GROUP ASSIGNMENT

CT038-3-2-OODJ

Object Oriented Development With Java

MODULE CODE	:	CT038-3-2-OODJ
ASSIGNMENT TITLE	:	Group Assignment Report
INTAKE	:	APD2F2206CS(CYB)
DATE ASSIGNED	:	20th June 2022
DATE COMPLETED	:	2nd September 2022
LECTURER	:	Mr. Usman Hashmi

No.	Student Name	TP Number
1	Chang Shiau Huei	TP060322
2	Darrshan A/L Rajenderan	TP060967

Table of Contents

1.0 Introduction **5**
1.1 Assumptions 6
2.0 Designs **7**
2.1 Use Case Diagram 7
Use Case Specifications 8
2.1.1 - Customer Register 8
2.1.2 - Customer Log In 9
2.1.3 - Verify User Credentials 9
2.1.4 - View Profile 10
2.1.5 - Top Up Account Balance 10
2.1.6 - View Menu 11
2.1.7 - Add to Cart 11
2.1.8 - Provide Payment 12
2.1.9 - View Order History 13
2.1.10 - Provide Feedback 13
2.1.11 - Manager Register 14
2.1.12 - Manager Log In 14
2.1.13 - Verify User Credentials 15
2.1.14 - View Profile 16
2.1.15 Manipulate Menu 16
2.1.16 - Create Menu Item 17
2.1.17 - Modify Menu Item 17
3.1.18 - Delete Menu Item 18
3.1.19 - Manage Orders 19
3.1.20 - View and Print Sales Report 19
3.1.21 - View Customer Feedback 20
3.1.22 - Manipulate Access Control 20
3.1.23 - Modify Access Control 21
3.1.24 - Delete Access Control 22
3.1.25 - View Audit Log 22
2.2 Class Diagram 24
3.0 Object-Oriented Programming Concept **26**
3.1 Classes & Objects 26
3.1.1 - Single Responsibility Principle 27
3.1.2 - DRY (Don't Repeat Yourself) Principle 28

3.2 Modularity 30
3.3 Inheritance 31
3.3.1 - Extending Regular or Abstract Classes 31
3.3.2 - Implementing Interfaces in Java 32
3.4 Abstraction 33
3.5 Polymorphism 34
3.5.1 - Method Overloading 34
3.5.2 - Method Overriding 35
3.6 Encapsulation 37
3.6.1 - Access Modifiers 37
3.6.2 - Getters and Setters 37

4.0 User Interface 39
4.1 Welcome Page 39
4.2 Customer Pages 40
4.2.1 - Customer Registration 40
4.2.2 - Customer Login 43
4.2.3 - Customer Home 45
4.2.4 - Customer Menu 46
4.2.5 - Customer Profile 49
4.2.6 - Customer Order History 51
4.3 Manager Pages 53
4.3.1 - Manager Registration 53
4.3.2 - Manager Login 56
4.3.3 - Manager Home 58
4.3.4 - Manager Menu 60
4.3.5 - Manager Profile 63
4.3.6 - Manager Approval 64
4.3.7 - Manager Manage Order 66
4.3.8 - Manager Order History 68
4.3.9 - Manager Feedback 69
4.3.10 - Manager Sales Report 70
4.3.11 - Manager Audit Log 71

5.0 Extra Features 73
5.1 Hashing 73
5.2 Report Generation in PDF 75
5.3 Duplication/Unique Value Check 78
5.4 Logging using Log4j 79

5.5 Top Up 82

5.0 Conclusion 84

6.0 References 85

1.0 Introduction

Object Oriented programming (OOP) is a programming paradigm based on the concepts of classes and objects. It is based around the idea of structuring a software program into simple, reusable pieces of code referred to as classes (Doherty, 2020). These classes are essentially blueprints to create instances of objects which can use the functionalities within the classes. Java is an example of a programming language that supports object oriented programming.

The following documentation is based on the GUI driven program we have developed called the APU Cafeteria Ordering System using Java which showcases a number of object oriented programming concepts throughout the process. The purpose of this program is to develop a new Cafeteria Ordering System for APU which is more convenient and efficient for both its customers and managers.

Within the system customers are able to register themselves using credentials such as their Customer ID, Name, Password and Email address. Upon successful registration and login to the system and begin ordering food by adding it into their cart, paying for it and waiting for their order to be served. After their order is served they will be able to view their order history and submit feedback for their orders if they choose to. Aside from that, the customers can view the profile page which shows their credentials along with a section to top up the current balance within their account.

The Manager on the other hand will go through a similar process of registering for an account. However, they will need to be approved by the system admin. Only after they have been approved will they be able to log in. However, if the admin chooses to reject them, their account information will be removed from the system. Upon login, the manager has many different functions to access within the system which are to update the menu, view the order history, manager orders, view their profile, view the sales report as well as feedback. The Admin essentially has the main functionalities of the Manager along with two other functions which are approving manager accounts and viewing the audit log.

1.1 Assumptions

1. The customer will be required to pay for their food before it is served to them.
2. The customers will be given a RM50 balance in their account when they have successfully registered.
3. The customer can top up a minimum of RM 1.00 up to a maximum of RM 100.00 to their account balance at one given time.
4. Only the admin can access the Manager Account Approval page and Manager Audit Log page.
5. Items in the menu can be set to "OUT OF STOCK" status temporarily which removes it from the view of the customers, and it could be altered back to "AVAILABLE" by modifying it in the Menu table in the Manager Menu page.

2.0 Designs

2.1 Use Case Diagram

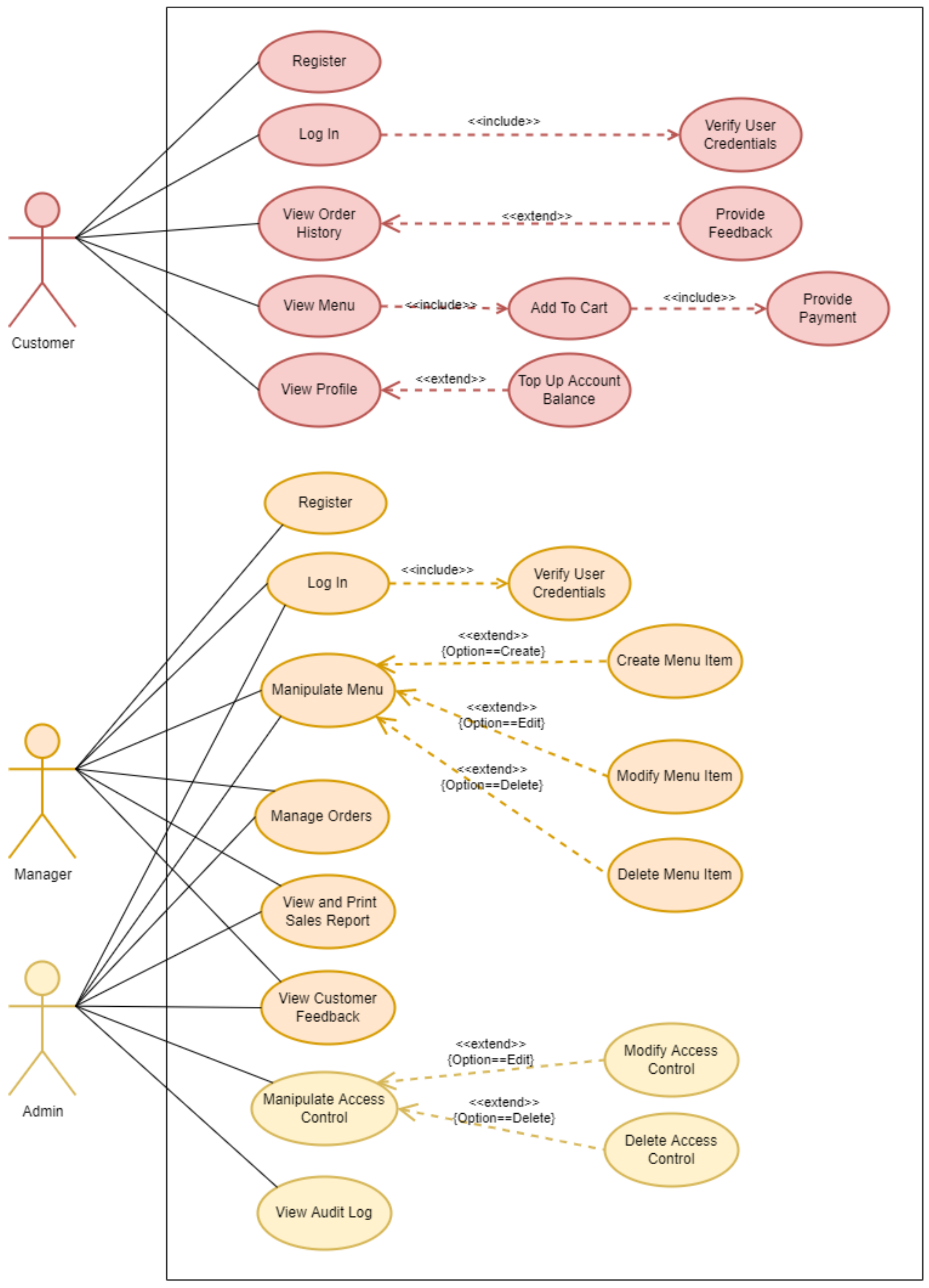


Figure 2.1 - Use Case Diagram

Use Case Specifications

Based on the use case diagram, there are a total of 3 different actors interacting with the system. The Customer can access the system and perform various actions related to ordering food and providing reviews. The Manager is responsible for creating the menu, and managing orders through the system for the customers. By managing the menu, they are essentially responsible for creating the items that the Customers will be able to order through the system. Aside from that they are also able to do other managerial actions such as view sales reports and the order history. The Admin or Administrator can perform similar actions as the Manager along with several other actions related to manipulating access control and viewing logs.

