design - problem statement, flowchart, pseudocode, data model, data dictionary, and input/output design.															
4	Project progress 1 - 25% project completion and project demonstration.															
5	Project progress 2 - 40-60% project completion and project demonstration.															
6	Project progress 3 - 80-100% project completion and project demonstration															
7	Project presentation and final report submission															

Figure 1 GANTT CHART

CHAPTER 2: ANALYSIS OF PROBLEM

2.1 Problem Decomposition Description

To create the "Sushi Restaurant Menu Suggestions System," it became imperative to breakdown the whole problem into several small bits. That was done in consideration of the many properties attached to restaurant management, and indeed correspond to the principles of computational thinking where complex problems are broken down into more manageable sub-problems that can thus be readily solved.

1. Customer Registration and Login:

Developing a well-secured yet user-friendly authentication module forms the crux of this requirement. This will allow customers to create accounts, login securely, and administer their profiles. Strong authentication protocols will safeguard user data, leading to a very seamless overall user experience.

2. Menu Management:

The organized function of the menus is vital for the functioning of a restaurant. This will aid administrators with the seamless adding, updating, or deleting of menu items. Such a dynamic menu management system-enabled restaurant will be able to adapt to the changes in seasons and customer preferences for certain foods as well as the availability of the ingredients in stock.

3. Sales Tracking and Analytics:

Through proper sales tracking, insight on how the restaurant performs in relation to the above can be observed. This module is designed to register all the transactions that happen in the restaurant and make it possible to create reports for sales taking place daily, monthly, and yearly. This data can also track changes in patterns or increase in demand and aid managerial decisions regarding the business.

4. Cart and Order Management:

Order-taking of customers becomes simple and comfortable. It gives the customer the option of selecting items, customizing the orders, and going through the cart before finalizing the order. For the staff, it provides an interface through which they can monitor orders coming in, the status of preparation, and the level of promptness in delivery.

5. Menu Recommendations:

Such a feature analyzes customers' buying habits and sales data to recommend menu items that could suit a customer's individual tastes or highlight some popular items within the menu. This type of system necessitates algorithms that have the capacity to process data in order to provide suitable recommendation suggestions.

Breakdown of the project into specific modules structures and compartmentalizes the development process. Each module focuses on the different aspect of restaurant work which thereby forming a complete system improving efficiency and customer satisfaction. The modular system has much simplification in development and is even straightforwardly extensible and maintainable, as every module can be updated or integrated to expand its capability without affecting the overall system.

2.2 Structured Chart

Visual representation to illustrate the hierarchical structure of Sushi restaurant menu recommendation order system, break down the system into its lowest functional modules.

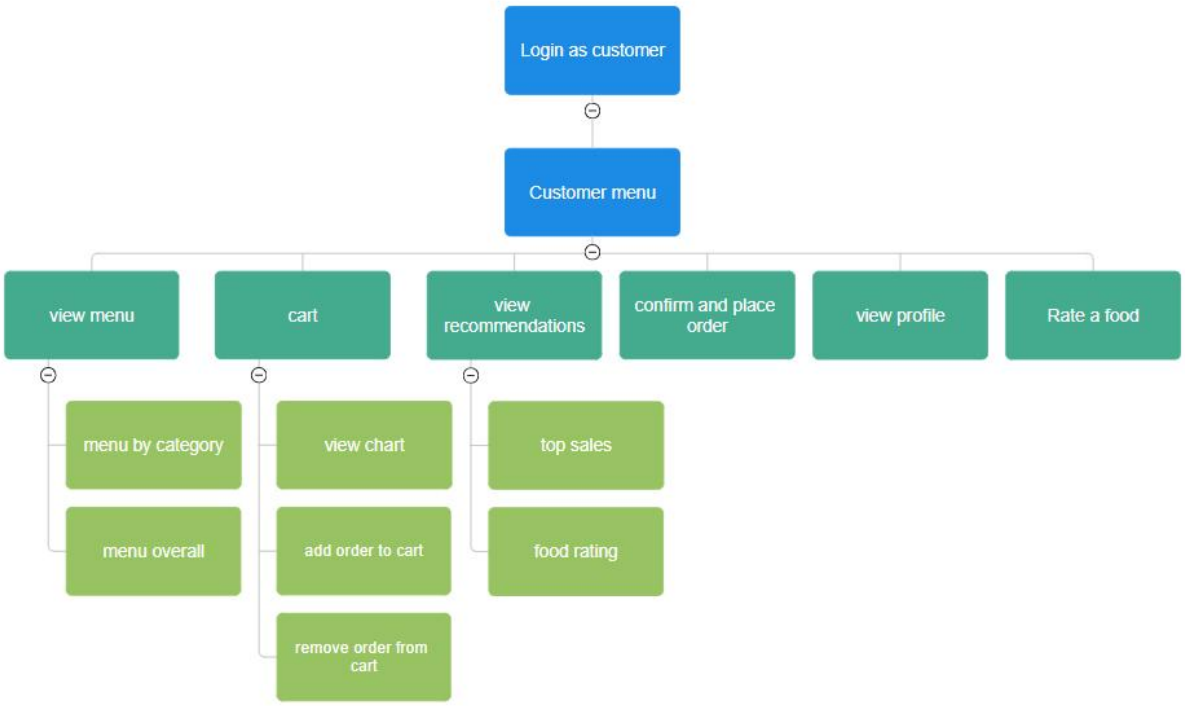


Figure 2: STRUCTURED CHART CUSTOMER

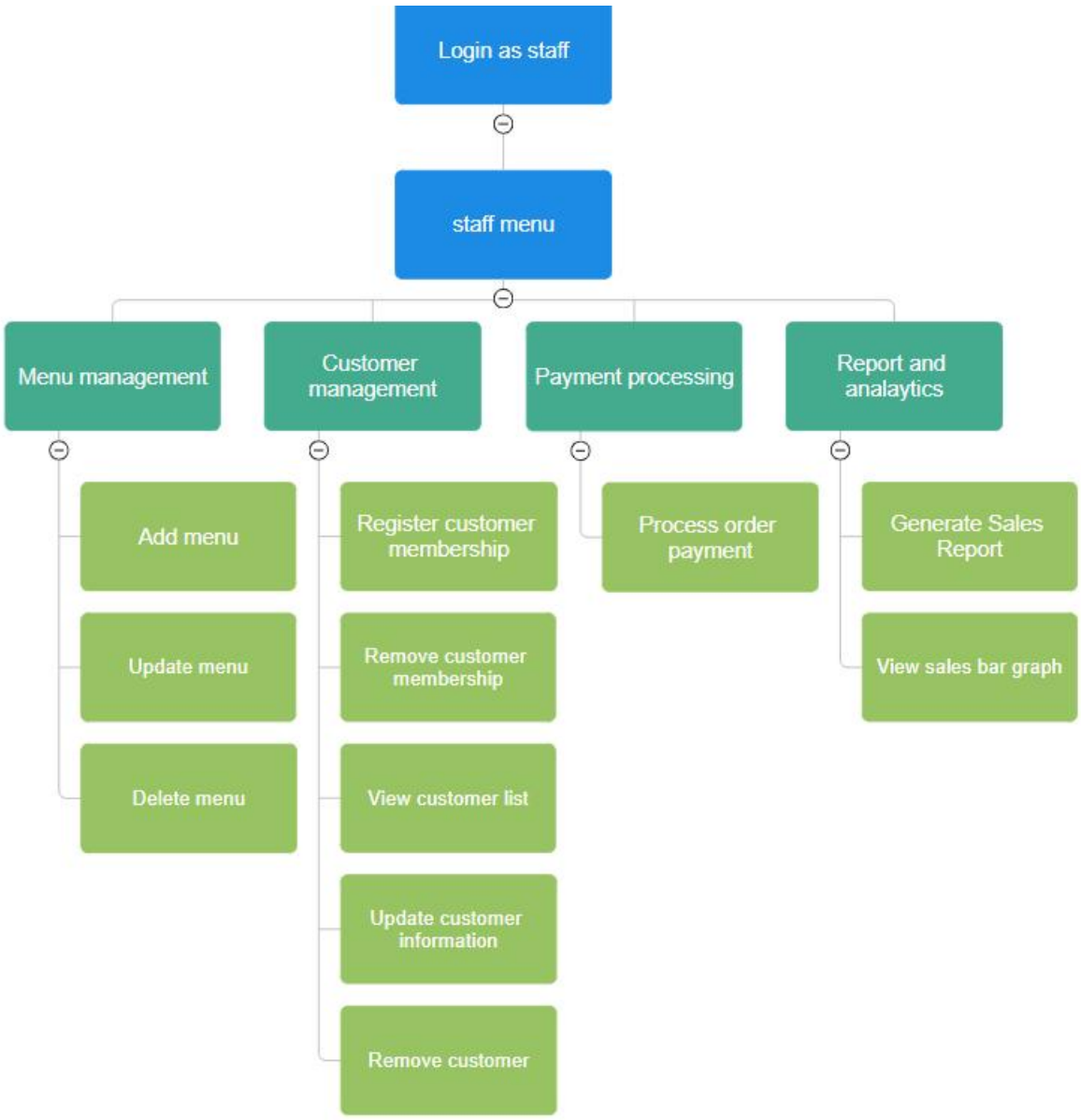


Figure 3: STRUCTURED CHART STAFF

CHAPTER 3: DESIGN

3.1 Flowchart

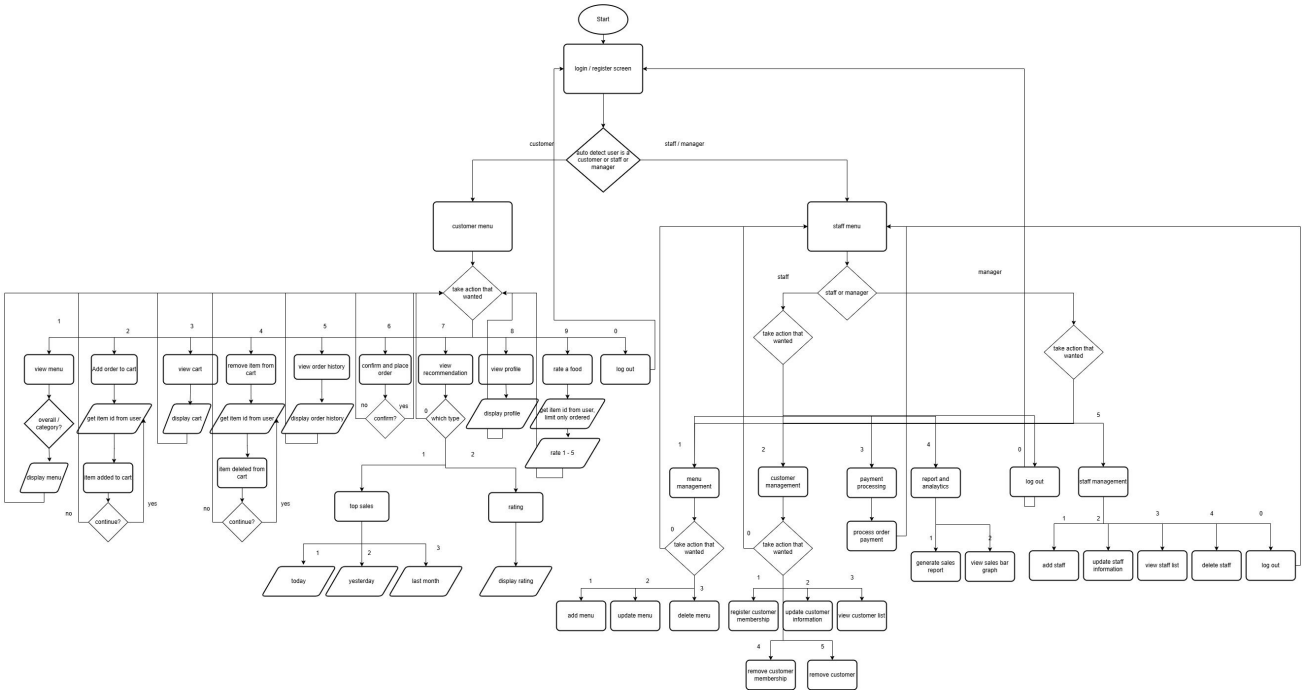


Figure 3: FLOWCHART OVERALL

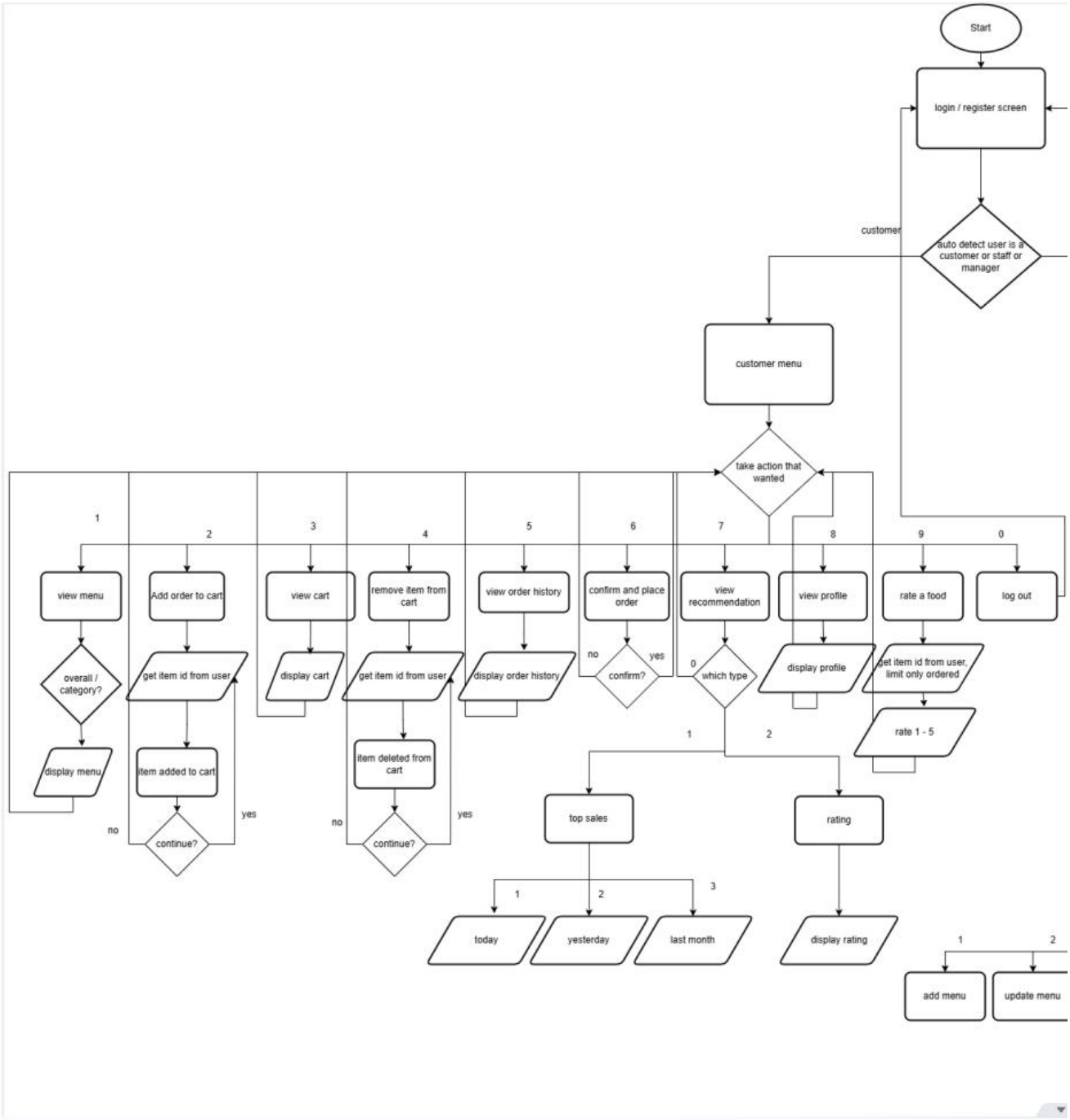


Figure 4 : FLOWCHART CUSTOMER

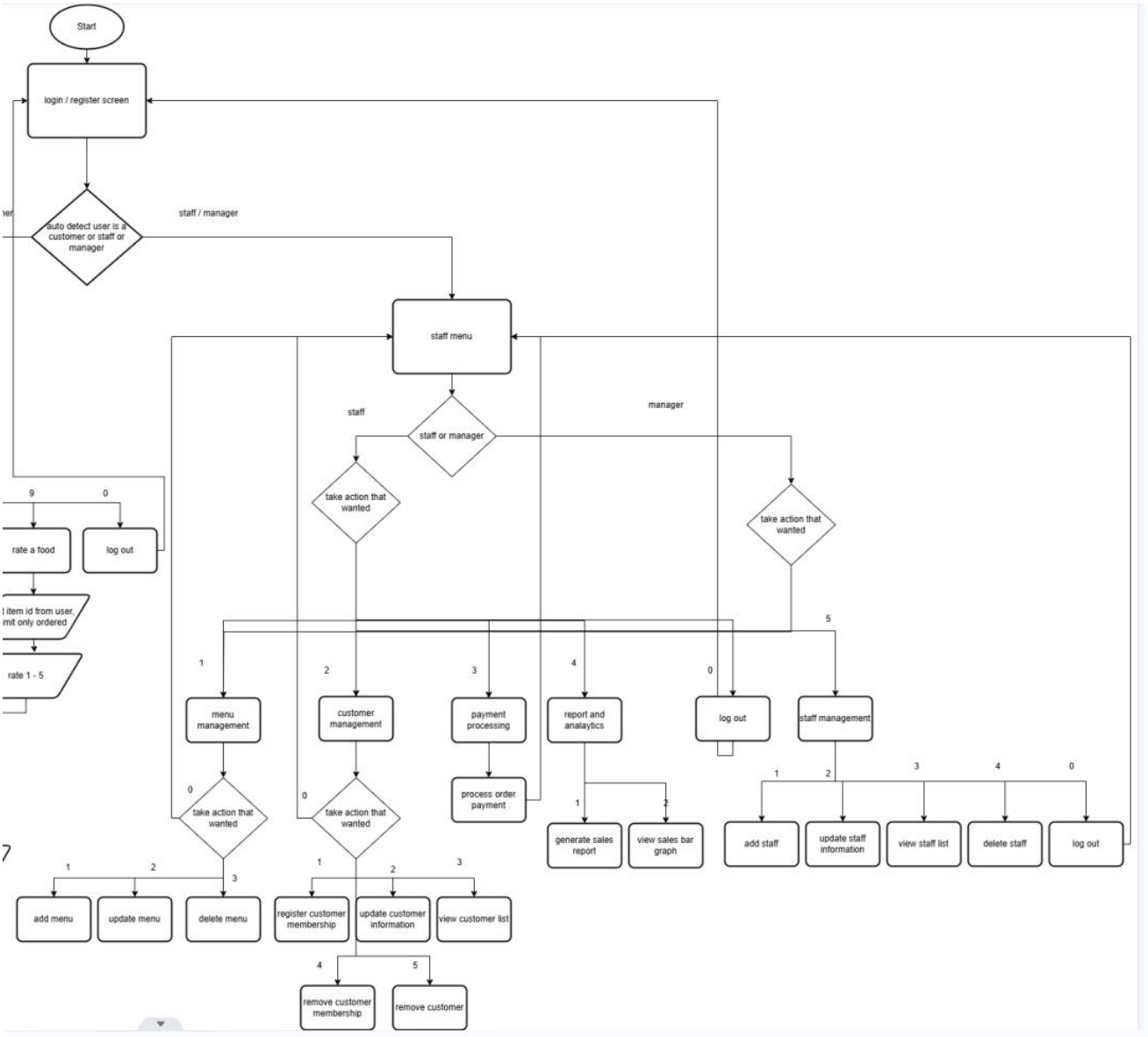


Figure 5 : FLOWCHART STAFF

3.2 ERD

ER Diagram used to produce high-quality database design to use in database creation, management, and maintenance.