2.1.1 - Customer Register

Use Case	Register
Brief Description	This use case allows for the Customer to register for an account to access the system.
Actors	Customer
Preconditions	Customers must key in all of the required credentials.
Main Flow	1. Customer must select the option to Register for a Customer Account. 2. Customer must then input all of the required details. 3. Customer must ensure that all details are accurate. 4. Customer will then click on the register button to Register .
Alternative Flow	1. Error message shown if any input is invalid and a new account will not be created.

2.1.2 - Customer Log In

Use Case	Log In
Brief Description	This use case allows Customers to log in to the system with their account credentials.
Actors	Customer
Preconditions	Customer needs to have an account prior to logging in.
Main Flow	1. The Customer must select the option to login as a Customer. 2. The Customer must input the ID and password that they used to register.

2.1.3 - Verify User Credentials

Use Case	Verify User Credentials
Brief Description	Verifies the credentials of the user when they are logging into the system as a Customer.
Actors	Customer
Preconditions	Customer needs to have an account created in the system prior to logging in.
Main Flow	1. System receives the Customer ID and password input of the Customer 2. The system will then compare the inputs to the data in the system. 3. ID and Password is validated. 4. Customer is then successfully logged in and taken to the Home Page.

Alternative Flow	1. Error message displayed when the Customer/Admin ID does not exist in the system or if the password entered does not match the Customer/Admin ID within the system.

2.1.4 - View Profile

Use Case	View Profile
Brief Description	Customers are able to view their Customer Profile on the system and are able to top up their account balance.
Actors	Customer
Preconditions	1. Customer is logged in to their account. 2. Customer has selected the View Profile page.
Main Flow	1. Customer views their details and credentials

2.1.5 - Top Up Account Balance

Use Case	Top Up Account Balance
Brief Description	Allows the Customer to top up their account balance by inputting the amount.
Actors	Customer
Preconditions	1. The Customer is logged in to their account. 2. The Customer selected the View Profile page.
Main Flow	1. The Customer views their credentials.

	2. The Customer keys in an amount to top up. 3. The system validates the amount inputted by the Customer. 4. The amount inputted is then added to their current account balance. 5. The new balance is displayed back to the Customer after being topped up.
Alternative Flows	1. Error message is displayed when Customer keys in invalid amount. 2. Error message is displayed when Customer keys in an invalid top up amount or an amount outside the range of RM 1 to RM 100.

2.1.6 - View Menu

Use Case	View Menu
Brief Description	Allows the Customer to view the customer menu.
Actors	Customer
Preconditions	1. The Customer is logged 2. The Customer selected the View Menu page.
Main Flow	1. Customer views the items that can be ordered in the menu

2.1.7 - Add to Cart

Use Case	Add To Cart

Brief Description	Adds an item selected by the Customer to their cart.
Actors	Customer
Preconditions	1. Customer is logged in to their account. 2. Customer selected View Menu page
Main Flow	1. Customer selects an item from the menu table 2. Customer clicks the button to add the item to their cart.
Alternative Flow	1. Error message shown to the user for not selecting an item from the menu before adding to cart and other input validations.

2.1.8 - Provide Payment

Use Case	Provide Payment
Brief Description	Customer provides payment for their order by clicking on the checkout button.
Actors	Customer
Preconditions	1. Customer is logged in to their account. 2. Customer selected View Menu page 3. Customer has added items to cart
Main Flow	1. Customer confirms the order by clicking on the checkout button. 2. Balance is deducted from the Customer account.
Alternative Flow	1. Error message will be shown to the user if the account balance is insufficient.

2.1.9 - View Order History

Use Case	View Order History
Brief Description	Customer is able to view history of past orders in the system.
Actors	Customer
Preconditions	1. Customer is logged in to their account. 2. Customer selected View Order History page
Main Flow	1. Customer selects on View Order History page 2. Customer views order history in the table
Alternative Flows	1. Error message is shown to the user if a review has already been submitted for a particular order in the system.

2.1.10 - Provide Feedback

Use Case	Provide Feedback
Brief Description	Customer provides feedback of a particular order within the order history table.
Actors	Customer
Preconditions	1. Customer is logged in to their account. 2. Customer selects View Menu page.
Main Flow	1. Customer views order history in the table 2. Customer selects a particular order from the table 3. Customer writes a review in the text area for the order selected 4. “Submit Review” button is clicked to submit the review into the

	system.
Alternative Flow	1. Error message is shown to the user if a review has already been submitted for a particular order in the system.

2.1.11 - Manager Register

Use Case	Manager Register
Brief Description	This use case allows for the Manager to register for an account to access the system.
Actors	Manager
Preconditions	Manager must key in all of the required credentials
Main Flow	1. Managermust select the option to Register for a Customer Account. 2. Manager must then input all of the required details 3. Manager must ensure that all details are accurate 4. Managerwill then click on the register button to Register
Alternative Flows	1. Error message shown if any input is invalid and a new account will not be created.

2.1.12 - Manager Log In

Use Case	Manager Log In

Brief Description	This use case allows the Manager or Admin to log in to the system with their account credentials.
Actors	Manager, Admin
Preconditions	Manager needs to have an account prior to logging in (Admin already exists by default in the system)
Main Flow	1. The Manager/Admin must select the option to login as a Manager. 2. The Manager must input the ID and password that they used to register. (Admin account already exists by default in the system)

2.1.13 - Verify User Credentials

Use Case	Verify User Credentials
Brief Description	Verifies the credentials of the user when they are logging into the system as a Manager and Admin.
Actors	Manager, Admin
Preconditions	Manager needs to have an account created in the system prior to logging in with an approved account by Admin. (Admin account already exists by default in the system)
Main Flow	1. System receives the User ID and password input of the Manager/Admin 2. The system will then compare the inputs to the data in the system. 3. ID and Password is validated. 4. Manager/Admin is then successfully logged in and taken into

	the Home Page.
Alternative Flow	Error message displayed when the Customer ID does not exist in the system or if the password entered does not match the Customer ID within the system.

2.1.14 - View Profile

Use Case	View Profile
Brief Description	Manager/Admin is able to view their Profile on the system.
Actors	Manager, Admin
Preconditions	1. Manager/Admin is logged in to their account. 2. Manager/Admin has selected the View Profile page.
Main Flow	1. Manager/Admin views their credentials and other details.

2.1.15 Manipulate Menu

Use Case	Manipulate Menu
Brief Description	Manager/Admin can manipulate the contents of the menu.
Actors	Manager, Admin
Preconditions	1. Manager/Admin is logged in to their account. 2. Manager/Admin has selected the View Menu page.
Main Flow	1. Manager/Admin views contents of the menu.

2.1.16 - Create Menu Item

Use Case	Create Menu Item
Brief Description	Manager/Admin is able to create a new item in the menu table.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account. 2. Manager/Admin must be within the View Menu page.
Main Flow	1. Manager/Admin keys in details of the item that needs to be added into the Menu table such as Food ID, Food, Price, Type and Status. 2. The “Add” button is then clicked to add the item into the Menu table. 3. Prompt will be shown if data is added successfully into the Menu table.
Alternative Flow	1. Error message will be displayed to the user if all text fields are not filled out or other input validations.

2.1.17 - Modify Menu Item

Use Case	Modify Menu Item
Brief Description	Manager/Admin is able to modify an item in the Menu table.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account.

	2. Manager/Admin must be within the View Menu page.
Main Flow	1. Manager/Admin selects an item from the Menu table. 2. Item data is then manipulated through the text fields. 3. Menu items are saved back into the table by clicking the "Update" button. 4. Prompt will be shown if data is added successfully back into the Menu table.
Alternative Flow	1. Error message will be displayed to the user if all text fields are not are not filled out, invalid values are keyed in or if the item already exists.

3.1.18 - Delete Menu Item

Use Case	Delete Menu Item
Brief Description	Manager/Admin is able to delete an item in the Menu table.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account. 2. Manager/Admin must be within the View Menu page.
Main Flow	1. Manager/Admin selects an item from the Menu table. 2. Item is then deleted from the Menu table when clicking the "Delete" button. 3. Prompt message will be displayed that the item has been successfully deleted from the menu.
Alternative Flow	1. Error message will be displayed to the user if a row is not selected for deletion or if the table is empty.

3.1.19 - Manage Orders

Use Case	Manage Orders
Brief Description	Manager/Admin is able to manage the orders of the customers by completing orders and marking them on the system.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account. 2. Manager/Admin must be within the Manager Manage Orders page.
Main Flow	1. Manager/Admin must select a row from the pending orders table. 2. Once a row has been selected, they will need to click the "Order Completed" button if the order has been completed. 3. The order will then be set as completed within the system. 4. A prompt message displays that the order has been marked as completed.
Alternative Flow	1. Error message will be displayed to the user if a row is not selected before clicking the "Order Completed" button.

3.1.20 - View and Print Sales Report

Use Case	View Sales Report
Brief Description	Manager/Admin is able to view the sales reports within the system.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account.