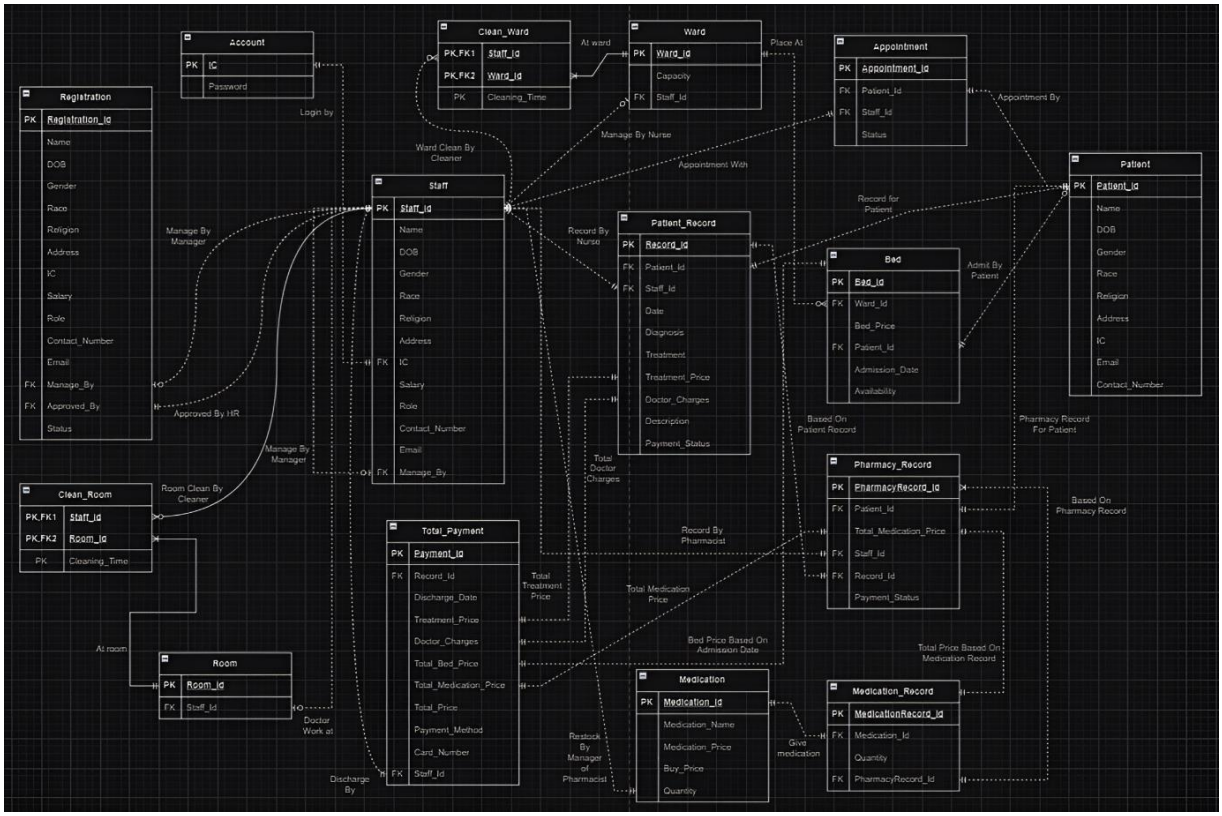


Figure 6: ERD

3.3 Data Dictionary

Provides information about entities, relationships, and attributes that are present in the system model.

Table 1: Data dictionary for customer

Field	Type	Null	Key	Default	description
customer_id	int	NO	PRI	NULL	
customer_name	varchar(45)	NO		NULL	
customer_email	varchar(45)	NO		NULL	
customer_password	varchar(45)	NO		NULL	
customer_phone_number	varchar(45)	YES		NULL	
customer_registration_date	varchar(45)	YES		NULL	
is_member	tinyint(1)	YES		0	Check user is a member or not
membership_points	int	YES		0	Record member_points
customer_birthday	date	YES		NULL	

Table 2: Data dictionary for food_ratings

Field	Type	Null	Key	Default	Extra	description
rating_id	int	NO	PRI	NULL	auto_increment	PK
customer_id	int	YES	MUL	NULL		FK
menu_item_id	int	YES	MUL	NULL		FK
rating	int	YES		NULL		Record the rating point
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED	The rating is created by customer at time
updated_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP	Customer update their rating at time

Table 3: Data dictionary for menu_item

Field	Type	Null	Key	Default	description
menu_item_id	int	NO	PRI	NULL	PK
item_name	varchar(45)	YES		NULL	
item_price	double	YES		NULL	
category	varchar(45)	YES		NULL	Item's category
total_rating	int	YES		0	Rating of the menu_item
rating_count	int	YES		0	How many rating of the manu_item

Table 4: Data dictionary for order

Field	Type	Null	Key	Default	description
order_id	int	NO	PRI	NULL	auto_increment
order_date	varchar(45)	NO		NULL	
total_amount	double	NO		NULL	
customer_id	varchar(45)	NO		NULL	
is_paid	tinyint(1)	NO		0	Record customer already paid or not

Table 5: Data dictionary for order_details

Field	Type	Null	Key	Default	Description
order_details_id	int	NO	PRI	NULL	auto_increment
order_id	int	NO	MUL	NULL	
menu_item_id	int	NO		NULL	
quantity	int	NO		NULL	Order_quantity
item_price	double	NO		NULL	

Table 6: Data dictionary for sales

Field	Type	Null	Key	Default	Description
sales_id	int	NO	PRI	NULL	auto_increment
sale_date	varchar(45)	NO		NULL	
quantity_sales	int	NO		NULL	
daily_sales	double	NO		0	
monthly_sales	double	NO		0	
yearly_sales	double	NO		0	
revenue	double	NO		0	Revenue sales
menu_item_id	varchar(45)	YES	MUL	NULL	
order_id	int	YES	MUL	NULL	

Table 7: Data dictionary for staff

Field	Type	Null	Key	Default	Extra
staff_id	int	NO	PRI	NULL	auto_increment
staff_name	varchar(45)	YES		NULL	
staff_email	varchar(45)	YES		NULL	
staff_password	varchar(45)	YES		NULL	
staff_phone_number	varchar(45)	YES		NULL	
staff_role	varchar(45)	YES		NULL	
staff_hire_date	date	YES		NULL	