	2. Manager/Admin must be within the Sales Report page.
Main Flow	1. Manager/Admin can view the general information such as system balance, total orders and total customers. 2. The sales of each item ordered on that particular day can be viewed by the manager by manually searching for the date. 3. The total sales for the selected date can be viewed at the bottom of the page.

3.1.21 - View Customer Feedback

Use Case	View Customer Feedback
Brief Description	Manager/Admin is able to view the Customer Feedback within the text file.
Actors	Manager, Admin
Preconditions	1. Manager/Admin must be logged in to their account. 2. Manager/Admin must be within the View Feedback page.
Main Flow	1. Manager/Admin can see each and every review/feedback submitted by the Customers in one table.

3.1.22 - Manipulate Access Control

Use Case	Manipulate Access Control
Brief Description	Admin is able to manipulate the access control of the Managers within the system.

Actors	Admin
Preconditions	1. Admin must be logged in to their account. 2. Admin must select the Manager Approval page..
Main Flow	1. Admin views the Manager Approval page 2. Admin can select a manager from the manager approval page to manipulate access control for.

3.1.23 - Modify Access Control

Use Case	Modify Access Control
Brief Description	Admin is able to approve the manager which would allow them to log in to the system.
Actors	Admin
Preconditions	1. Admin must be logged in to their account. 2. Admin must be within the Manager Approval page.
Main Flow	1. Admin can select a manager from the table to approve 2. The “Approve” button is clicked to change the Approval Status of the manager from “TO BE APPROVED” to “APPROVED”. 3. The manager approved will then be able to log in to the system.
Alternative Flow	1. Error message is displayed if a row from the table has not been selected to perform actions on.

3.1.24 - Delete Access Control

Use Case	Delete Access Control
Brief Description	Admin is able to reject a manager from the system by completely deleting their account.
Actors	Admin
Preconditions	1. Admin must be logged in to their account. 2. Admin must be within the Manager Approval page.
Main Flow	1. Admin can select a manager from the table to reject. 2. The “Reject” button must then be clicked to permanently remove a manager from the system.
Alternative Flow	1. Error message is displayed if a row from the table has not been selected to perform actions on.

3.1.25 - View Audit Log

Use Case	View Audit Log
Brief Description	Admin is able to view audit logs of all the user interactions within the system.
Actors	Admin
Preconditions	1. Admin must be logged in to their account. 2. Admin must be within the Audit Log page.
Main Flow	1. Admin can view the audit logs within the table. 2. The audit log can be further filtered to view specific

	information from the audit logs.