3.4 Interface Design

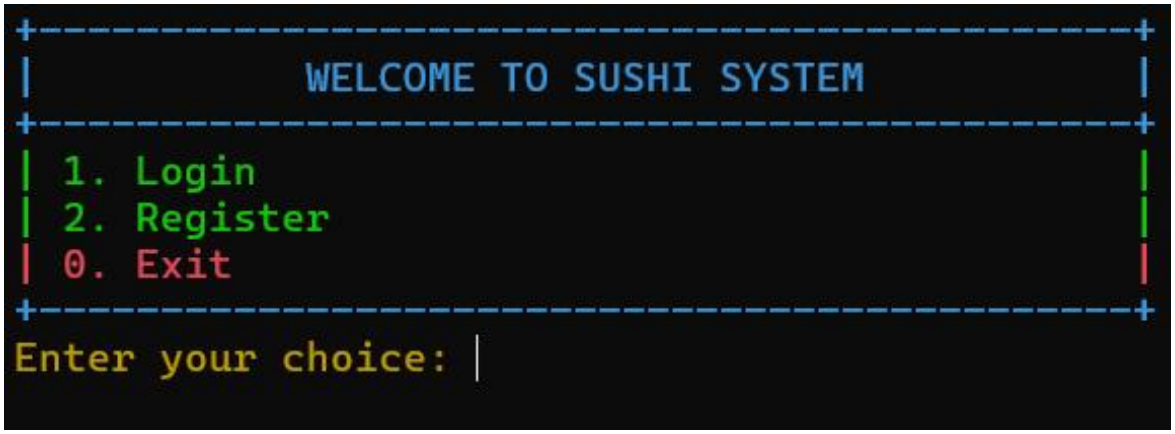


Figure 7: MAIN_MENU

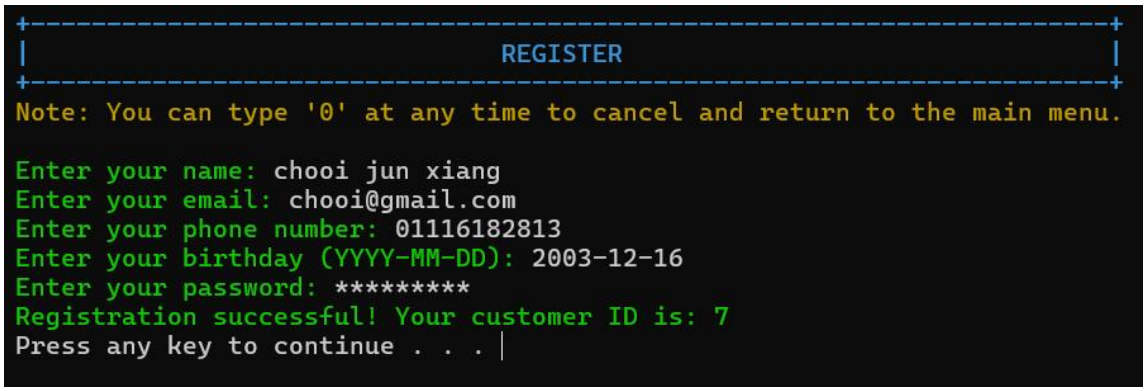


Figure 8: MAIN_REGISTER_MENU

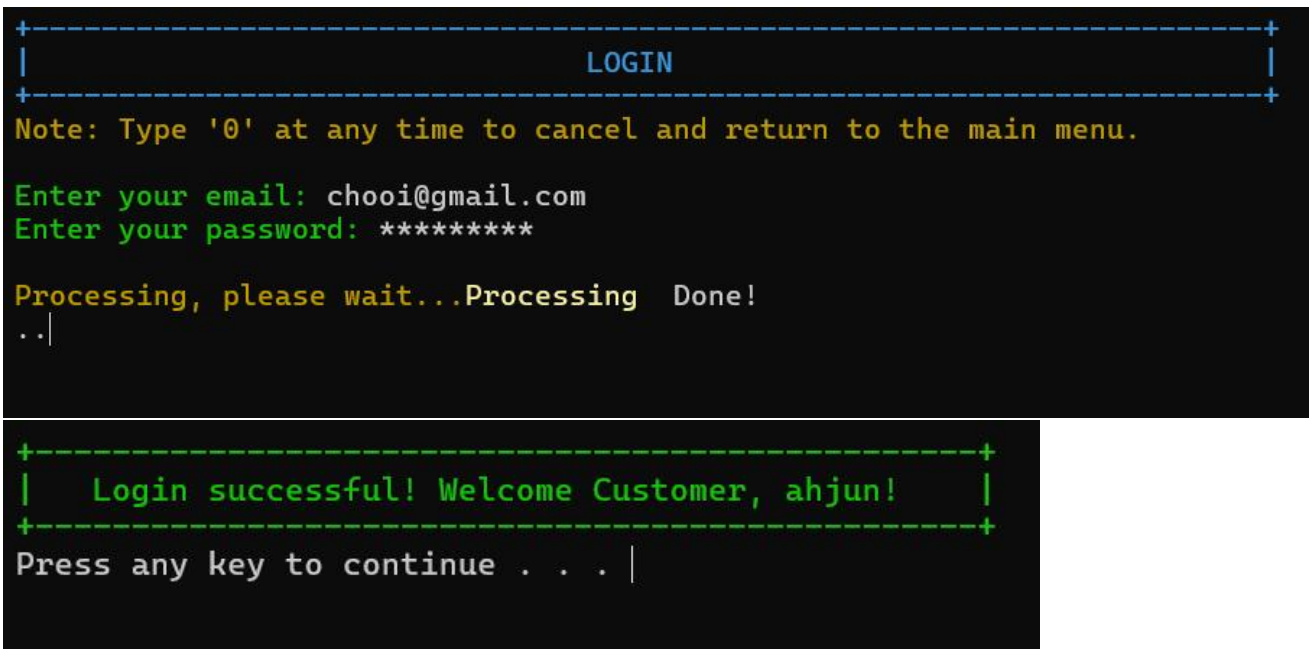


Figure 9: MAIN_LOGIN_MENU_WITH_ANIMATION

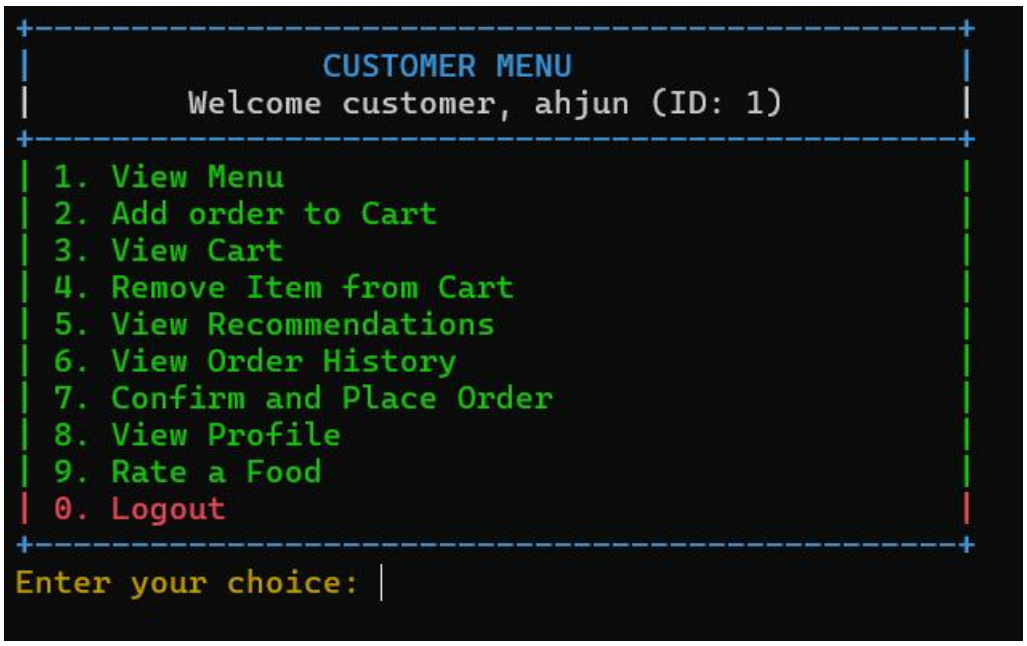


FIGURE 10 : CUSTOMER MENU

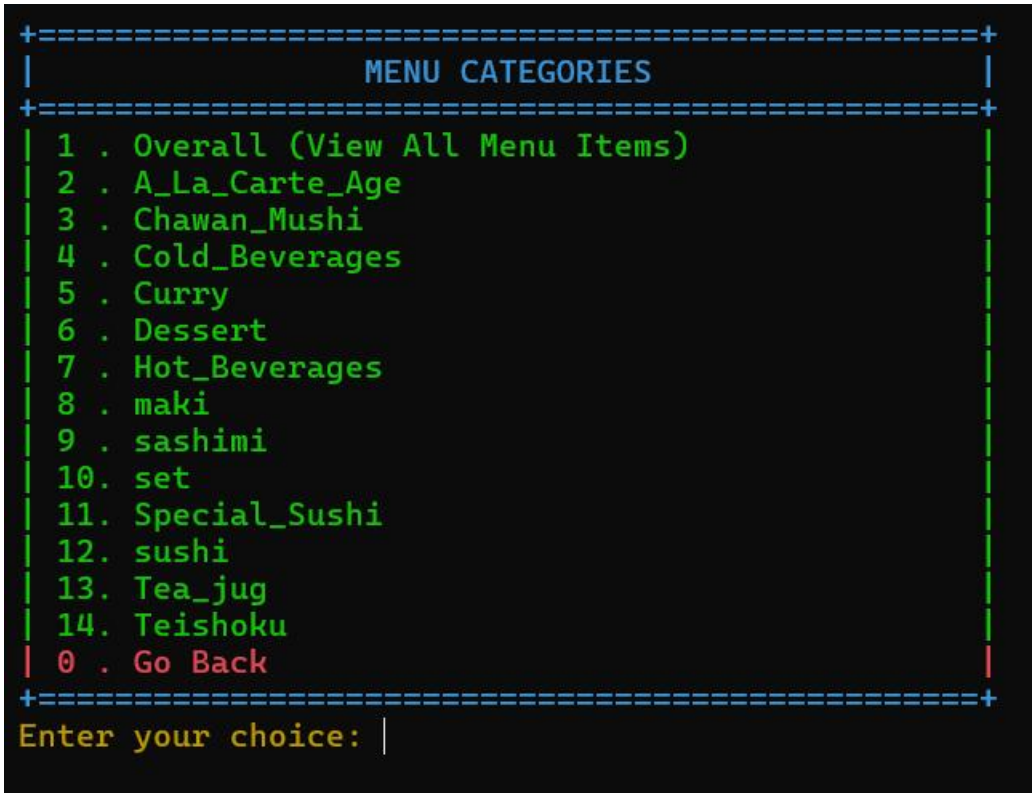
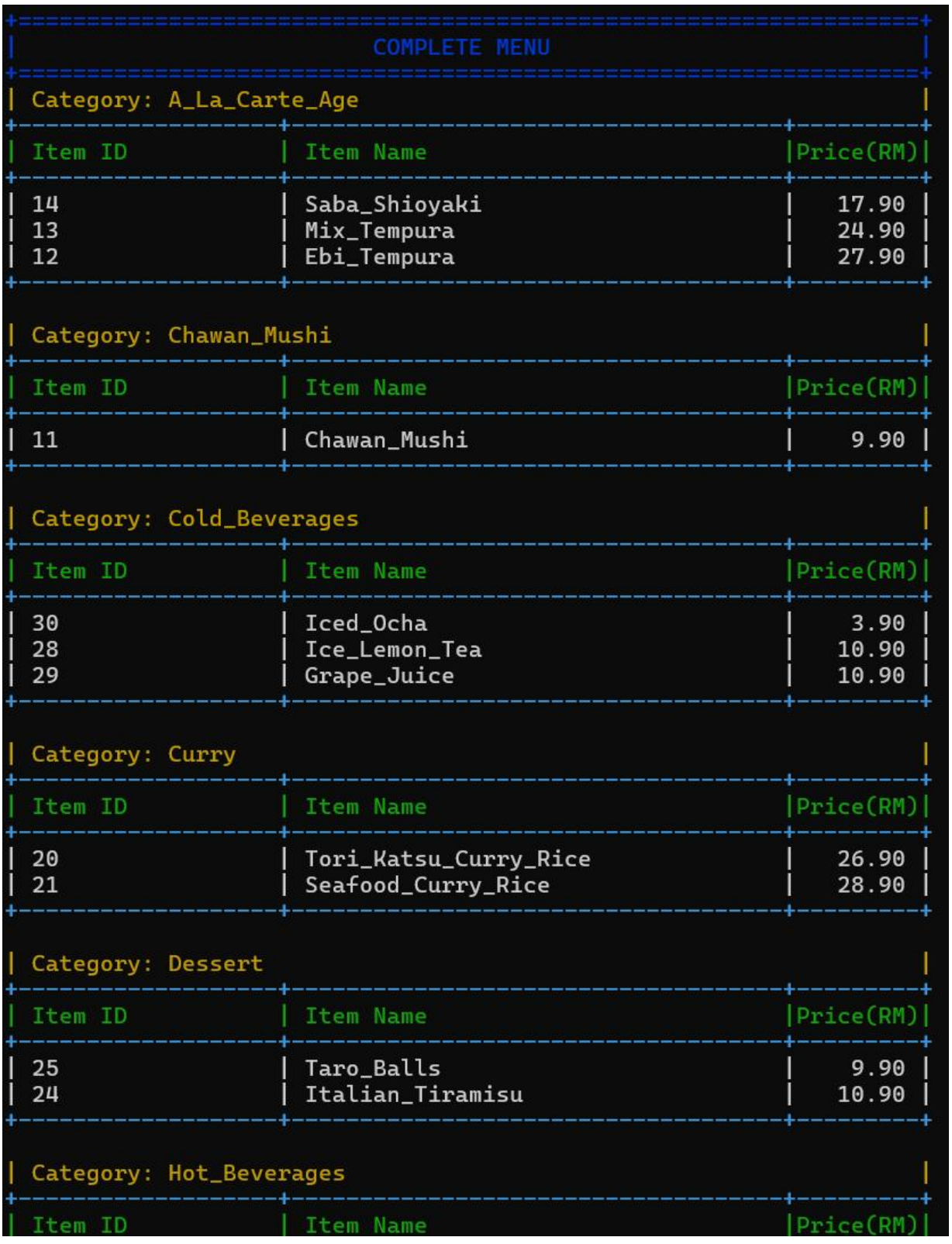


FIGURE 11 : VIEW MENU CAN CHOOSE WHAT CATEGORIES TO VIEW



COMPLETE MENU

Category: A_La_Carte_Age

Item ID	Item Name	Price(RM)
14	Saba_Shioyaki	17.90
13	Mix_Tempura	24.90
12	Ebi_Tempura	27.90

Category: Chawan_Mushi

Item ID	Item Name	Price(RM)
11	Chawan_Mushi	9.90

Category: Cold_Beverages

Item ID	Item Name	Price(RM)
30	Iced_Ocha	3.90
28	Ice_Lemon_Tea	10.90
29	Grape_Juice	10.90

Category: Curry

Item ID	Item Name	Price(RM)
20	Tori_Katsu_Curry_Rice	26.90
21	Seafood_Curry_Rice	28.90

Category: Dessert

Item ID	Item Name	Price(RM)
25	Taro_Balls	9.90
24	Italian_Tiramisu	10.90

Category: Hot_Beverages

Item ID	Item Name	Price(RM)

FIGURE 12 : OVERALL IN VIEW MENU

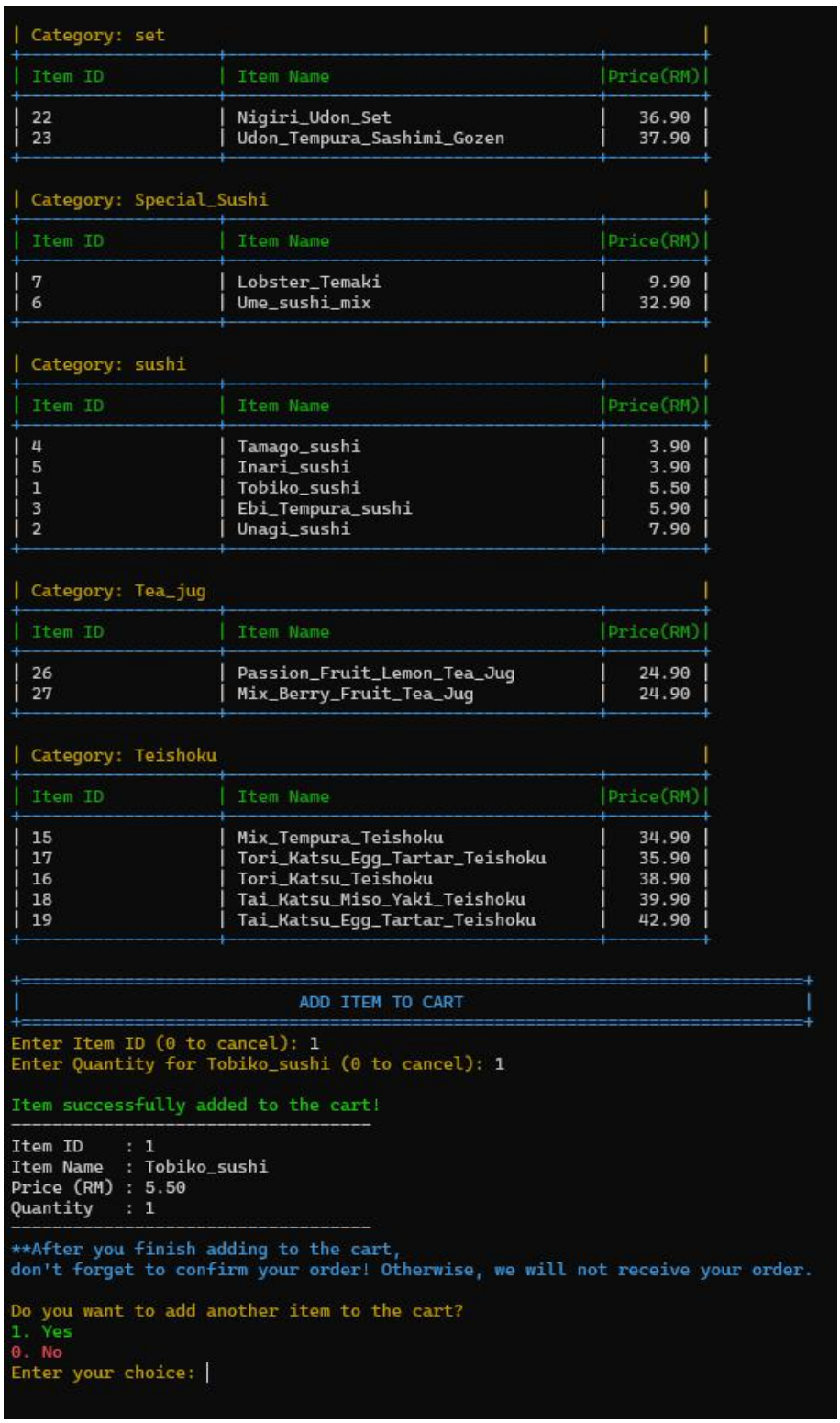


FIGURE 13 : ADD ORDER TO CART

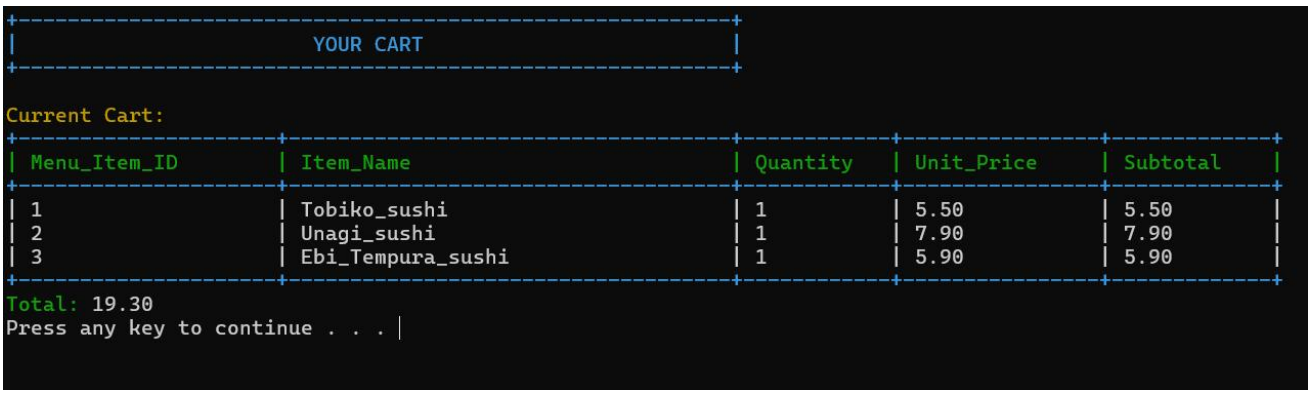


FIGURE 14 VIEW CART

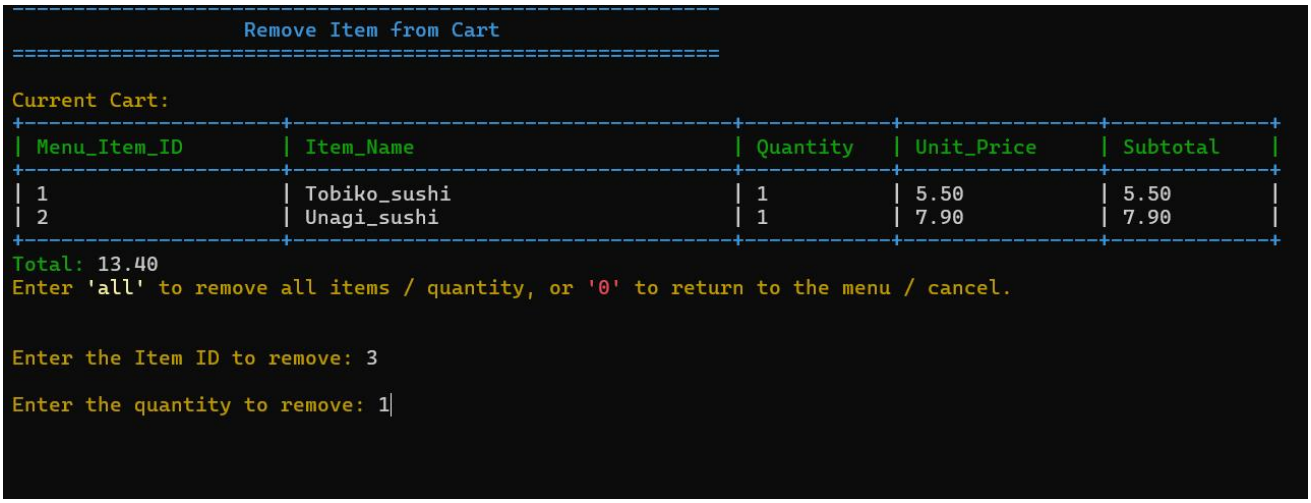


FIGURE 15 : REMOVE ITEM FROM CART PART 1

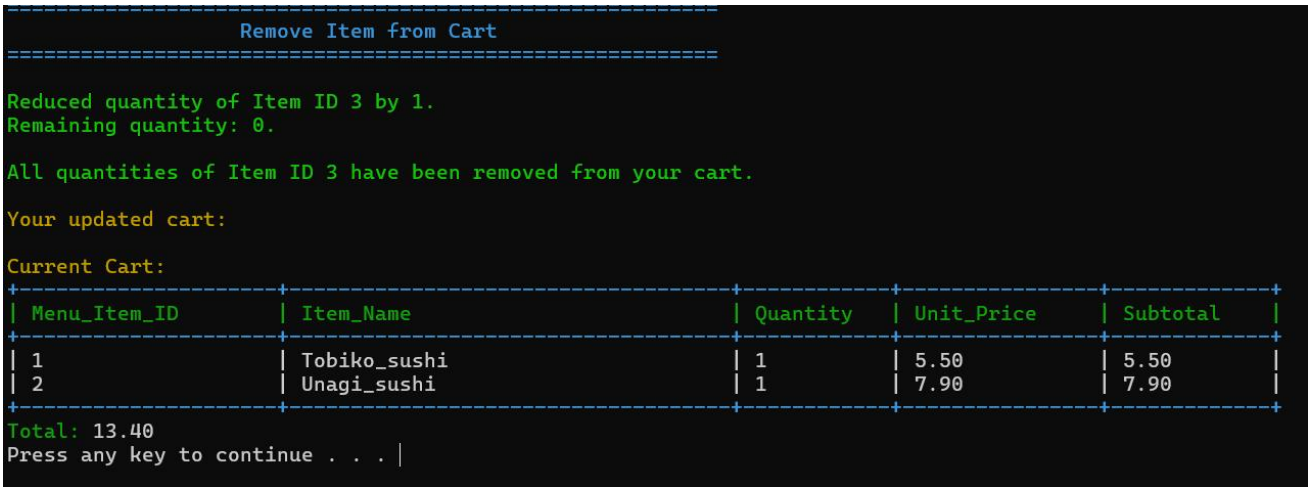


FIGURE 16 : REMOVE ITEM FROM CART PART 2

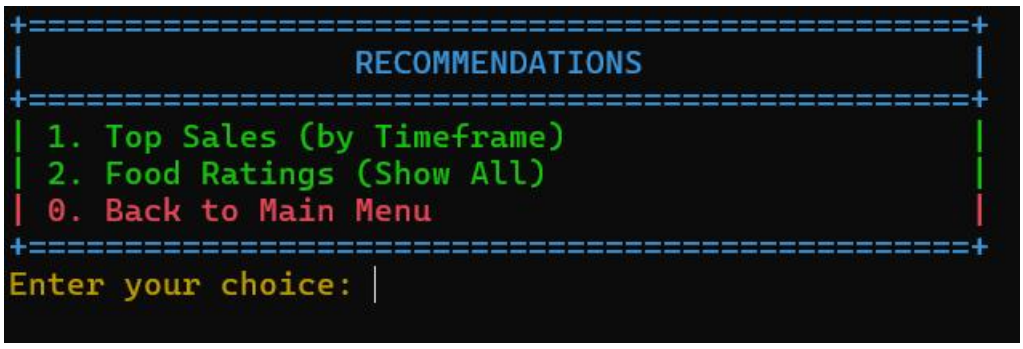


FIGURE 17: VIEW RECOMMENDATIONS

```
+=====================================================+
|                      TOP SALES                      |
+=====================================================+
Choose a Timeframe:
1. Today
2. Yesterday
3. Last Month
0. Go Back
Enter your choice:
```

FIGURE 18 : TOP SALES RECOMMENDATIONS

```
+=============================================================================================+
|                                      FOODS BY RATINGS                                       |
+=============================================================================================+

All Foods by Ratings:
+-----------------+----------------------------------------+---------------+---------------+
| Item ID         | Item Name                              | Average Rating| Rating Count  |
+-----------------+----------------------------------------+---------------+---------------+
| 1               | Tobiko_sushi                           | 5.00          | 3             |
| 14              | Saba_Shioyaki                          | 5.00          | 1             |
| 2               | Unagi_sushi                            | 4.50          | 2             |
| 7               | Lobster_Temaki                         | 4.00          | 1             |
| 13              | Mix_Tempura                            | 4.00          | 1             |
| 26              | Passion_Fruit_Lemon_Tea_Jug            | 4.00          | 1             |
| 9               | Sake_Belly_Sashimi                     | 3.00          | 1             |
| 5               | Inari_sushi                            | 3.00          | 1             |
| 6               | Ume_sushi_mix                          | 3.00          | 1             |
| 4               | Tamago_sushi                           | 0.00          | 0             |
| 8               | California_Maki                        | 0.00          | 0             |
| 10              | Sake_Sashimi                           | 0.00          | 0             |
| 11              | Chawan_Mushi                           | 0.00          | 0             |
| 12              | Ebi_Tempura                            | 0.00          | 0             |
| 15              | Mix_Tempura_Teishoku                   | 0.00          | 0             |
| 16              | Tori_Katsu_Teishoku                    | 0.00          | 0             |
| 17              | Tori_Katsu_Egg_Tartar_Teishoku         | 0.00          | 0             |
| 18              | Tai_Katsu_Miso_Yaki_Teishoku           | 0.00          | 0             |
| 19              | Tai_Katsu_Egg_Tartar_Teishoku          | 0.00          | 0             |
| 20              | Tori_Katsu_Curry_Rice                  | 0.00          | 0             |
| 21              | Seafood_Curry_Rice                     | 0.00          | 0             |
| 22              | Nigiri_Udon_Set                        | 0.00          | 0             |
| 23              | Udon_Tempura_Sashimi_Gozen             | 0.00          | 0             |
| 24              | Italian_Tiramisu                       | 0.00          | 0             |
| 25              | Taro_Balls                             | 0.00          | 0             |
| 3               | Ebi_Tempura_sushi                      | 0.00          | 0             |
| 27              | Mix_Berry_Fruit_Tea_Jug                | 0.00          | 0             |
| 28              | Ice_Lemon_Tea                          | 0.00          | 0             |
| 29              | Grape_Juice                            | 0.00          | 0             |
| 30              | Iced_Ocha                              | 0.00          | 0             |
| 31              | Ocha(cup)                              | 0.00          | 0             |
| 32              | Ocha(pot)                              | 0.00          | 0             |
| 33              | Hot_Citron                             | 0.00          | 0             |
+-----------------+----------------------------------------+---------------+---------------+
Press any key to continue . . .
```

FIGURE 19 : FOOD RATING RECOMMENDATIONS

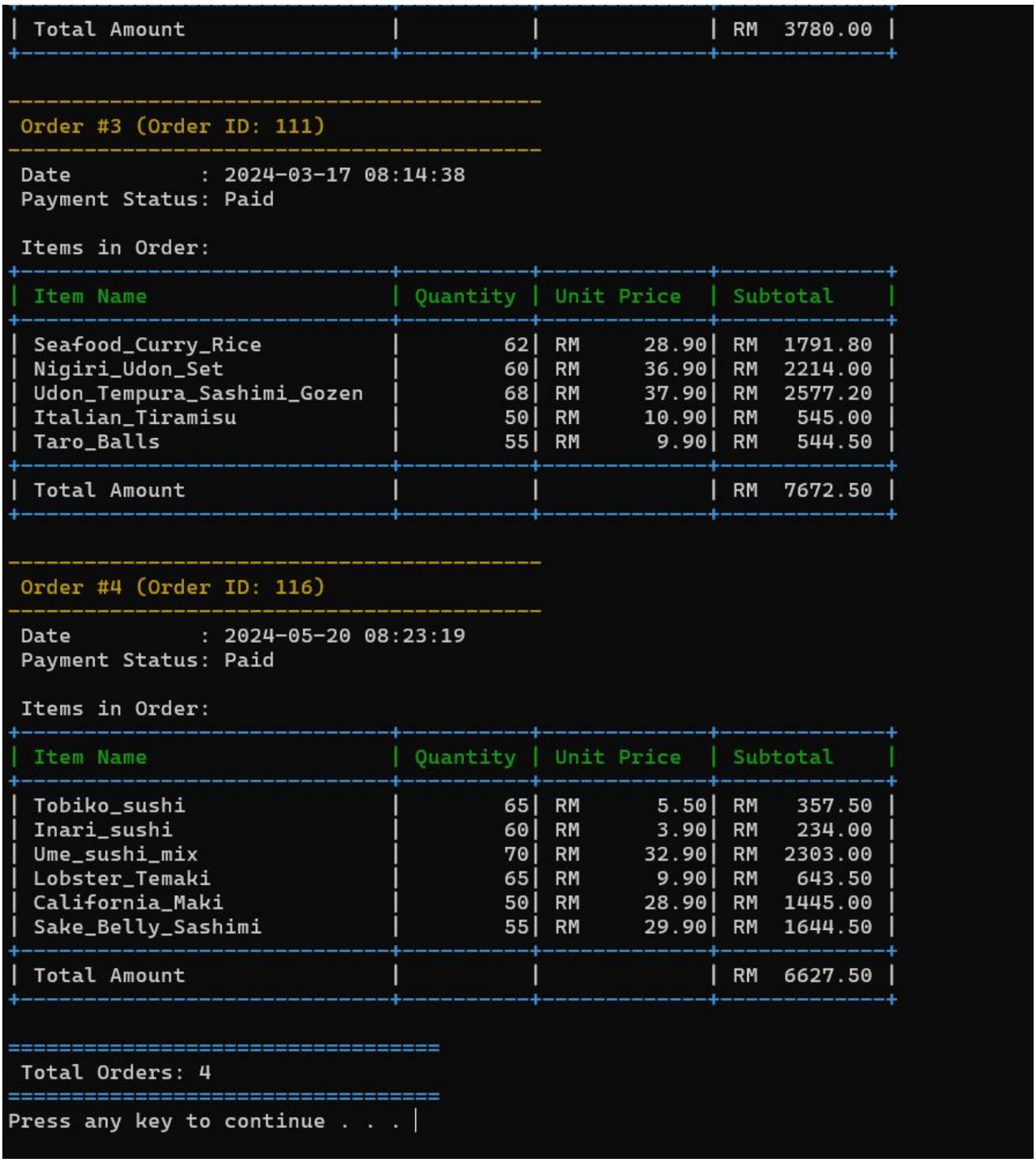


FIGURE 20 : VIEW ORDER HISTORY

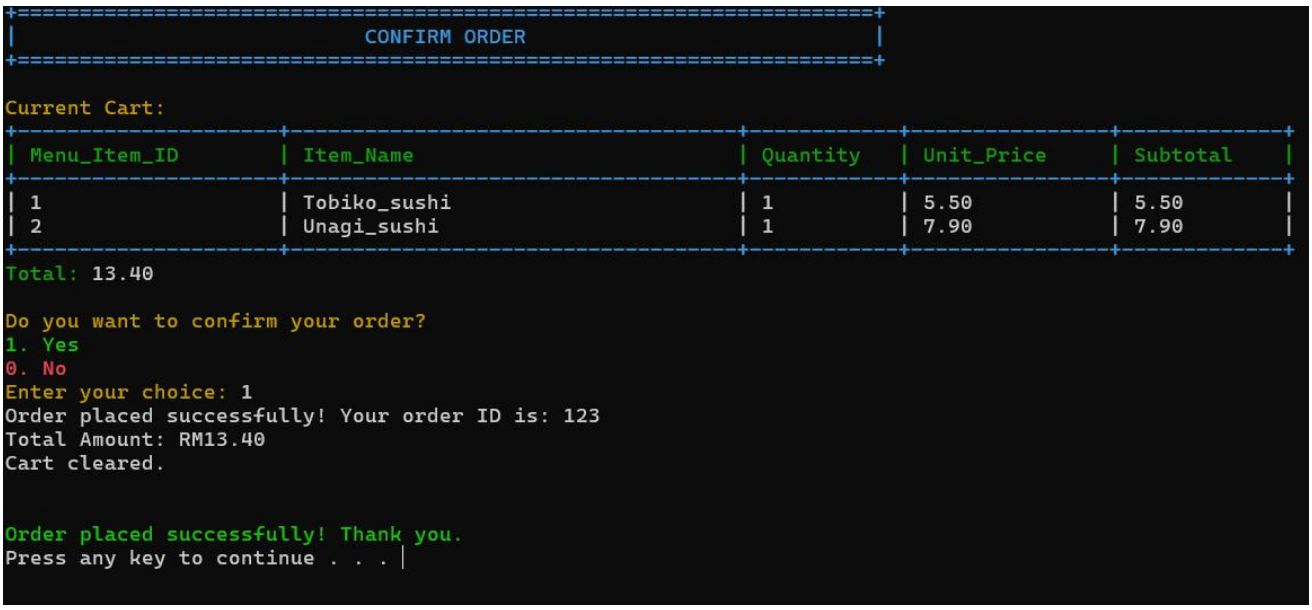


FIGURE 21 : CONFIRM AND PLACE ORDER

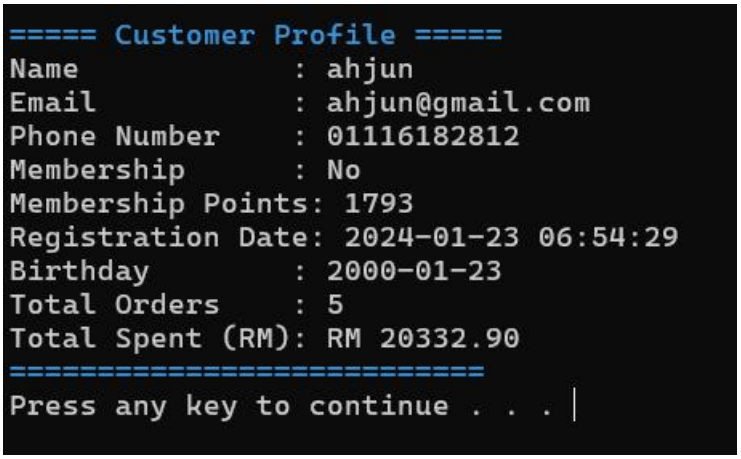


FIGURE 22 : VIEW PROFILE

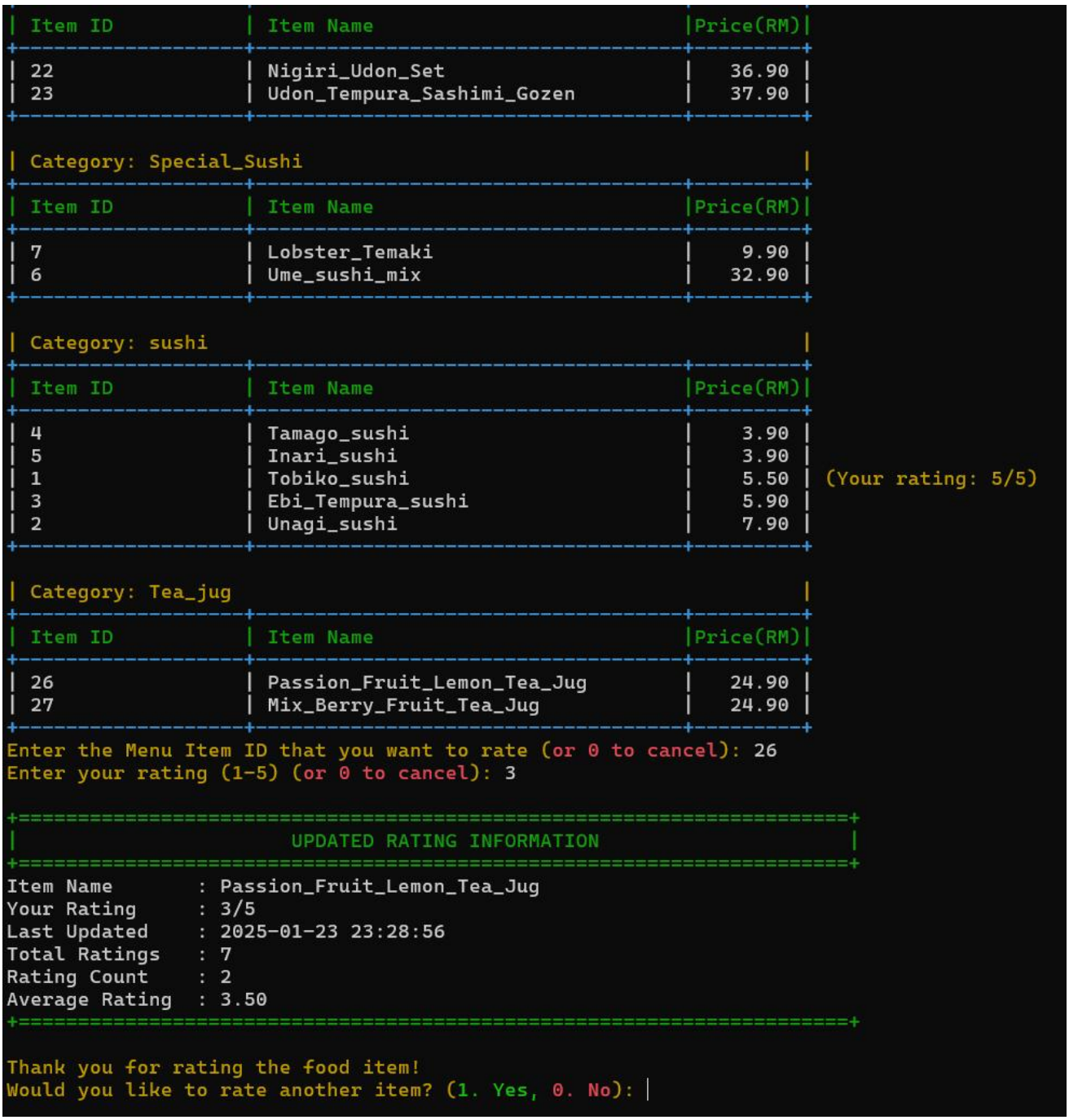


FIGURE 23 : RATING FOOD

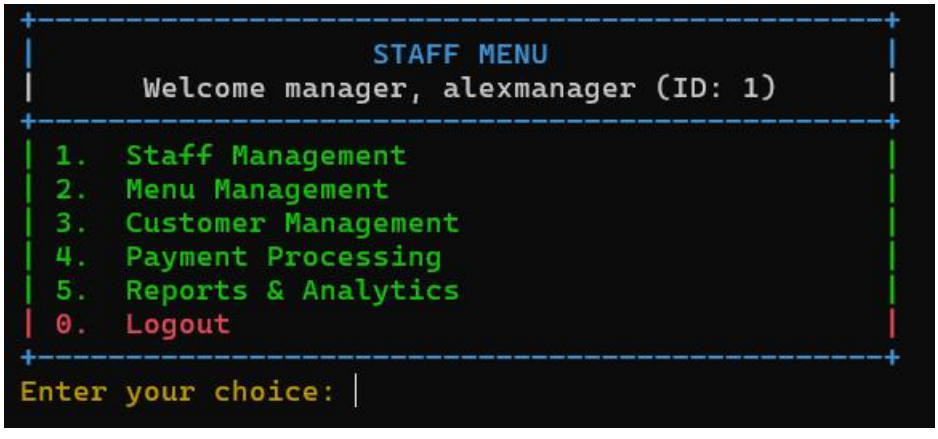


FIGURE 24 : STAFF MENU

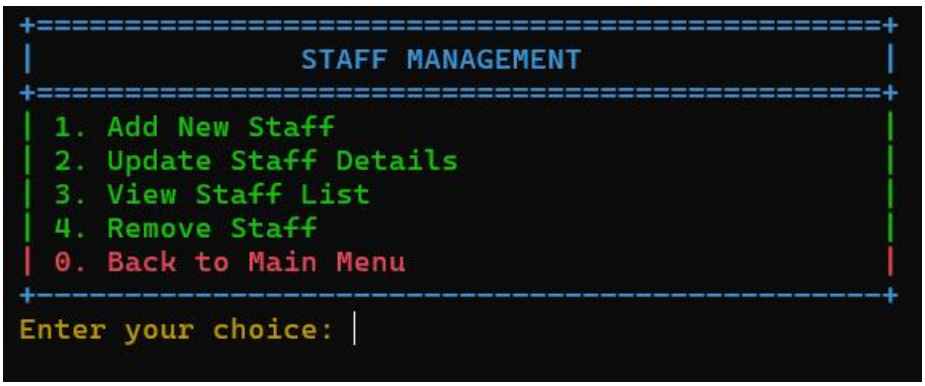


FIGURE 25 : STAFF MANAGEMENT (ADD,UPDATE,VIEW AND REMOVE)