2.2 Class Diagram

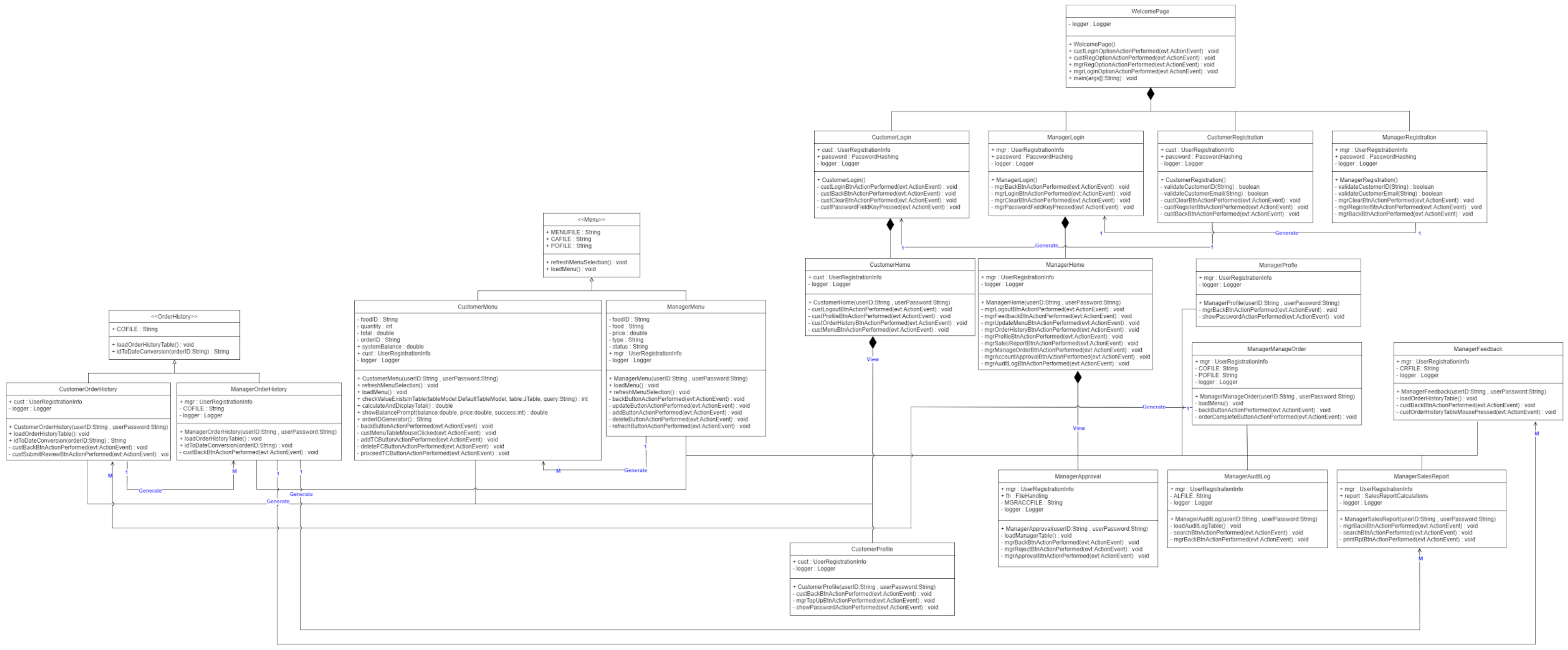

WelcomePage
- logger : Logger
+ WelcomePage()
+ custLoginOptionActionPerformed(evt:ActionEvent) : void
+ custRegOptionActionPerformed(evt:ActionEvent) : void
+ mgrRegOptionActionPerformed(evt:ActionEvent) : void
+ mgrLoginOptionActionPerformed(evt:ActionEvent) : void
+ main(args[]:String) : void
CustomerLogin
+ cust : UserRegistrationInfo
+ password : PasswordHashing
- logger : Logger
+ CustomerLogin()
- custLoginBtnActionPerformed(evt:ActionEvent) : void
- custBackBtnActionPerformed(evt:ActionEvent) : void
- custClearBtnActionPerformed(evt:ActionEvent) : void
- custPasswordFieldKeyPressed(evt:ActionEvent) : void
ManagerLogin
+ mgr : UserRegistrationInfo
+ password : PasswordHashing
- logger : Logger
+ ManagerLogin()
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
- mgrLoginBtnActionPerformed(evt:ActionEvent) : void
- mgrClearBtnActionPerformed(evt:ActionEvent) : void
- mgrPasswordFieldKeyPressed(evt:ActionEvent) : void
CustomerRegistration
+ cust : UserRegistrationInfo
+ password : PasswordHashing
- logger : Logger
+ CustomerRegistration()
- validateCustomerID(String) : boolean
- validateCustomerEmail(String) : boolean
- custClearBtnActionPerformed(evt:ActionEvent) : void
- custRegisterBtnActionPerformed(evt:ActionEvent) : void
- custBackBtnActionPerformed(evt:ActionEvent) : void
ManagerRegistration
+ mgr : UserRegistrationInfo
+ password : PasswordHashing
- logger : Logger
+ ManagerRegistration()
- validateCustomerID(String) : boolean
- validateCustomerEmail(String) : boolean
- mgrClearBtnActionPerformed(evt:ActionEvent) : void
- mgrRegisterBtnActionPerformed(evt:ActionEvent) : void
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
Generate
1
1
Generate
1
1
<<Menu>>
+ MENUFILE : String
+ CAFILE : String
+ POFILE : String
+ refreshMenuSelection() : void
+ loadMenu() : void
CustomerHome
+ cust : UserRegistrationInfo
- logger : Logger
+ CustomerHome(userID:String , userPassword:String)
- custLogoutBtnActionPerformed(evt:ActionEvent) : void
- custProfileBtnActionPerformed(evt:ActionEvent) : void
- custOrderHistoryBtnActionPerformed(evt:ActionEvent) : void
- custMenuBtnActionPerformed(evt:ActionEvent) : void
ManagerHome
+ mgr : UserRegistrationInfo
- logger : Logger
+ ManagerHome(userID:String , userPassword:String)
- mgrLogoutBtnActionPerformed(evt:ActionEvent) : void
- mgrFeedbackBtnActionPerformed(evt:ActionEvent) : void
- mgrUpdateMenuBtnActionPerformed(evt:ActionEvent) : void
- mgrOrderHistoryBtnActionPerformed(evt:ActionEvent) : void
- mgrProfileBtnActionPerformed(evt:ActionEvent) : void
- mgrSalesReportBtnActionPerformed(evt:ActionEvent) : void
- mgrManageOrderBtnActionPerformed(evt:ActionEvent) : void
- mgrAccountApprovalBtnActionPerformed(evt:ActionEvent) : void
- mgrAuditLogBtnActionPerformed(evt:ActionEvent) : void
ManagerProfle
+ mgr : UserRegistrationInfo
- logger : Logger
+ ManagerProfile(userID:String , userPassword:String)
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
- showPasswordActionPerformed(evt:ActionEvent) : void
<<OrderHistory>>
+ COFILE : String
+ loadOrderHistoryTable() : void
+ idToDateConversion(orderID:String) : String
CustomerMenu
- foodID : String
- quantity : int
- total : double
- orderID : String
+ systemBalance : double
+ cust : UserRegistrationInfo
- logger : Logger
+ CustomerMenu(userID:String , userPassword:String)
+ refreshMenuSelection() : void
+ loadMenu() : void
+ checkValueExistsInTable(tableModel:DefaultTableModel, table:JTable, query:String) : int
+ calculateAndDisplayTotal() : double
+ showBalancePrompt(balance:double, price:double, success:int) : double
+ orderIDGenerator() : String
- backButtonActionPerformed(evt:ActionEvent) : void
- custMenuTableMouseClicked(evt:ActionEvent) : void
- addTCButtonActionPerformed(evt:ActionEvent) : void
- deleteFCButtonActionPerformed(evt:ActionEvent) : void
- proceedTCButtonActionPerformed(evt:ActionEvent) : void
ManagerMenu
- foodID : String
- food : String
- price : double
- type : String
- status : String
+ mgr : UserRegistrationInfo
- logger : Logger
+ ManagerMenu(userID:String , userPassword:String)
+ loadMenu() : void
+ refreshMenuSelection() : void
- backButtonActionPerformed(evt:ActionEvent) : void
- updateButtonActionPerformed(evt:ActionEvent) : void
- addButtonActionPerformed(evt:ActionEvent) : void
- deleteButtonActionPerformed(evt:ActionEvent) : void
- refreshButtonActionPerformed(evt:ActionEvent) : void
ManagerManageOrder
+ mgr : UserRegistrationInfo
- COFILE: String
- POFILE: String
- logger : Logger
+ ManagerManageOrder(userID:String , userPassword:String)
- loadMenu() : void
- backButtonActionPerformed(evt:ActionEvent) : void
- orderCompleteButtonActionPerformed(evt:ActionEvent) : void
ManagerFeedback
+ mgr : UserRegistrationInfo
- CRFILE: String
- logger : Logger
+ ManagerFeedback(userID:String , userPassword:String)
- loadOrderHistoryTable() : void
- custBackBtnActionPerformed(evt:ActionEvent) : void
- custOrderHistoryTableMousePressed(evt:ActionEvent) : void
View
View
Generate
1
CustomerOrderHistory
+ cust : UserRegistrationInfo
- logger : Logger
+ CustomerOrderHistory(userID:String , userPassword:String)
+ loadOrderHistoryTable(): void
+ idToDateConversion(orderID:String) : String
- custBackBtnActionPerformed(evt:ActionEvent) : void
- custSubmitReviewBtnActionPerformed(evt:ActionEvent) : voi
ManagerOrderHistory
+ mgr : UserRegistrationInfo
- COFILE : String
- logger : Logger
+ ManagerOrderHistory(userID:String , userPassword:String)
+ loadOrderHistoryTable() : void
+ idToDateConversion(orderID:String) : void
- custBackBtnActionPerformed(evt:ActionEvent) : void
M
1
Generate
M
1
1
Generate
Generate
M
Generate
1
M
ManagerApproval
+ mgr : UserRegistrationInfo
+ fh : FileHandling
- MGRACCFILE : String
- logger : Logger
+ ManagerApproval(userID:String , userPassword:String)
- loadManagerTable() : void
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
- mgrRejectBtnActionPerformed(evt:ActionEvent) : void
- mgrApprovalBtnActionPerformed(evt:ActionEvent) : void
ManagerAuditLog
+ mgr : UserRegistrationInfo
- ALFILE: String
- logger : Logger
+ ManagerAuditLog(userID:String , userPassword:String)
- loadAuditLogTable() : void
- searchBtnActionPerformed(evt:ActionEvent) : void
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
ManagerSalesReport
+ mgr : UserRegistrationInfo
+ report : SalesReportCalculations
- logger : Logger
+ ManagerSalesReport(userID:String , userPassword:String)
- mgrBackBtnActionPerformed(evt:ActionEvent) : void
- searchBtnActionPerformed(evt:ActionEvent) : void
- printRptBtnActionPerformed(evt:ActionEvent) : void
M
CustomerProfile
+ cust : UserRegistrationInfo
- logger : Logger
+ CustomerProfile(userID:String , userPassword:String)
- custBackBtnActionPerformed(evt:ActionEvent) : void
- mgrTopUpBtnActionPerformed(evt:ActionEvent) : void
- showPasswordActionPerformed(evt:ActionEvent) : void

Figure 2.2.1 - Class Diagram for General Classes and Interfaces

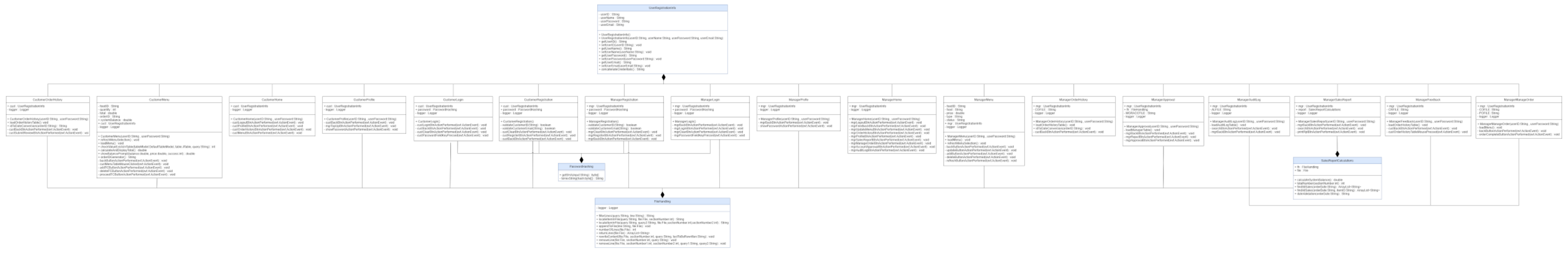


Figure 2.2.2 - Class Diagram for Utility Classes applied on General Classes

The full original diagram files can be found in the link below:

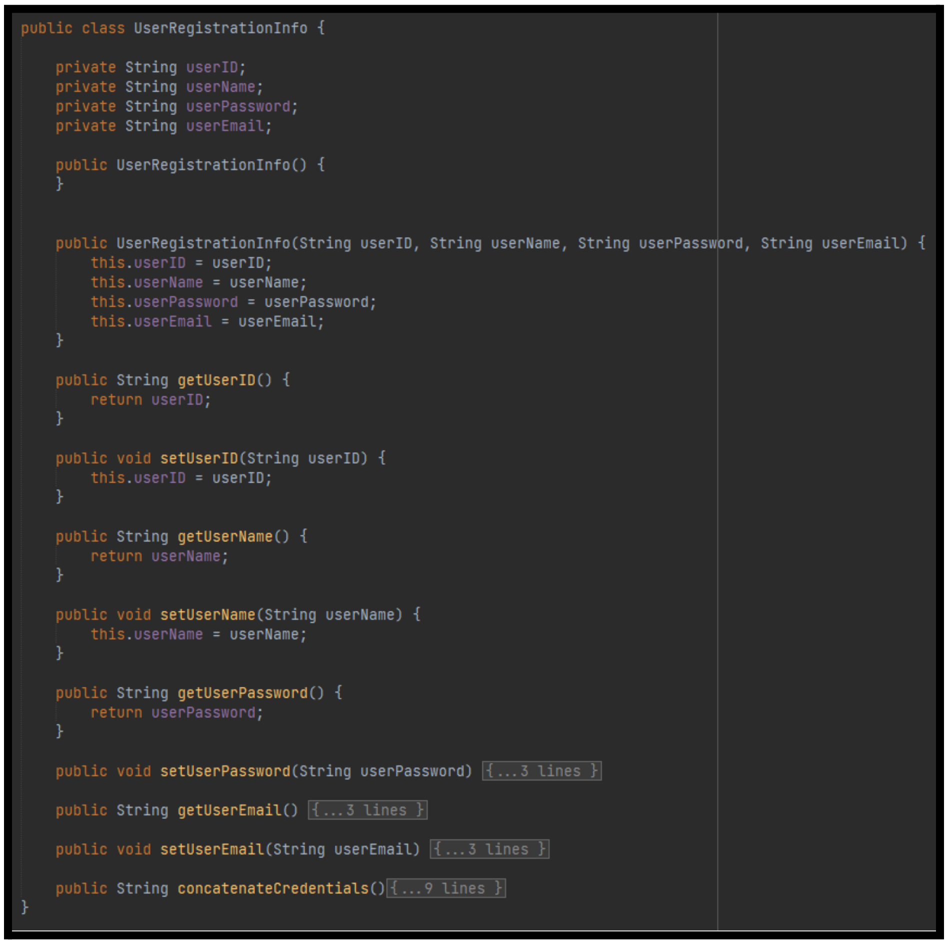
https://drive.google.com/drive/folders/1HL73YGg6JFCKG8N8sdrtgFMnwVDDokvV?usp=sharing

The diagrams are divided into two parts, where the first class diagram explains the entire flow of the project which includes both general classes and interfaces, while another class diagram explains the utility classes used in the general classes.

3.0 Object-Oriented Programming Concept

3.1 Classes & Objects

Classes and objects are fundamental concepts of OOP which are based on real life entities (geeksforgeeks, 2022). A class acts as a blueprint designed by the user that represents the set of properties that are common to all objects of a similar type. An object is an instance of the class which allows it to use all the methods within the class it was created from (javatpoint, 2011).