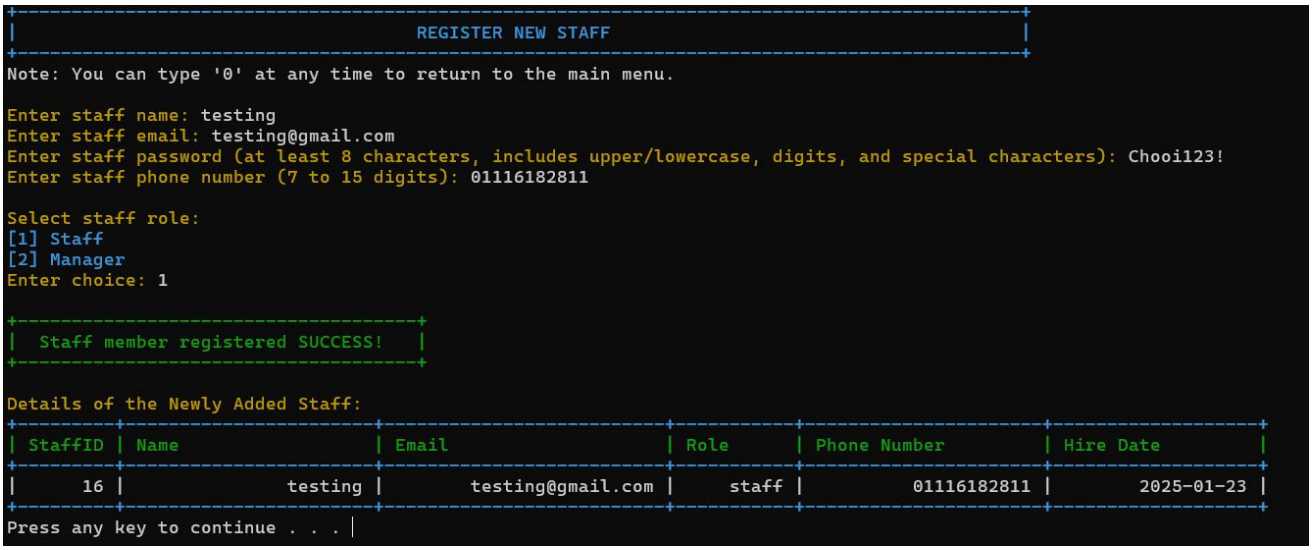


FIGURE 26 : ADD NEW STAFF

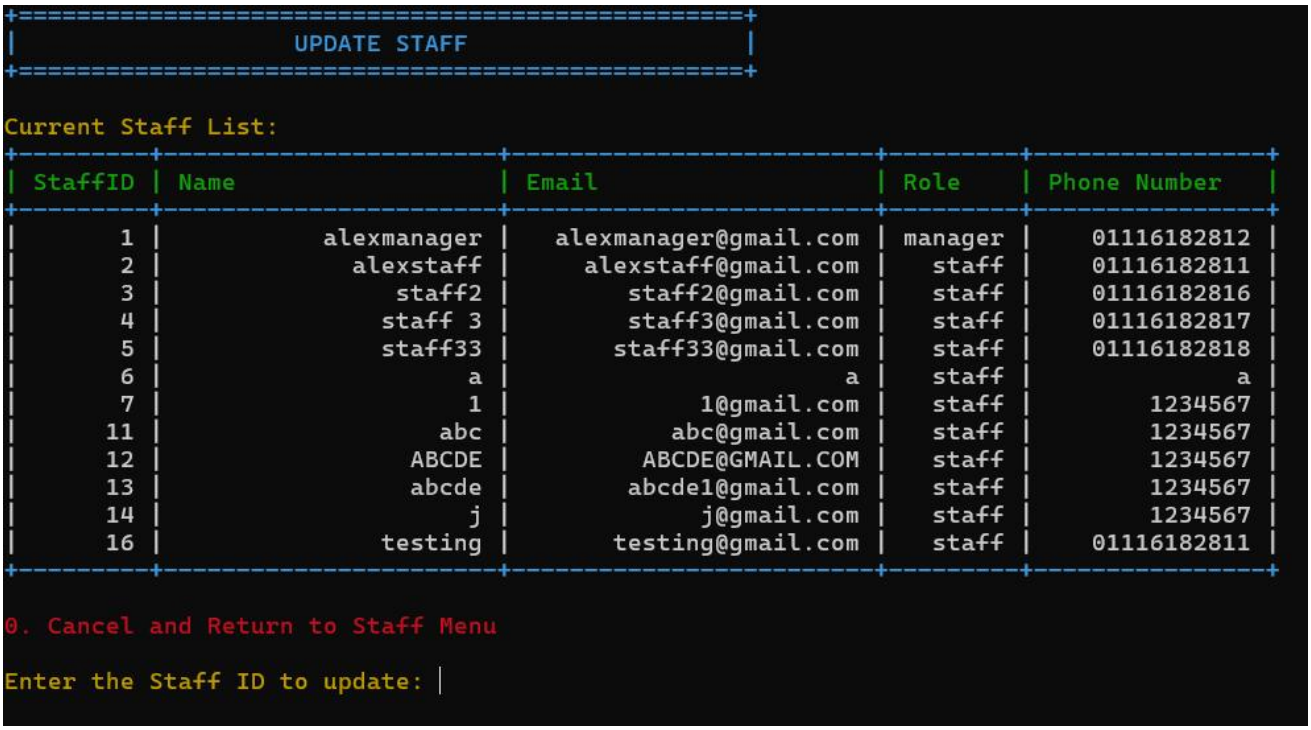


FIGURE 27 : UPDATE STAFF (PART 1)

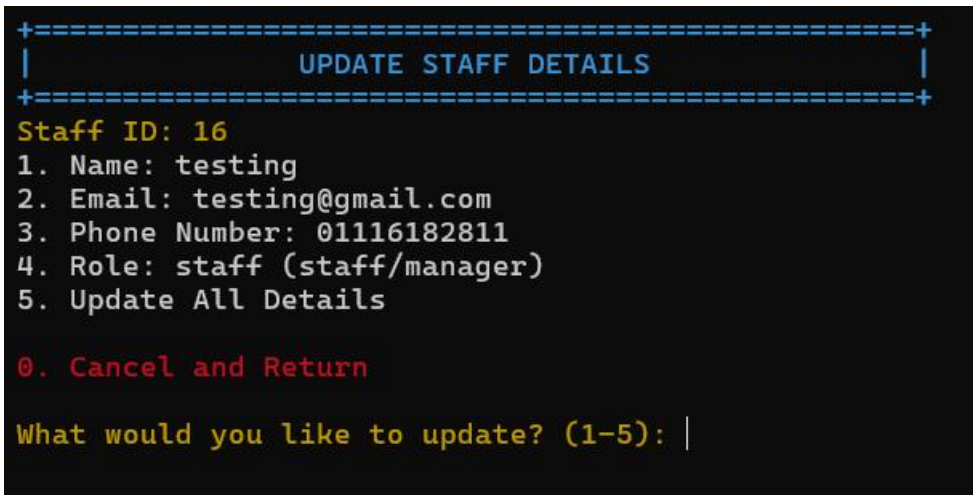


FIGURE 28 : UPDATE STAFF (PART 2)

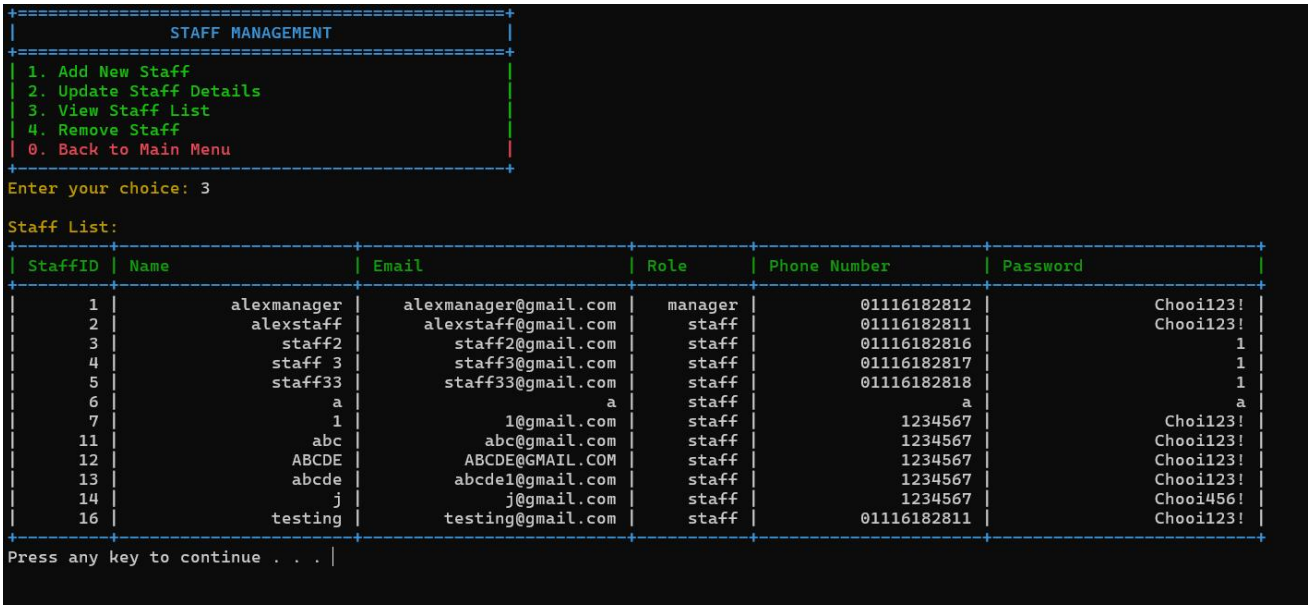


FIGURE 29 : VIEW STAFF

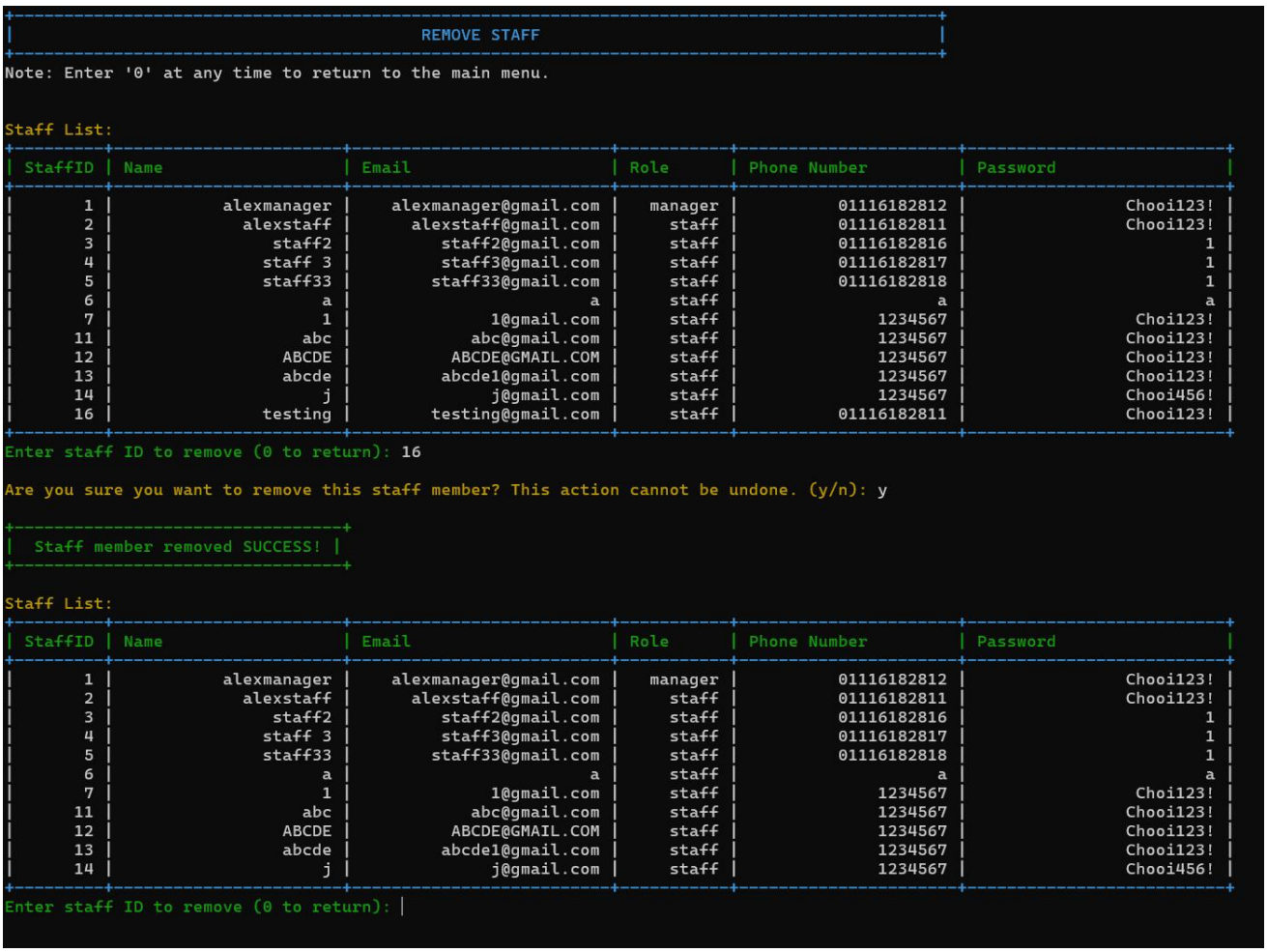


FIGURE 30 : REMOVE STAFF

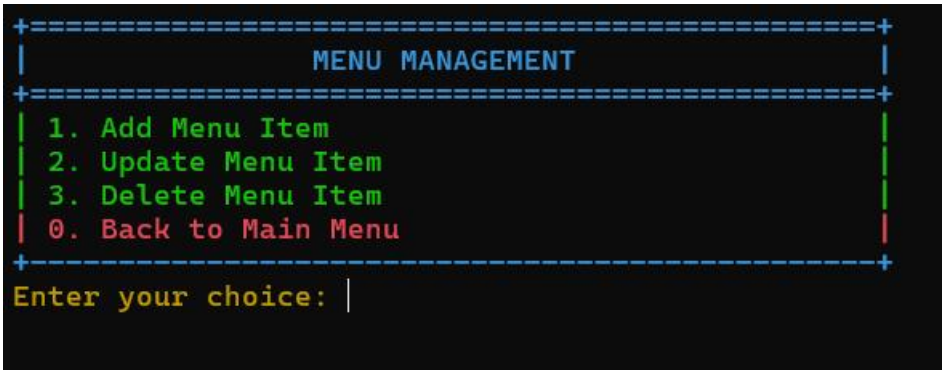


FIGURE 31 : MENU MANAGEMENT (ADD , UPDATE AND DELETE)

FIGURE 32 : ADD ITEM (PART 1)

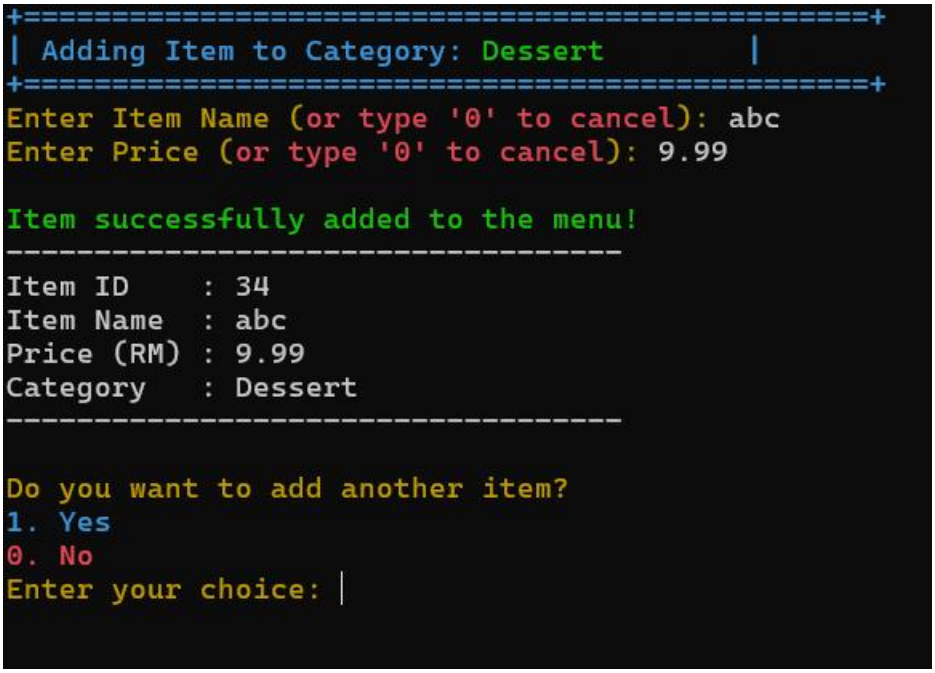


FIGURE 33 ADD ITEM (PART 2)

FIGURE 34 : UPDATE MENU (PART 1)

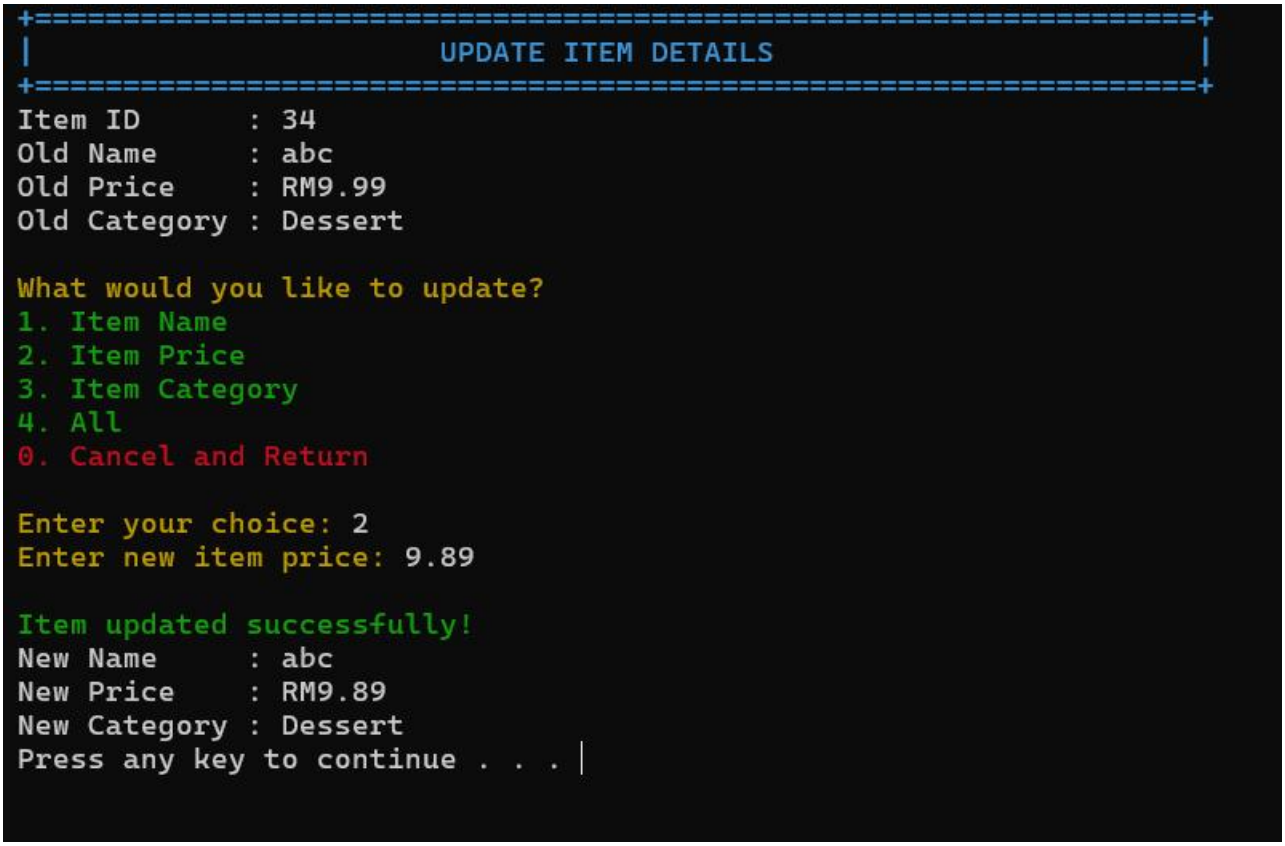


FIGURE 35 : UPDATE MENU (PART 2)

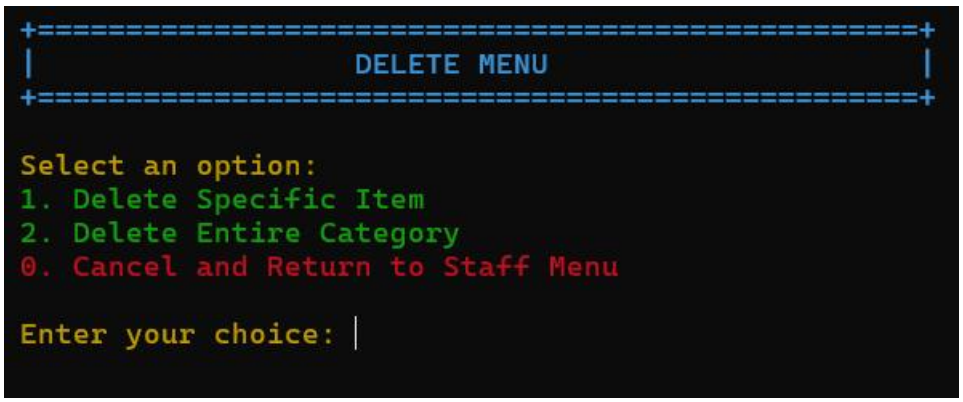


FIGURE 36 : DELETE MENU (PART 1)

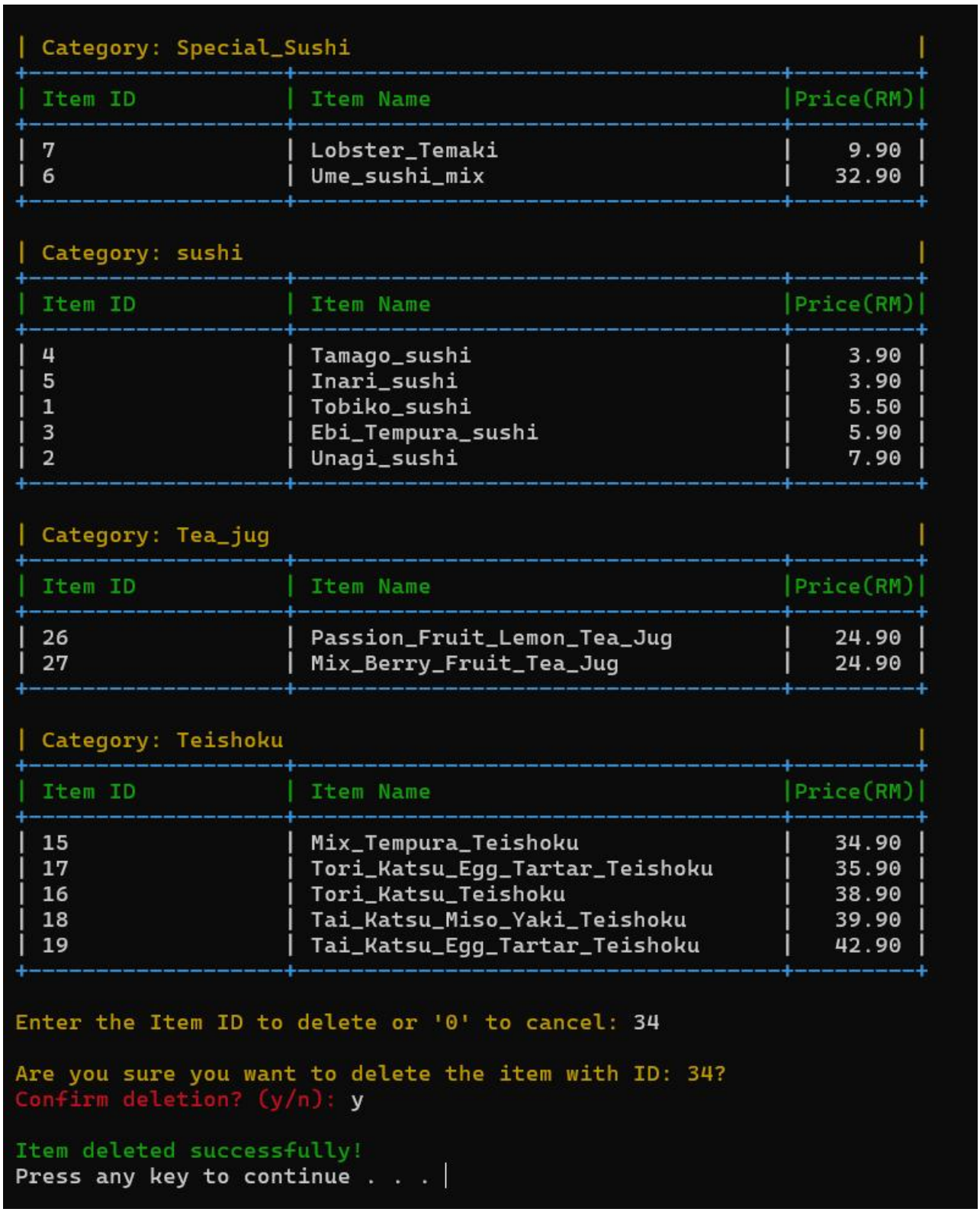


FIGURE 37 : DELETE MENU (PART 2)

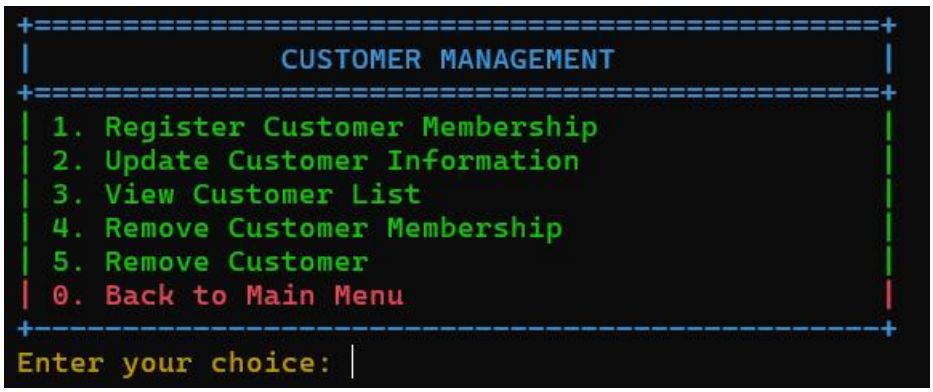


FIGURE 38 : CUSTOMER MANAGEMENT (REGISTER / REMOVE MEMBERSHIP, UPDATE , VIEW AND REMOVE CUSTOMER INFORMATION.)

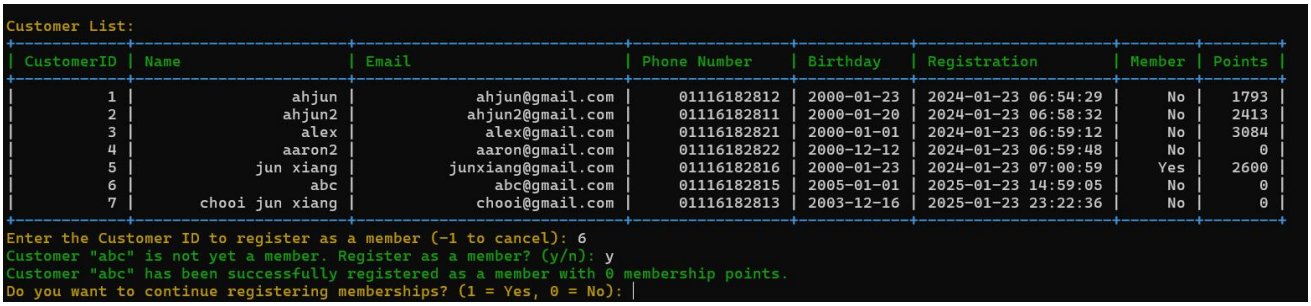


FIGURE 39 : REGISTER CUSTOMER MEMBERSHIP

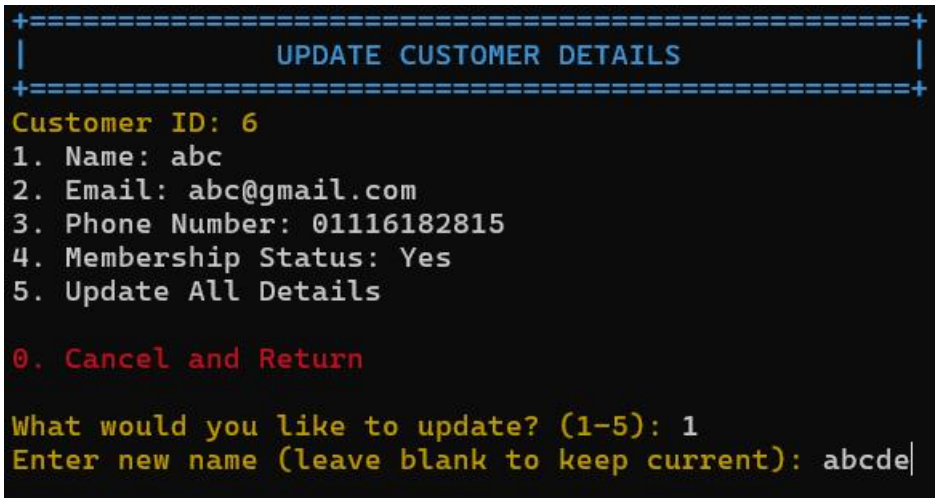


FIGURE 40 : UPDATE CUSTOMER INFORMATION

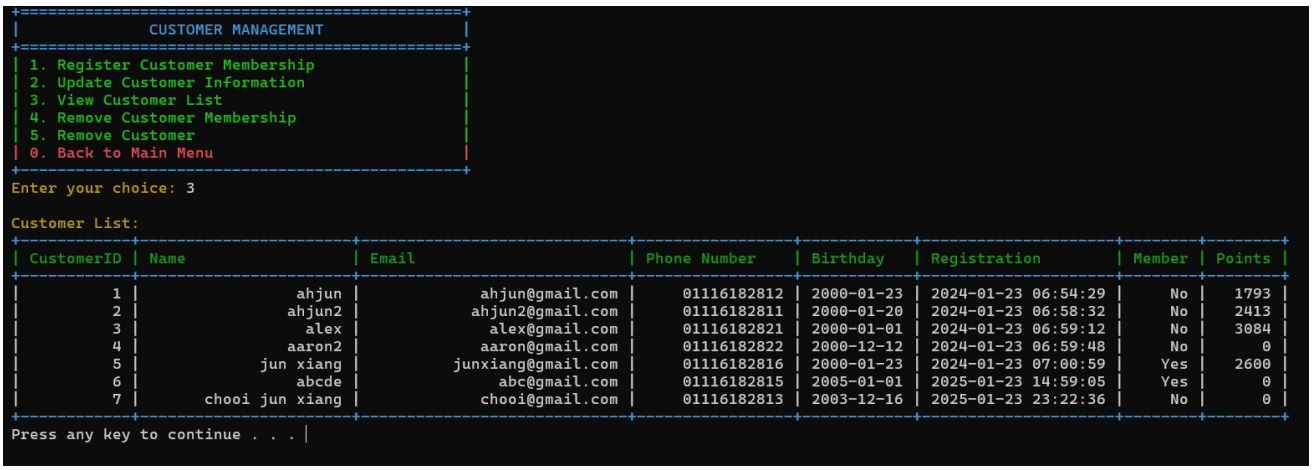


FIGURE 41 : VIEW CUSTOMER LIST

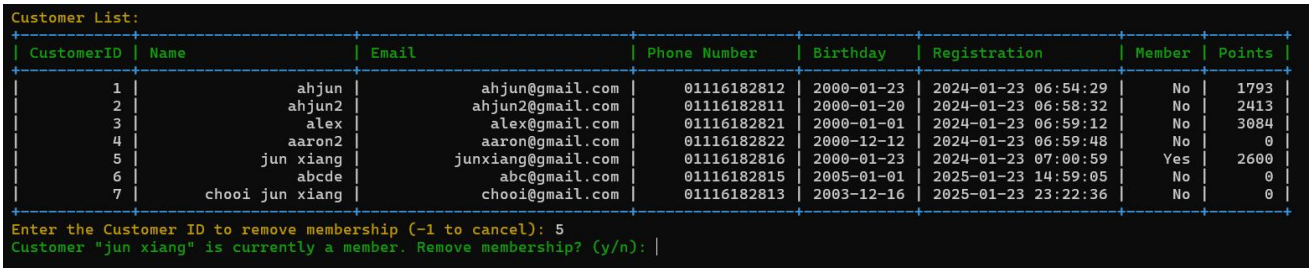


FIGURE 42 : REMOVE CUSTOMER MEMBERSHIP (PART 1)

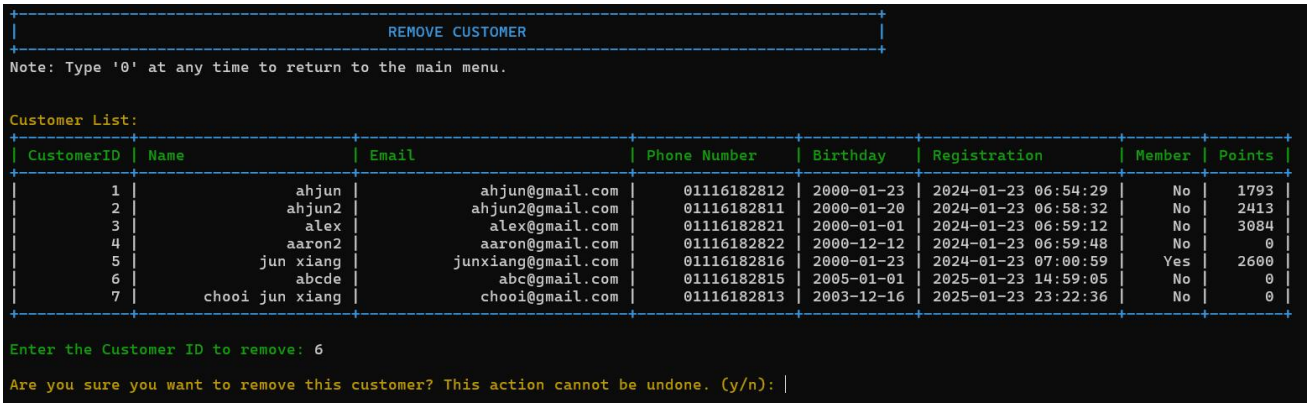


FIGURE 43 REMOVE CUSTOMER MEMBERSHIP (PART 2)

```
Customer List:
+------------+-----------------+-------------------------+--------------+------------+---------------------+--------+--------+---------------+
| CustomerID | Name            | Email                   | Phone Number | Birthday   | Registration        | Member | Points | Unpaid Orders |
+------------+-----------------+-------------------------+--------------+------------+---------------------+--------+--------+---------------+
| 1          | ahjun           | ahjun@gmail.com         | 01116182812  | 2000-01-23 | 2024-01-23 06:54:29 | Yes    | 1794   | 1             |
| 2          | ahjun2          | ahjun2@gmail.com        | 01116182811  | 2000-01-20 | 2024-01-23 06:58:32 | No     | 2413   | 0             |
| 3          | alex            | alex@gmail.com          | 01116182821  | 2000-01-01 | 2024-01-23 06:59:12 | No     | 3084   | 0             |
| 4          | aaron2          | aaron@gmail.com         | 01116182822  | 2000-12-12 | 2024-01-23 06:59:48 | No     | 0      | 0             |
| 5          | jun xiang       | junxiang@gmail.com      | 01116182816  | 2000-01-23 | 2024-01-23 07:00:59 | Yes    | 2600   | 0             |
| 6          | abcde           | abc@gmail.com           | 01116182815  | 2005-01-01 | 2025-01-23 14:59:05 | No     | 0      | 0             |
| 7          | chooi jun xiang | chooi@gmail.com         | 01116182813  | 2003-12-16 | 2025-01-23 23:22:36 | No     | 0      | 0             |
+------------+-----------------+-------------------------+--------------+------------+---------------------+--------+--------+---------------+

Enter customer ID (or 0 to go back): 1

========================================
               Order Details
========================================
Total amount before discount: RM110.00

Member discount applied (5%): RM5.50
over RM100 discount applied (10%): RM11.00
Birthday discount applied (15%): RM16.50
----------------------------------------
Total discount: RM33.00
Final amount after discount: RM77.00
========================================

Current membership points: 1794
Do you want to redeem points? (1 for Yes, 0 for No): 1
Enter points to redeem (max 1794): 10
Points redeemed: 10
New final amount to pay: RM67.00

Enter payment amount: 67

Payment successful! Change to return: RM0.00
You earned 6 points for this payment.
Do you want a receipt? (1 for Yes, 0 for No): 1

Receipt saved as: receipt_124.txt

Do you want to process another payment? (1 for Yes, 0 to go back):
```