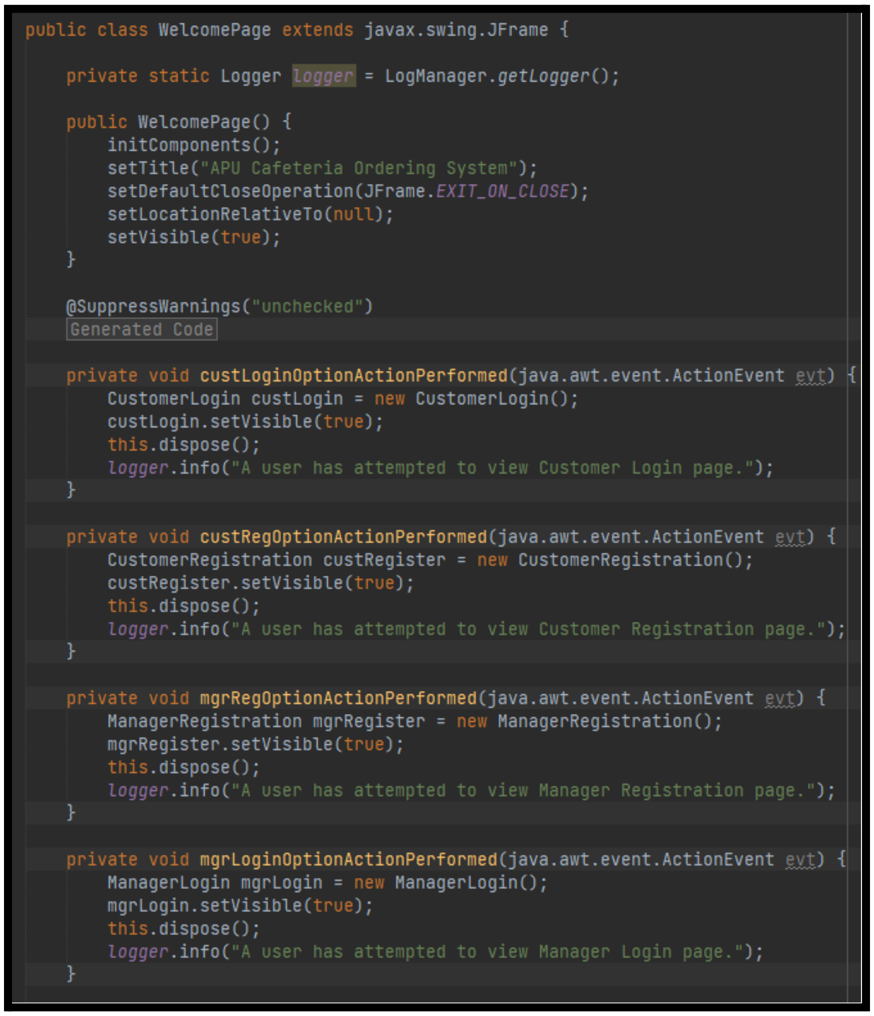
```java
public class UserRegistrationInfo {

    private String userID;
    private String userName;
    private String userPassword;
    private String userEmail;

    public UserRegistrationInfo() {
    }


    public UserRegistrationInfo(String userID, String userName, String userPassword, String userEmail) {
        this.userID = userID;
        this.userName = userName;
        this.userPassword = userPassword;
        this.userEmail = userEmail;
    }

    public String getUserID() {
        return userID;
    }

    public void setUserID(String userID) {
        this.userID = userID;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserPassword() {
        return userPassword;
    }

    public void setUserPassword(String userPassword) {...3 lines }

    public String getUserEmail() {...3 lines }

    public void setUserEmail(String userEmail) {...3 lines }

    public String concatenateCredentials(){...9 lines }
}
```

Figure 3.1 - Class & Object Sample

An example of a class that was implemented in the system is the UserRegistrationInfo class which is used to create an object consisting of attributes and operations. Aside from that is that it shares relationships with other classes such as CustomerRegistration and ManagerRegistration.

3.1.1 - Single Responsibility Principle

Single responsibility refers to the design of a class with one sole responsibility to carry out. The idea behind it is that each class within the program should have only a single purpose (Abba, 2022). This has become a standard for developers when designing classes to ensure that they are easily maintainable, reusable and flexible.



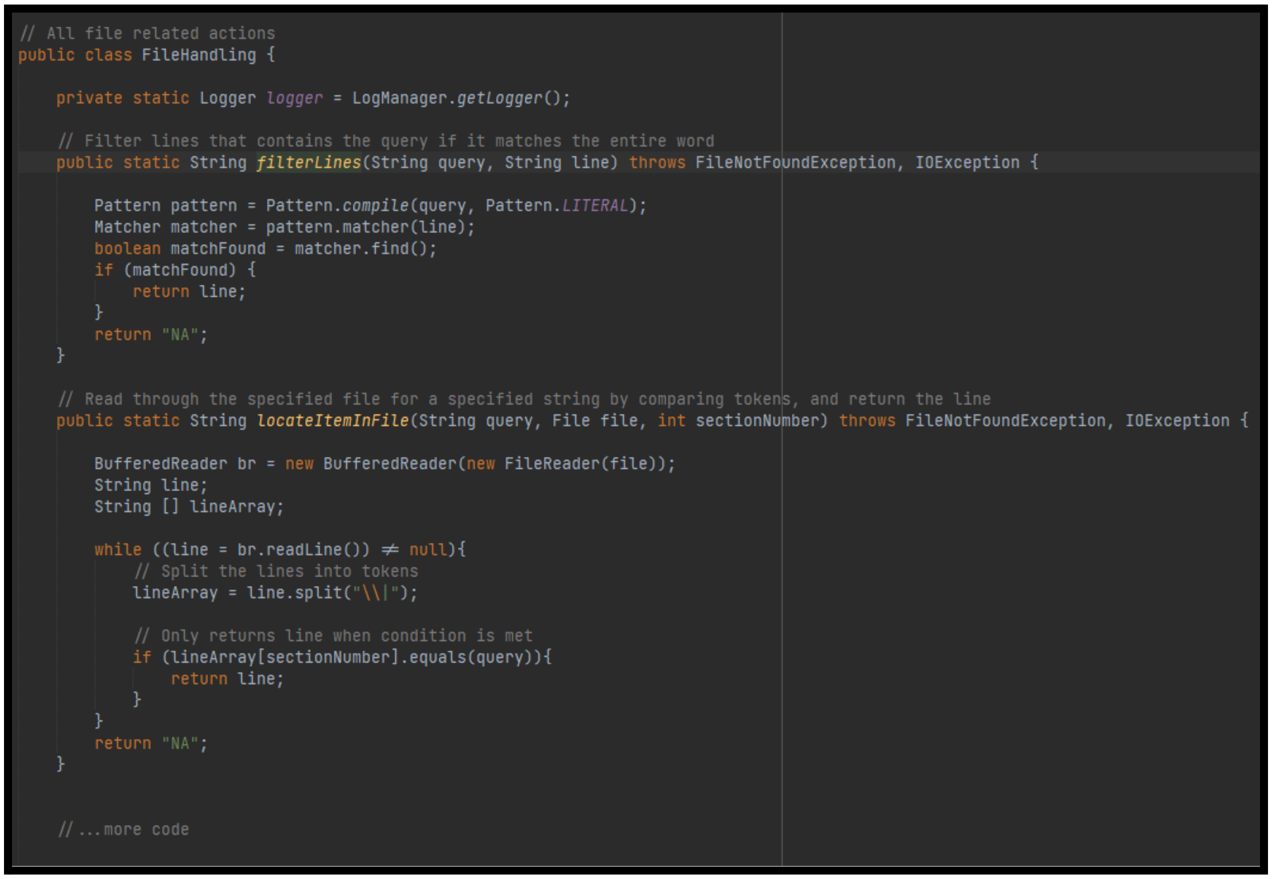
```java
public class WelcomePage extends javax.swing.JFrame {

    private static Logger logger = LogManager.getLogger();

    public WelcomePage() {
        initComponents();
        setTitle("APU Cafeteria Ordering System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void custLoginOptionActionPerformed(java.awt.event.ActionEvent evt) {
        CustomerLogin custLogin = new CustomerLogin();
        custLogin.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Customer Login page.");
    }

    private void custRegOptionActionPerformed(java.awt.event.ActionEvent evt) {
        CustomerRegistration custRegister = new CustomerRegistration();
        custRegister.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Customer Registration page.");
    }

    private void mgrRegOptionActionPerformed(java.awt.event.ActionEvent evt) {
        ManagerRegistration mgrRegister = new ManagerRegistration();
        mgrRegister.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Manager Registration page.");
    }

    private void mgrLoginOptionActionPerformed(java.awt.event.ActionEvent evt) {
        ManagerLogin mgrLogin = new ManagerLogin();
        mgrLogin.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Manager Login page.");
    }
```

Figure 3.1.1 - Single Responsibility Example

An example of this is the WelcomePage class which is only used to redirect the users to different pages upon launching the program. The pages that the users can be redirected to in this case includes the customer registration, customer login, manager registration and manager login pages. Consequently, the design of the class is easier to understand and maintain compared to combining various functionalities into one as that would be significantly more messy and difficult to understand, especially when several developers are collaborating for a single project.