FIGURE 44 : PAYMENT

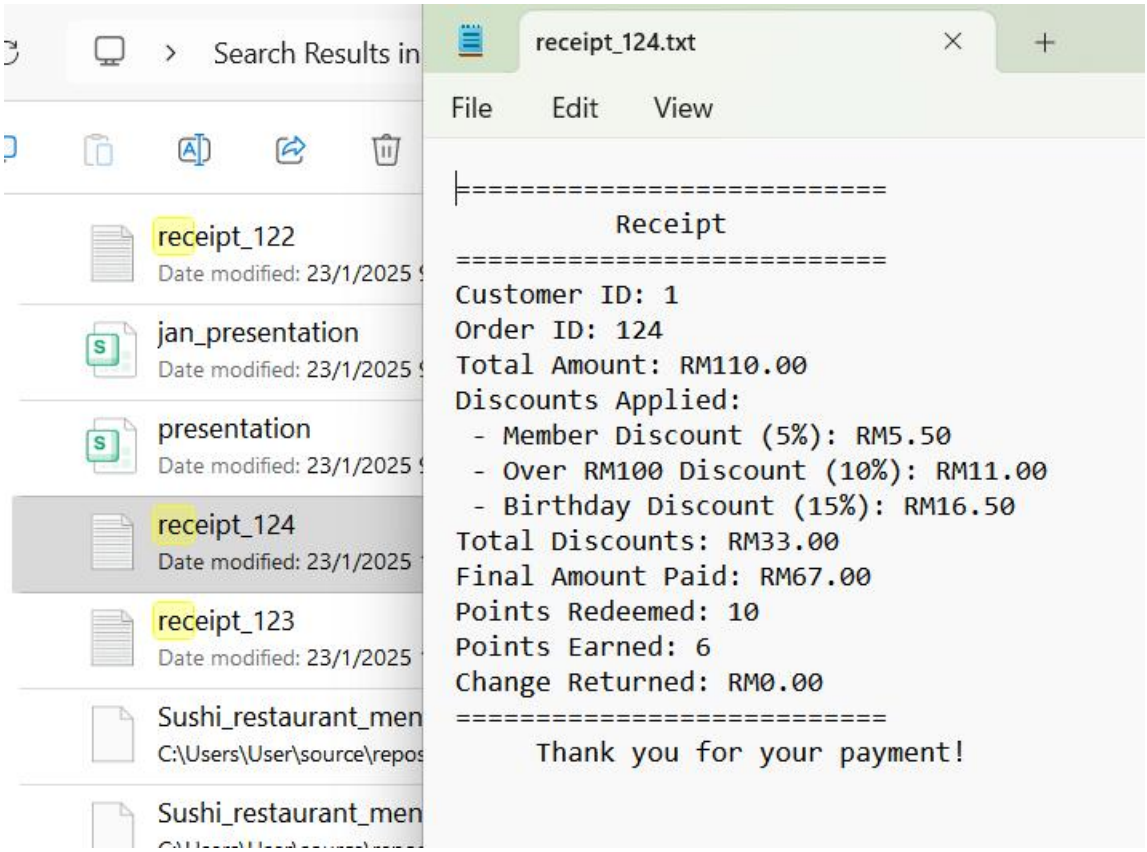


FIGURE 45 : PAYMENT RECEIPT SAVE AS TXT FILE

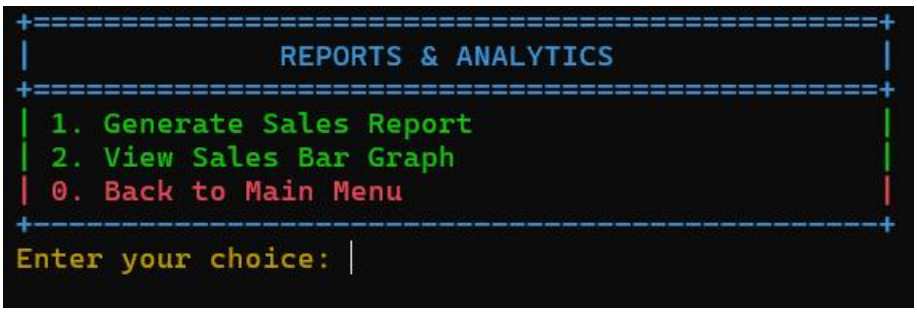


FIGURE 46 : REPORT AND ANALYTICS

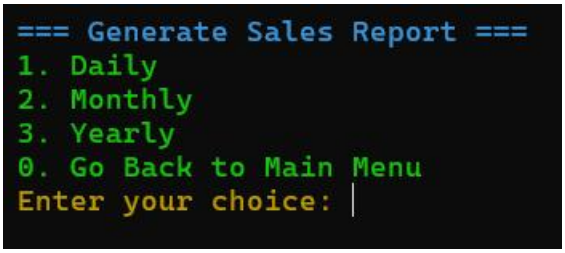


FIGURE 47 : GENERATE SALES REPORT BY TIME FRAME

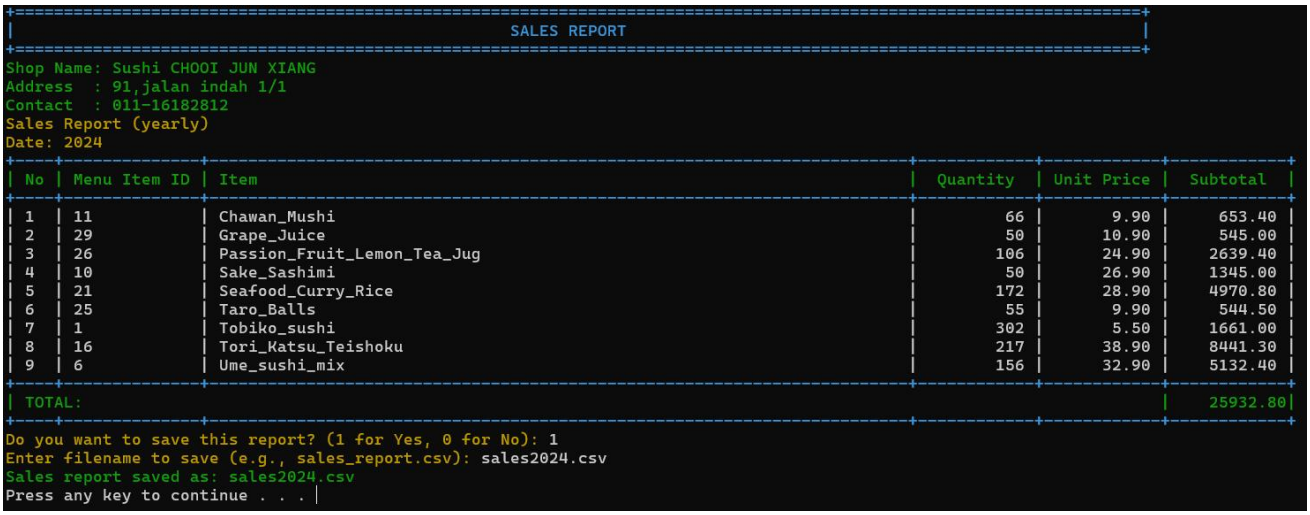


FIGURE 48 : SALES REPORT AND PROMPT ASK WANT TO SAVE CSV FILE OR NOT

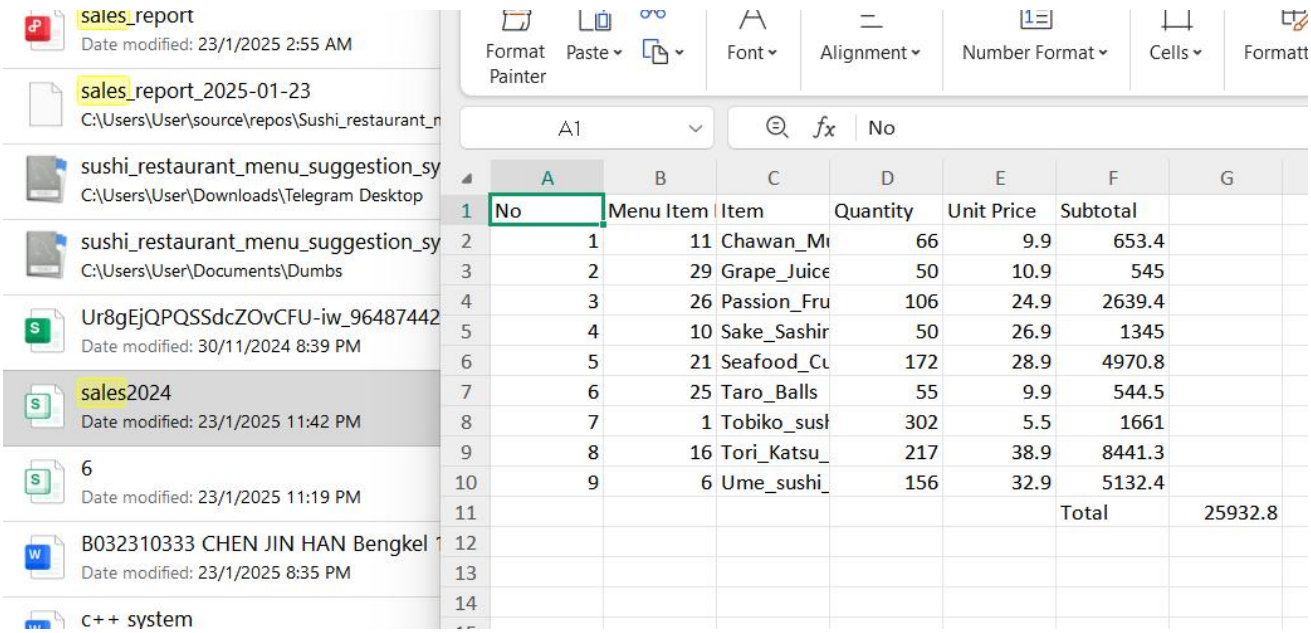


FIGURE 49 : SALES REPORT SAVE AS CSV FILE

FIGURE 50 : SALES BAR GRAPH MENU

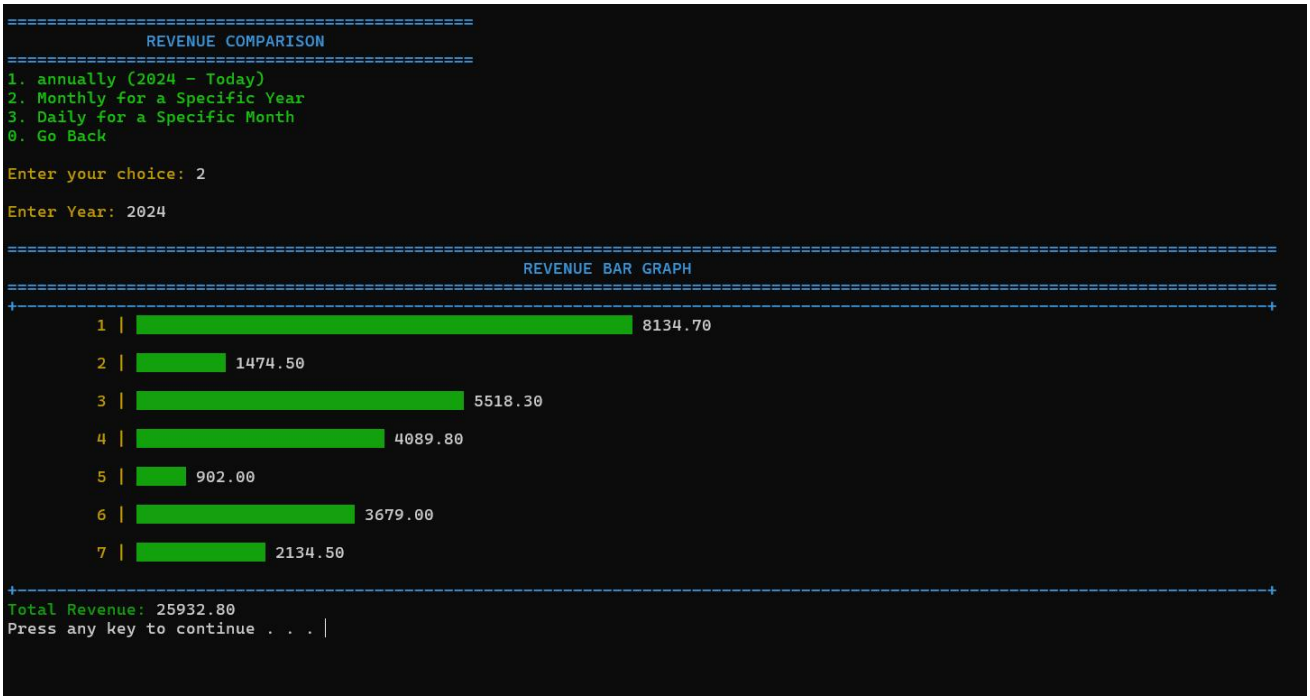


FIGURE 51 : SALES BAR GRAPH FOR 2024

FIGURE 52 : ITEM SALES MENU

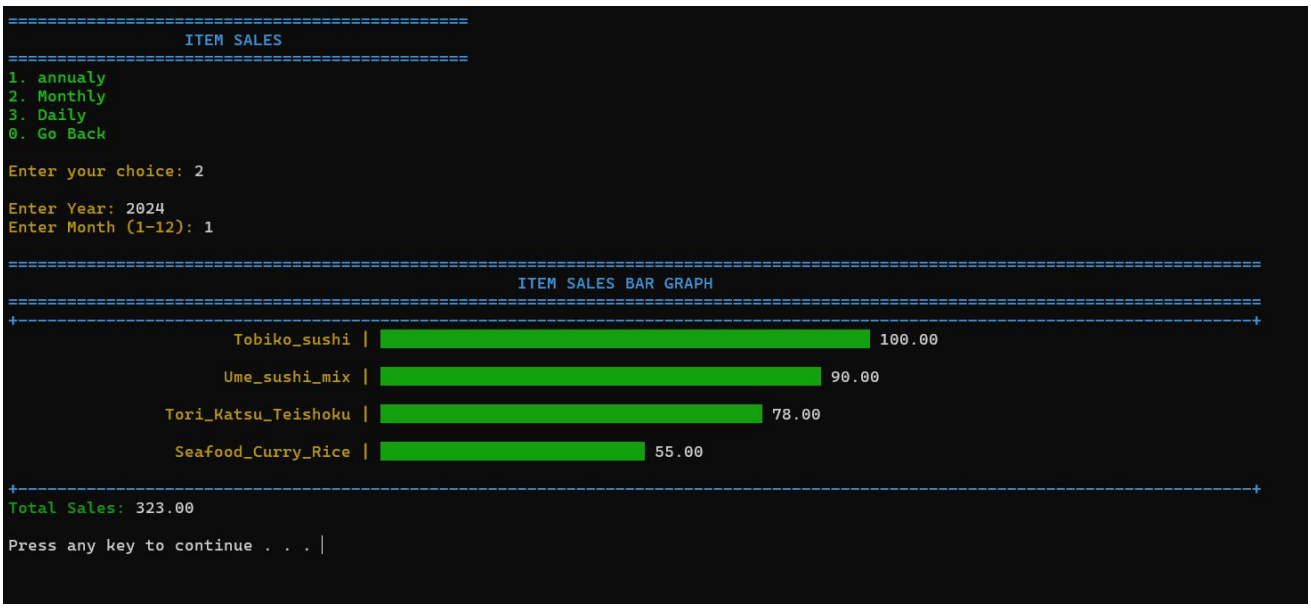
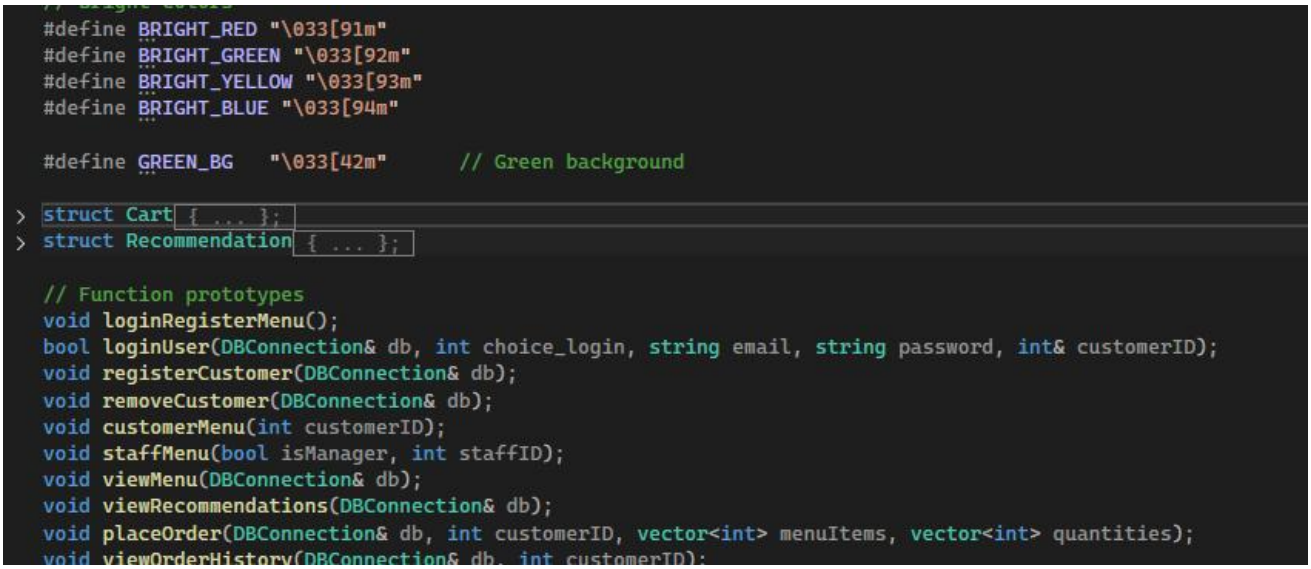


FIGURE 53 : ITEM SALES GRAF BAR FOR 2024 JAN

CHAPTER 4: IMPLEMENTATION

4.1 Naming Convention

Naming Convention are guidelines that used to make code more readable and maintainable.



```
#define BRIGHT_RED "\033[91m"
#define BRIGHT_GREEN "\033[92m"
#define BRIGHT_YELLOW "\033[93m"
#define BRIGHT_BLUE "\033[94m"

#define GREEN_BG   "\033[42m"      // Green background

struct Cart { ... };
struct Recommendation { ... };

// Function prototypes
void loginRegisterMenu();
bool loginUser(DBConnection& db, int choice_login, string email, string password, int& customerID);
void registerCustomer(DBConnection& db);
void removeCustomer(DBConnection& db);
void customerMenu(int customerID);
void staffMenu(bool isManager, int staffID);
void viewMenu(DBConnection& db);
void viewRecommendations(DBConnection& db);
void placeOrder(DBConnection& db, int customerID, vector<int> menuItems, vector<int> quantities);
void viewOrderHistory(DBConnection& db, int customerID);
```

Figure 54: NAMING CONVENTION EXAMPLE

The project adheres to a clear and descriptive naming convention:

i) Variables: Use camelCase, e.g., customerID.

ii) Functions: Use descriptive names, e.g., viewMenu().

iii) Constants: Use uppercase, e.g., BRIGHT_GREEN.