3.1.2 - DRY (Don't Repeat Yourself) Principle

DRY is a principle that is most often implemented by developers to reduce the amount of redundant code present within the program. In software development, it aims to reduce repeating patterns, duplication of code by prioritizing abstraction and avoiding redundancy as a whole (Muldrow, 2020). Hence, this principle is applied by placing the code within methods from classes that can be used in various different pages throughout the classes without having to be rewritten entirely.



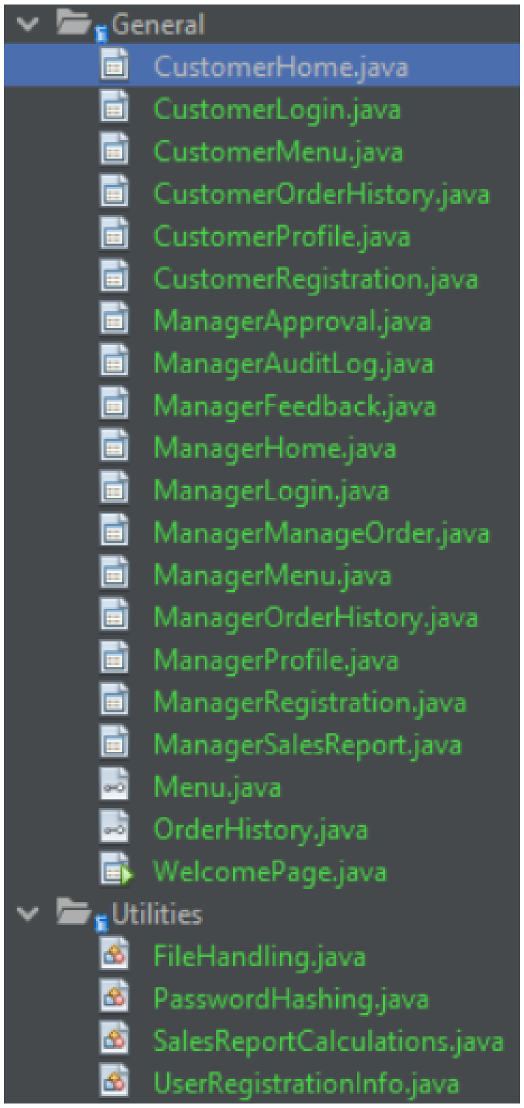
```java
// All file related actions
public class FileHandling {

    private static Logger logger = LogManager.getLogger();

    // Filter lines that contains the query if it matches the entire word
    public static String filterLines(String query, String line) throws FileNotFoundException, IOException {

        Pattern pattern = Pattern.compile(query, Pattern.LITERAL);
        Matcher matcher = pattern.matcher(line);
        boolean matchFound = matcher.find();
        if (matchFound) {
            return line;
        }
        return "NA";
    }

    // Read through the specified file for a specified string by comparing tokens, and return the line
    public static String locateItemInFile(String query, File file, int sectionNumber) throws FileNotFoundException, IOException {

        BufferedReader br = new BufferedReader(new FileReader(file));
        String line;
        String [] lineArray;

        while ((line = br.readLine()) ≠ null){
            // Split the lines into tokens
            lineArray = line.split("\\|");

            // Only returns line when condition is met
            if (lineArray[sectionNumber].equals(query)){
                return line;
            }
        }
        return "NA";
    }


    //...more code
```

Figure 3.1.2 - DRY Principle Example

The code above is an example of a class that is used in many places for file handling purposes. By creating objects of this in order to use the required methods in other classes, the code redundancy was greatly reduced. Furthermore, the application of this principle has also enabled more efficient programming as other classes that require similar methods can simply create an object of this class in order to use the methods instead of creating them again from scratch.

3.2 Modularity

Modularity in OOP is basically breaking up something complex into manageable pieces, for example smaller modules in order to help people understand complex systems.



```
General
    CustomerHome.java
    CustomerLogin.java
    CustomerMenu.java
    CustomerOrderHistory.java
    CustomerProfile.java
    CustomerRegistration.java
    ManagerApproval.java
    ManagerAuditLog.java
    ManagerFeedback.java
    ManagerHome.java
    ManagerLogin.java
    ManagerManageOrder.java
    ManagerMenu.java
    ManagerOrderHistory.java
    ManagerProfile.java
    ManagerRegistration.java
    ManagerSalesReport.java
    Menu.java
    OrderHistory.java
    WelcomePage.java
Utilities
    FileHandling.java
    PasswordHashing.java
    SalesReportCalculations.java
    UserRegistrationInfo.java
```

Figure 3.2 - List of packages, classes and interfaces in the project

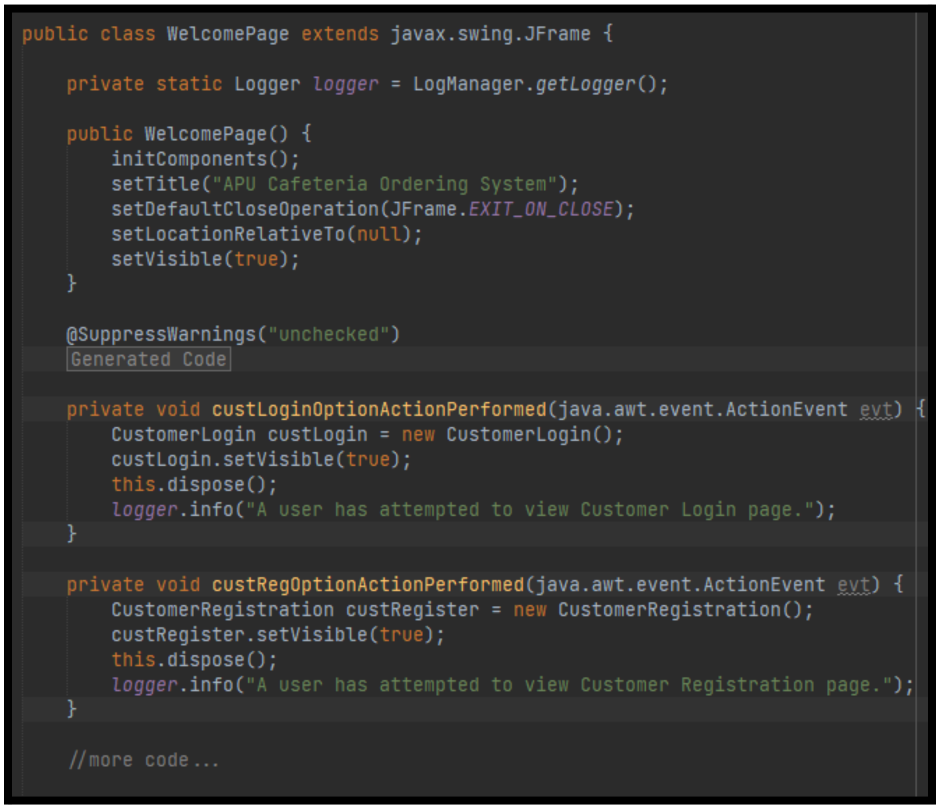
Based on the list above, the APU Cafeteria Ordering System consists of many smaller modules that represent separate functions. In this way, stability of the code while making changes to individual sections can be maintained as the new changes are hidden from customers and can be tested without affecting the existing software. Besides that, it also allows developers to quickly create modules that run in parallel. Once all of the modules have been created, they can be combined via an interface to form a suite which saves time. (Vats, 2022)

3.3 Inheritance

Inheritance is another fundamental concept of OOP used to derive one class from another class. This would usually result in a hierarchy of classes that share various attributes and methods (Janssen, 2017). In layman's terms it can be interpreted as an "IS-A" relationship between classes.

3.3.1 - Extending Regular or Abstract Classes

A common way of implementing inheritance in Java is by using the "extend" keyword to to inherit the code from another class.



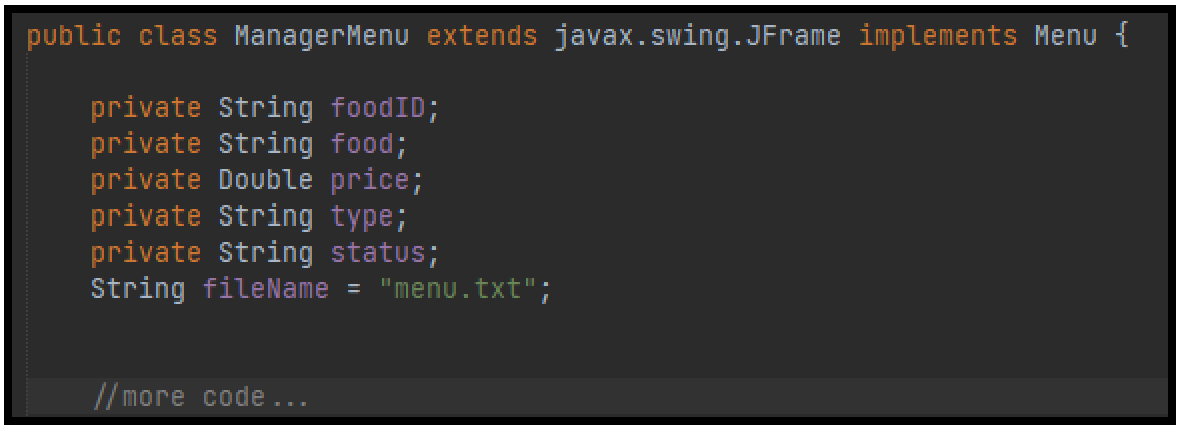
```
public class WelcomePage extends javax.swing.JFrame {

    private static Logger logger = LogManager.getLogger();

    public WelcomePage() {
        initComponents();
        setTitle("APU Cafeteria Ordering System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @SuppressWarnings("unchecked")
    Generated Code

    private void custLoginOptionActionPerformed(java.awt.event.ActionEvent evt) {
        CustomerLogin custLogin = new CustomerLogin();
        custLogin.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Customer Login page.");
    }

    private void custRegOptionActionPerformed(java.awt.event.ActionEvent evt) {
        CustomerRegistration custRegister = new CustomerRegistration();
        custRegister.setVisible(true);
        this.dispose();
        logger.info("A user has attempted to view Customer Registration page.");
    }

    //more code...
```

Figure 3.3.1 - Inheritance of Jframe Class

For instance, by extending the Jframe class the WelcomePage class can access all the methods related within the Jframe class which can be used for the GUI without having to type it out manually or creating an object of that class to access the methods.