4.2 Function

```
> struct Cart { ... };
> struct Recommendation { ... };

  // Function prototypes
  void loginRegisterMenu();
  bool loginUser(DBConnection& db, int choice_login, string email, string password, int& customerID);
  void registerCustomer(DBConnection& db);
  void removeCustomer(DBConnection& db);
  void customerMenu(int customerID);
  void staffMenu(bool isManager, int staffID);
  void viewMenu(DBConnection& db);
  void viewRecommendations(DBConnection& db);
  void placeOrder(DBConnection& db, int customerID, vector<int> menuItems, vector<int> quantities);
  void viewOrderHistory(DBConnection& db, int customerID);
  void addMenu(DBConnection& db);
  void updateMenu(DBConnection& db);
  void deleteMenu(DBConnection& db);
  void generateSalesReport(DBConnection& db, const string& timeframe, const string& date);
  int getOrCreateUnpaidOrder(DBConnection& db, int customerID);
  void payOrder(DBConnection& db);
  void generateSalesBarGraph(DBConnection& db);
  void registerMembership(DBConnection& db);
  void removeMembership(DBConnection& db);
  double calculateDiscount(double totalAmount, bool isMember, const string& birthday);
  void viewProfile(DBConnection& db, int customerID);
  void managerMenu();
  void registerStaff(DBConnection& db);
  void updateStaff(DBConnection& db);
  void viewStaffList(DBConnection& db);
  void viewCustomerList(DBConnection& db);
  void updateCustomerInformation(DBConnection& db);
  bool rateFood(DBConnection& db, int customerID, int menuItemID, int rating);
  string getMonthName(int month);
  void displayMessageBox(const string& message, const string& color);
  void loadingAnimation();
  double calculateBirthdayDiscount(double totalAmount, const string& birthday);
  void removeStaff(DBConnection& db);
  void removeCustomer(DBConnection& db);
  void deleteSpecificItem(DBConnection& db);
  void deleteEntireCategory(DBConnection& db);
```

Figure 55: FUNCTION EXAMPLE

Based on the figure, this project includes function where it return data type or no, and some of the parameters allowed pass by reference or pass by value to fulfil their task on specific place which made the project more efficient.

Key functions include:
addItem(): Adds an item to the customer's cart.
placeOrder(): Processes and confirms the order.
viewRecommendations(): Displays menu recommendations.

4.3 Array

```
// Step 1: Fetch categories
string query = "SELECT DISTINCT category FROM menu_item ORDER BY category";
db.prepareStatement(query);
db.QueryResult();

vector<string> categories;

// Display options with color coding
cout << BRIGHT_GREEN << "| " << left << setw(46) << "1 . Overall (View All Menu Items)" << "  |" << RESET << endl;

int index = 2;
while (db.res->next()) {
    string category = db.res->getString("category");
    categories.push_back(category);
    cout << BRIGHT_GREEN << "| " << left << setw(2) << index << ". "
        << left << setw(43) << category << " |" << RESET << endl;
    index++;
}

cout << BRIGHT_RED << "| " << left << setw(46) << "0 . Go Back" << "  |" << RESET << endl;
cout << CYAN << "+================================================+" << RESET << endl;
```

```
// Helper function to convert month number to name
string getMonthName(int month) {
    const string months[] = { "January", "February", "March", "April", "May", "June",
                              "July", "August", "September", "October", "November", "December" };
    return months[month - 1];
}
```

```
struct Cart {
    map<int, int> items; // Menu item ID -> Quantity

    void addItem(int itemID, int quantity, DBConnection& db) {
        // Add the item to the cart
        items[itemID] += quantity;

        // Fetch item details from the database
        db.prepareStatement("SELECT item_name, item_price FROM menu_item WHERE menu_item_id = ?");
        db.stmt->setInt(1, itemID);
        db.QueryResult();

        // Check if the item exists in the database
        if (db.res->next()) {
            string itemName = db.res->getString("item_name");
            double itemPrice = db.res->getDouble("item_price");

            // Display the item details
            cout << BRIGHT_GREEN << "\nItem successfully added to the cart!\n" << RESET;
            cout << "-----------------------------------\n";
            cout << "Item ID    : " << itemID << "\n";
            cout << "Item Name  : " << itemName << "\n";
            cout << "Price (RM) : " << fixed << setprecision(2) << itemPrice << "\n";
            cout << "Quantity   : " << quantity << "\n";
            cout << "-----------------------------------\n";
        }
        else {
            // Handle the case where the item ID doesn't exist
            cout << "Error: Item ID " << itemID << " not found in the database.\n";
        }

        // Reminder for the user
        cout << CYAN << "**After you finish adding to the cart,\n" << RESET;
        cout << CYAN << "don't forget to confirm your order! Otherwise, we will not receive your order.\n\n" << RESET;
    }

    void removeItem(int itemID, const string& quantityInput, DBConnection& db) {
```

Figure 56 , 57 , 58 : ARRAY EXAMPLE

The project utilizes arrays for menu storage and vectors for dynamic data handling, such as cart items and quantities.These figures showing the using of array, which are constant array and dynamic array in C++.

4.4 Selection

```
bool isValid = true;
// Input validation
if (cin.fail() || choice_customer_menu < 0 || choice_customer_menu > 9) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << BRIGHT_RED << " Invalid choice! Please enter 1 - 9 only.\n" << RESET;
    choice_customer_menu = -1;
    system("pause");
    continue;
}
```

Figure 59: IF ELSE SELECTION FUNCTION

Selection statements (if, switch) are extensively used to handle user choices in menus and error validation

```
switch (choice_customer_menu) {
case 1: // View Menu
    viewMenu(db);
    system("pause");
    break;

case 2: { // Add order to Cart
    int itemID;
    do {
        system("cls");

        // Display the menu header
        cout << CYAN << "+================================================================+" << RESET << endl;
        cout << CYAN << "|                          ORDER MENU                            |" << RESET << endl;
        cout << CYAN << "+================================================================+" << RESET << endl;

        // Query to get all menu items
        string query = "SELECT category, menu_item_id, item_name, item_price FROM menu_item ORDER BY category, item_price";
        db.prepareStatement(query);
        db.QueryResult();

        string currentCategory = "";
        bool hasItems = false;

        // Display all items by category
        while (db.res->next()) {
            hasItems = true;
            string category = db.res->getString("category");

            if (category != currentCategory) {
                if (!currentCategory.empty()) {
                    cout << CYAN << "+------------------+----------------------------------+---------+" << RESET << endl << endl;
                }
                currentCategory = category;
                cout << YELLOW << "| Category: " << left << setw(39) << category << "               |" << RESET << endl;
                cout << CYAN << "+------------------+----------------------------------+---------+" << RESET << endl;
                cout << GREEN << "| Item ID          | Item Name                        |Price(RM)|" << RESET << endl;
                cout << CYAN << "+------------------+----------------------------------+---------+" << RESET << endl;
            }
```

Figure 60: SWITCH CASE SELECTION FUNCTION

These figures show different selection methods that used in this project, which are if else selection and switch case selection.

4.5 Control (Repetition statement)

Control statements like while, for, and do-while loops manage user interactions and program flow effectively.

```
        cout << CYAN << "+-------------------------------------------------------------------------------------------------------------------+\n" << RESET;
        for (const auto& entry : data) {
            int barLength = static_cast<int>((entry.second / maxValue) * maxBarLength);
            cout << YELLOW << setw(10) << entry.first << " | " << RESET;
            // Draw the bar with colored blocks
            for (int i = 0; i < barLength; i++) {
                cout << GREEN_BG << " " << RESET; // Colored block for each unit
            }
            cout << " " << fixed << setprecision(2) << entry.second << "\n\n";
        }
        cout << CYAN << "+-------------------------------------------------------------------------------------------------------------------+\n" << RESET;
        cout <<GREEN << "Total Revenue: " << RESET << fixed << setprecision(2) << totalValue << "\n";
    }
```

Figure 61 : FOR LOOP EXAMPLE

```
while (db.res->next()) {
    hasItems = true;
    string category = db.res->getString("category");

    if (category != currentCategory) {
        if (!currentCategory.empty()) {
            cout << CYAN << "+------------------+-----------------------------------+---------+" << RESET << endl << endl;
        }
        currentCategory = category;
        cout << YELLOW << "| Category: " << left << setw(39) << category << "               |" << RESET << endl;
        cout << CYAN << "+------------------+-----------------------------------+---------+" << RESET << endl;
        cout << GREEN << "| Item ID          | Item Name                         |Price(RM)|" << RESET << endl;
        cout << CYAN << "+------------------+-----------------------------------+---------+" << RESET << endl;
    }

    int itemID = db.res->getInt("menu_item_id");
    string itemName = db.res->getString("item_name");
    double itemPrice = db.res->getDouble("item_price");

    // Format price to 2 decimal places
    stringstream priceStr;
    priceStr << fixed << setprecision(2) << itemPrice;

    cout << "| " << left << setw(18) << itemID
        << "| " << left << setw(35) << itemName
        << "| " << right << setw(7) << priceStr.str() << " |" << endl;
}
```

Figure 62 : WHILE LOOP EXAMPLE

```
do {
    system("cls"); // Clear screen for a refreshed menu view

    // Header with "Welcome customer"
    stringstream header;
    header << "Welcome customer, " << customerName << " (ID: " << customerID << ")";
    string headerStr = header.str();
    int totalWidth = 48; // Total width of the box, including borders
    int contentWidth = totalWidth - 2; // Space between the borders (| |)
    int padding = (contentWidth - headerStr.length()) / 2;
    int paddingLeft = padding;
    int paddingRight = contentWidth - headerStr.length() - paddingLeft; // Adjust right padding dynamically

    cout << CYAN << "+------------------------------------------------+" << RESET << endl;
    cout << CYAN << "|              CUSTOMER MENU                     |" << RESET << endl;
    cout << "| " << string(paddingLeft, ' ') << headerStr << string(paddingRight, ' ') << " |\n" << RESET;
    cout << CYAN << "+------------------------------------------------+" << RESET << endl;

    // Menu Options
    cout << BRIGHT_GREEN << "| 1. View Menu                                   |\n" << RESET;
    cout << BRIGHT_GREEN << "| 2. Add order to Cart                           |\n" << RESET;
    cout << BRIGHT_GREEN << "| 3. View Cart                                   |\n" << RESET;
    cout << BRIGHT_GREEN << "| 4. Remove Item from Cart                       |\n" << RESET;
    cout << BRIGHT_GREEN << "| 5. View Recommendations                        |\n" << RESET;
    cout << BRIGHT_GREEN << "| 6. View Order History                          |\n" << RESET;
    cout << BRIGHT_GREEN << "| 7. Confirm and Place Order                     |\n" << RESET;
    cout << BRIGHT_GREEN << "| 8. View Profile                                |\n" << RESET;
    cout << BRIGHT_GREEN << "| 9. Rate a Food                                 |\n" << RESET;
    cout << BRIGHT_RED << "| 0. Logout                                      |\n" << RESET;
    cout << CYAN << "+------------------------------------------------+" << RESET << endl;
    cout << YELLOW << "Enter your choice: " << RESET;
    cin >> choice_customer_menu;


    bool isValid = true;
    // Input validation
    if (cin.fail() || choice_customer_menu < 0 || choice_customer_menu > 9){ ... }

    system("cls"); // Refresh screen after every menu option

    switch (choice_customer_menu){ ... }

} while (choice_customer_menu != 0);
```

Figure 63 : DO WHILE LOOP EXAMPLE

These figures showing the using of repetition statement control technique in this project.

4.6 Pointer

Pointers manage dynamic memory allocation, such as database query results and customer data.

```
// loading animation
void loadingAnimation() {
    const char* spinner = "|/-\\";
    cout << PROMPT << "Processing " << RESET;
    for (int i = 0; i < 10; ++i) {
        cout << spinner[i % 4] << flush;
        this_thread::sleep_for(chrono::milliseconds(150));
        cout << "\b";
    }
    cout << " Done!\n";
}
```

Figure 64 : POINTER EXAMPLE

This figure shows the example of pointer using in this project.

4.7 Error Handling

The system incorporates robust error-handling mechanisms, including:

1. Input validation for user choices.

2. Database query error checks.

3. Exception handling for edge cases.

```
while (true) {
    system("cls"); // Clear the screen (for Windows, use "clear" on Unix)
    cout << CYAN << "+-------------------------------------------------+\n" << RESET;
    cout << CYAN << "|           WELCOME TO SUSHI SYSTEM               |\n" << RESET;
    cout << CYAN << "+-------------------------------------------------+\n" << RESET;
    cout << BRIGHT_GREEN << "| 1. Login                                        |\n" << RESET;
    cout << BRIGHT_GREEN << "| 2. Register                                     |\n" << RESET;
    cout << BRIGHT_RED << "| 0. Exit                                         |\n" << RESET;
    cout << CYAN << "+-------------------------------------------------+\n" << RESET;
    cout << YELLOW << "Enter your choice: " << RESET;
    int choice;
    cin >> choice;

    if (cin.fail() || choice < 0 || choice > 2) { // Validate input
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << ERROR << " Invalid choice! Please enter 1, 2, or 0.\n" << RESET;
        system("pause");
        continue;
    }
```

```
// Function to login for customer or staff
bool loginUser(DBConnection& db, int choice_login, string email, string password, int& customerID) {
    try {
        if (choice_login == 1) {
            // Check credentials for customer
            db.prepareStatement("SELECT * FROM customer WHERE customer_email = ? AND customer_password = ?");
            db.stmt->setString(1, email);
            db.stmt->setString(2, password);
            db.QueryResult();

            if (db.res->next()) {
                customerID = db.res->getInt("customer_id");  // Get the customer ID
                return true;  // Customer login successful
            }
        }
        else if (choice_login == 2) {
            // Check credentials for staff
            db.prepareStatement("SELECT * FROM staff WHERE staff_email = ? AND staff_password = ?");
            db.stmt->setString(1, email);
            db.stmt->setString(2, password);
            db.QueryResult();

            if (db.res->next()) {
                return true;  // Staff login successful
            }
        }
        cout << "Invalid login credentials." << endl;
        return false;  // Invalid login
    }
    catch (sql::SQLException& e) {
        system("cls");
        cerr << "Error logging in: " << e.what() << endl;
        return false;
    }
```

Figure 65, 66, 67: ERROR HANDLING EXAMPLE

This figure represents the error handling mechanisms used in this project. If the error occurs during processing due to failed query or wrong data, it will throw and catch Error handling and perform a certain action preventing this bad data and system failure.

CHAPTER 5: CONCLUSION

5.1 Constraints

The implementation of "Sushi Restaurant Menu Suggestions System" has numerous constraints that will be addressed to optimize its effectiveness and sustainability.

1. Dependence on Database: The entire smooth operation of the system is dependant on proper and uninterrupted database connectivity. Any aberration will have effects on operation concerning order processing and menu recommendation.

2. Limited Analytics Scope: At present, the system only confines its recommendation engine towards sales data analysis. Therefore, recommendations cannot be made most especially personal recommendations from much further data such as comments from customers or dining preferences.

3. Inventory Management Challenges: Accurate inventory management is one of the essentials in the successful operation of a restaurant. The very fact that ingredients are perishable necessitates accurate tracking in order to avoid overstocking or understocking, which would eventually lead to wastefulness or shortages. Advanced inventory management should resolve those challenges.

4. High Employees Turnover: The high turnover of employees in the restaurant industry usually denotes high training costs, which can add to inconsistency in service delivery. The training modules embedded in the system would help with procedure standardization to mitigate the effects of staff turnover.

5. Integration of Existing Systems: Despite this, the integration with existing hardware and software that the restaurant is using can sometimes prove challenging. Adaptation issues may come up requiring more funding to allow for customizations and tests.

5.2 Future Improvements

To enhance the system's capabilities and address existing constraints, the following improvements are proposed:

1.Bringing machine learning algorithms on board: Machine learning would enable the system analyze different datasets, which may consist of customer tastes and preferences and such, for offering a more personalized and more accurate menu recommendation. It will enhance the dining experience and possibly increase customer loyalty.

2.Higher Order Integration into Inventory Management: The development of a module to monitor real-time inventory levels, forecast the historical demand projection for stock requirements, and alert management about potential shortages or surpluses would have a stock optimization which reduces waste.

3.Multi-Lingual: This would improve the service with many more customers, as well bring more access and inclusivity. The language feature has been very relevant in diversified communities and also improve customer satisfaction.

4.Improved Staff Training Modules: Availability of training resources in the system will facilitate standardized onboarding of employees, further reduce the training time and effort put into training, and create consistency in service quality to come into better grips with high employee turnover issues.

5.Application Development Mobile: A mobile application creates convenience for customers by giving them access to the menu; ordering online and special promotions can help increase interactions and sales.

6. Integration with Modern POS Systems: The system needs to conform with the latest Point of Sale (POS) equipped with features like live-analysing sales with customers' relationship management for easy operations and catching valuable insights into business trends.

By acknowledging these constraints and proactively implementing the proposed improvements, the "Sushi Restaurant Menu Suggestions System" can evolve into a more robust, efficient, and user-friendly platform, ultimately contributing to the restaurant's success and customer satisfaction.

REFERENCES

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book* (2nd ed.). Pearson.

MySQL. (n.d.). *MySQL Connector/C++ Developer Guide*. MySQL Documentation. Retrieved from https://dev.mysql.com/doc/connector-cpp/9.0/en/

Noraswaliza Binti Abdullah, Mashanum Binti Osman, Zeratul Izzah Binti Mohd Yusoh., Aniza Binti Othman, Zarita Mohd Kosnin. (2023). *Computer Programming* (2023 ed.). Desktop Publisher.

Halterman, R. L. (2017). *Fundamentals of Programming C++*.