3.3.2 - Implementing Interfaces in Java

Interfaces are considered as a completely abstract class that has no implementation. However, they can still be used to achieve inheritance in Java. Interfaces are mainly used to define the operations for classes. The implementation itself is done by the subclass of the interface.



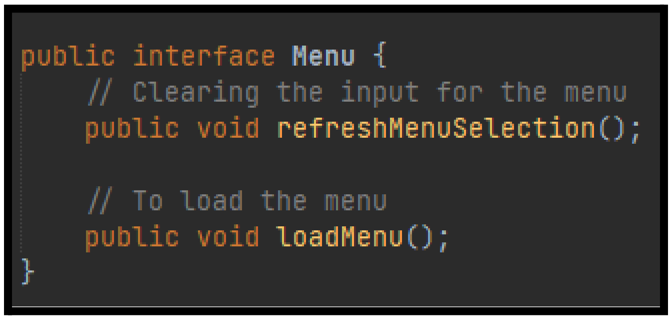
```java
public class ManagerMenu extends javax.swing.JFrame implements Menu {

    private String foodID;
    private String food;
    private Double price;
    private String type;
    private String status;
    String fileName = "menu.txt";


    //more code...
```

Figure 3.3.2 - Implementing Menu Interface

The code above is an example of inheritance whereby the ManagerMenu class implements the code from the Menu interface.

3.4 Abstraction

Abstraction is an OOP concept used to maintain required attributes and hides irrelevant information (Hartman, 2021). It is mainly used to keep unnecessary information hidden from specific users which also reduces programming complexity and efforts.



```java
public interface Menu {
    // Clearing the input for the menu
    public void refreshMenuSelection();

    // To load the menu
    public void loadMenu();
}
```

Figure 3.4 - Abstraction via Menu Interface

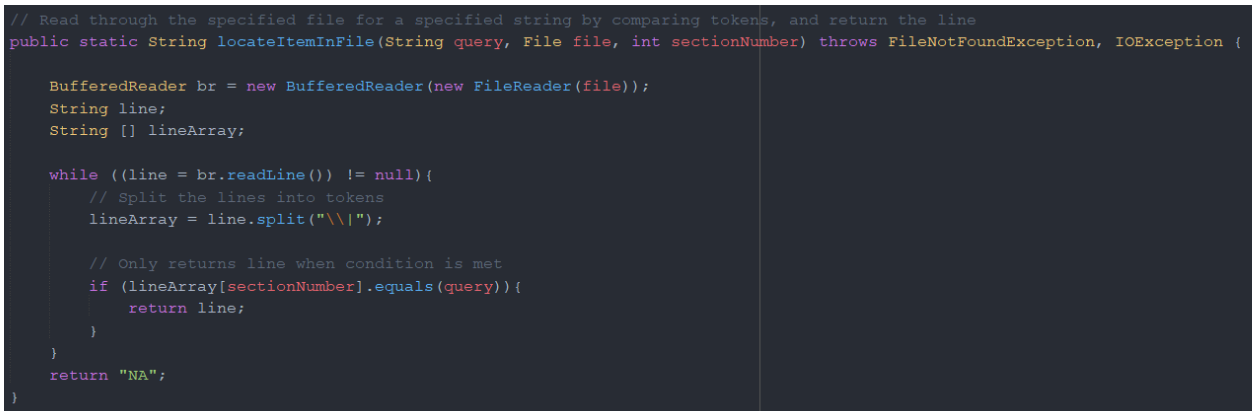
For example, the Menu interface above is used for the purpose of listing out any methods that must be present within menu related classes in the program. Abstraction is achieved by using an interface as the methods and information about the implementation is hidden from the user as all the methods are completely abstract (tutorialspoint, 2022).

3.5 Polymorphism

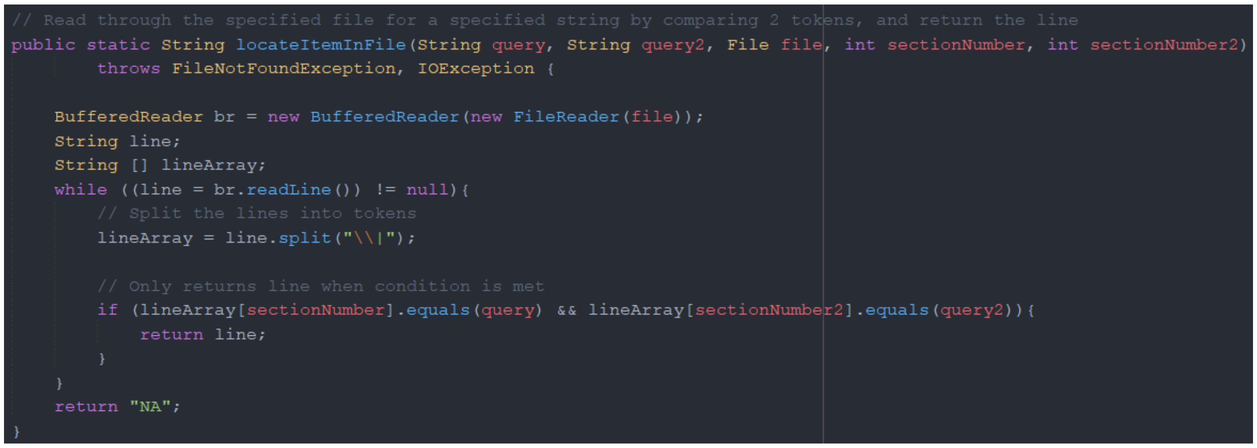
Polymorphism, which means "many forms," occurs when many classes are related by inheritance. Its concept is that through the same interface, one can access objects of various types (Janssen, 2022).

3.5.1 - Method Overloading

Overloading is a way to implement compile time polymorphism where multiple methods can have the same name but with different parameters count or type (Gupta, 2022).



```java
// Read through the specified file for a specified string by comparing tokens, and return the line
public static String locateItemInFile(String query, File file, int sectionNumber) throws FileNotFoundException, IOException {

    BufferedReader br = new BufferedReader(new FileReader(file));
    String line;
    String [] lineArray;

    while ((line = br.readLine()) != null){
        // Split the lines into tokens
        lineArray = line.split("\\|");

        // Only returns line when condition is met
        if (lineArray[sectionNumber].equals(query)){
            return line;
        }
    }
    return "NA";
}
```



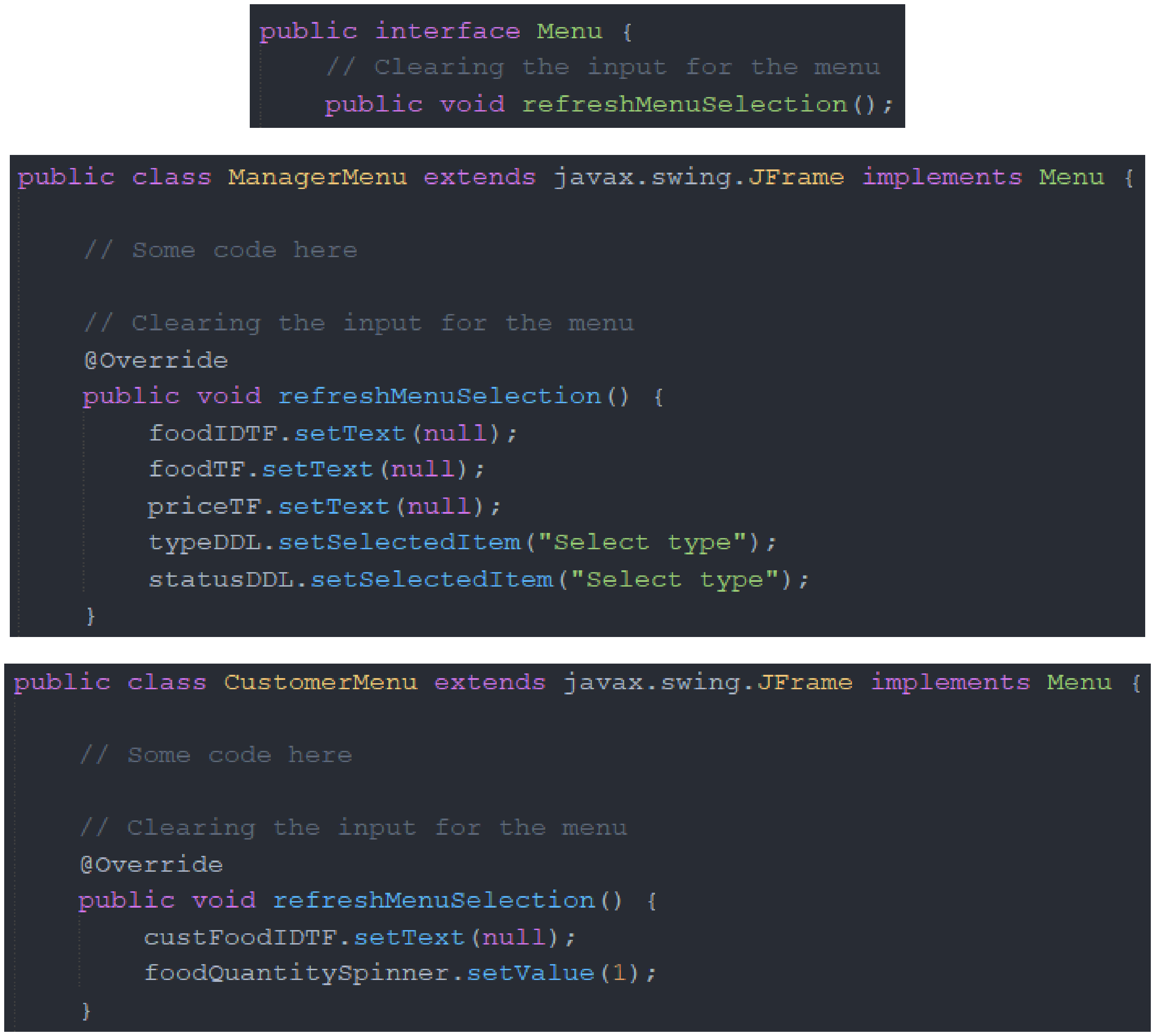
```java
// Read through the specified file for a specified string by comparing 2 tokens, and return the line
public static String locateItemInFile(String query, String query2, File file, int sectionNumber, int sectionNumber2)
        throws FileNotFoundException, IOException {

    BufferedReader br = new BufferedReader(new FileReader(file));
    String line;
    String [] lineArray;
    while ((line = br.readLine()) != null){
        // Split the lines into tokens
        lineArray = line.split("\\|");

        // Only returns line when condition is met
        if (lineArray[sectionNumber].equals(query) && lineArray[sectionNumber2].equals(query2)){
            return line;
        }
    }
    return "NA";
}
```

Figure 3.5.1 - Method Overloading: locateItemInFile()

The code above shows two methods with the same name locateItemInFile() but with a different signature due to different parameters count, with both methods performing similarly. By doing so, this increases code maintainability and code readability.

3.5.2 - Method Overriding

Overriding is done by supplying a different implementation of a method that exists in the superclass, and it must have the same method signature. It is a way to implement runtime polymorphism which can be achieved through implementing inheritance (Gupta, 2022).



```java
public interface Menu {
    // Clearing the input for the menu
    public void refreshMenuSelection();
```

```java
public class ManagerMenu extends javax.swing.JFrame implements Menu {

    // Some code here

    // Clearing the input for the menu
    @Override
    public void refreshMenuSelection() {
        foodIDTF.setText(null);
        foodTF.setText(null);
        priceTF.setText(null);
        typeDDL.setSelectedItem("Select type");
        statusDDL.setSelectedItem("Select type");
    }
```

```java
public class CustomerMenu extends javax.swing.JFrame implements Menu {

    // Some code here

    // Clearing the input for the menu
    @Override
    public void refreshMenuSelection() {
        custFoodIDTF.setText(null);
        foodQuantitySpinner.setValue(1);
    }
```

Figure 3.5.2 - Method Overriding: refreshMenuSelecton()

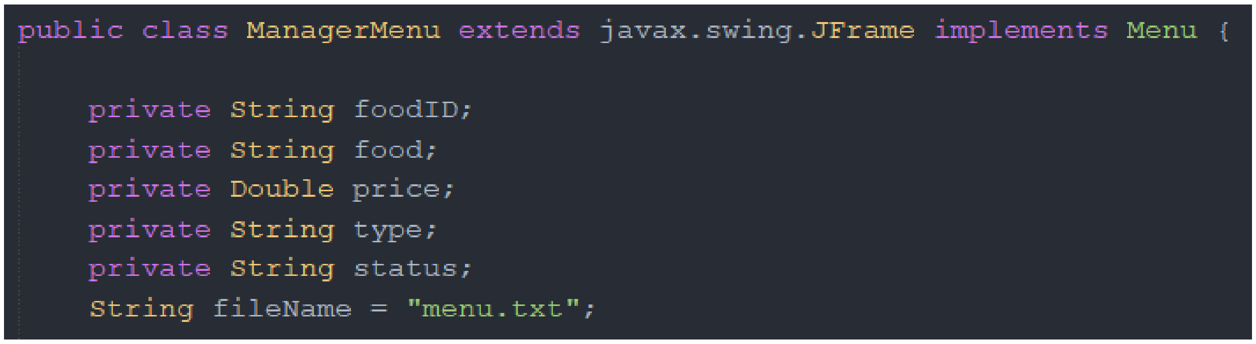
Through inheritance by implementing an interface, the refreshMenuSelecton() behaves differently in two different classes which is ManagerMenu and CustomerMenu. In this way, both classes can provide their own specific implementation to an inherited method without modifying the parent class code.

3.6 Encapsulation

In OOP, encapsulation means bundling data with the methods that operate on that data to hide the values or state of a structured data object within a class, preventing unauthorised parties from accessing them directly (Braunschweig, 2018).

3.6.1 - Access Modifiers

One way to achieve encapsulation is using access modifiers that are used to set the access level for attributes, constructors, classes and attributes.



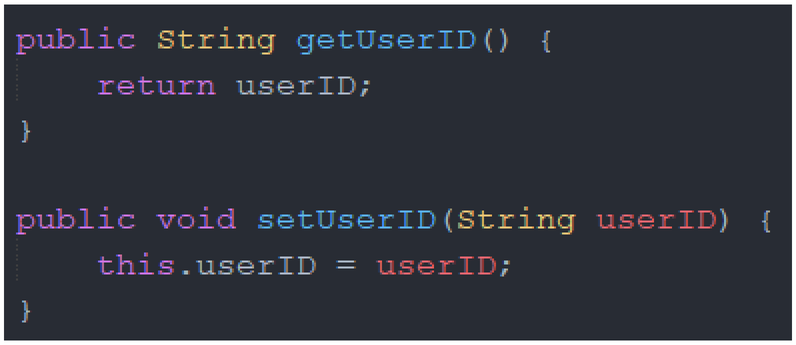
```
public class ManagerMenu extends javax.swing.JFrame implements Menu {

    private String foodID;
    private String food;
    private Double price;
    private String type;
    private String status;
    String fileName = "menu.txt";
```

Figure 3.6.1 - Private access modifiers

For instance, the food ID, food and other attributes are declared as private access modifiers, which means that the variables can only be accessed within the same class (w3schools, 2022). This increases the security of data as class attributes cannot be modified externally.

3.6.2 - Getters and Setters

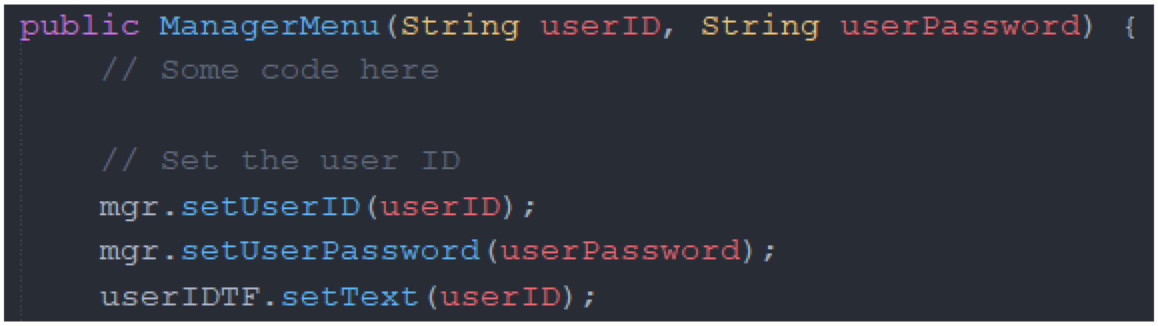
To access private attributes, get and set methods can be used to access them.



```java
public String getUserID() {
    return userID;
}

public void setUserID(String userID) {
    this.userID = userID;
}
```

Figure 3.6.2.1 - getUserID() and setUserID() body

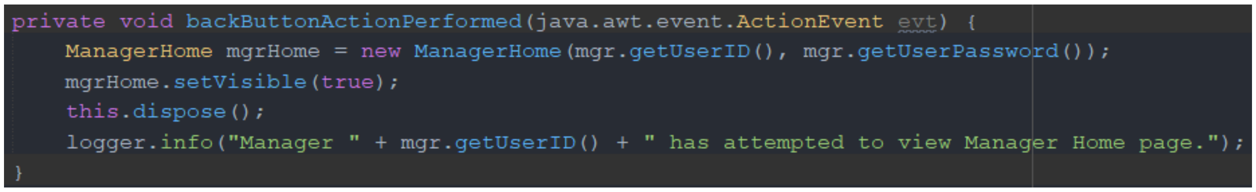
In the code below, multiple setters can be seen, for instance the usage of setUserID().



```java
public ManagerMenu(String userID, String userPassword) {
    // Some code here

    // Set the user ID
    mgr.setUserID(userID);
    mgr.setUserPassword(userPassword);
    userIDTF.setText(userID);
```

Figure 3.6.2.2 - setUserID() implementation

In the code below, a getter can be seen, for instance getUserID().



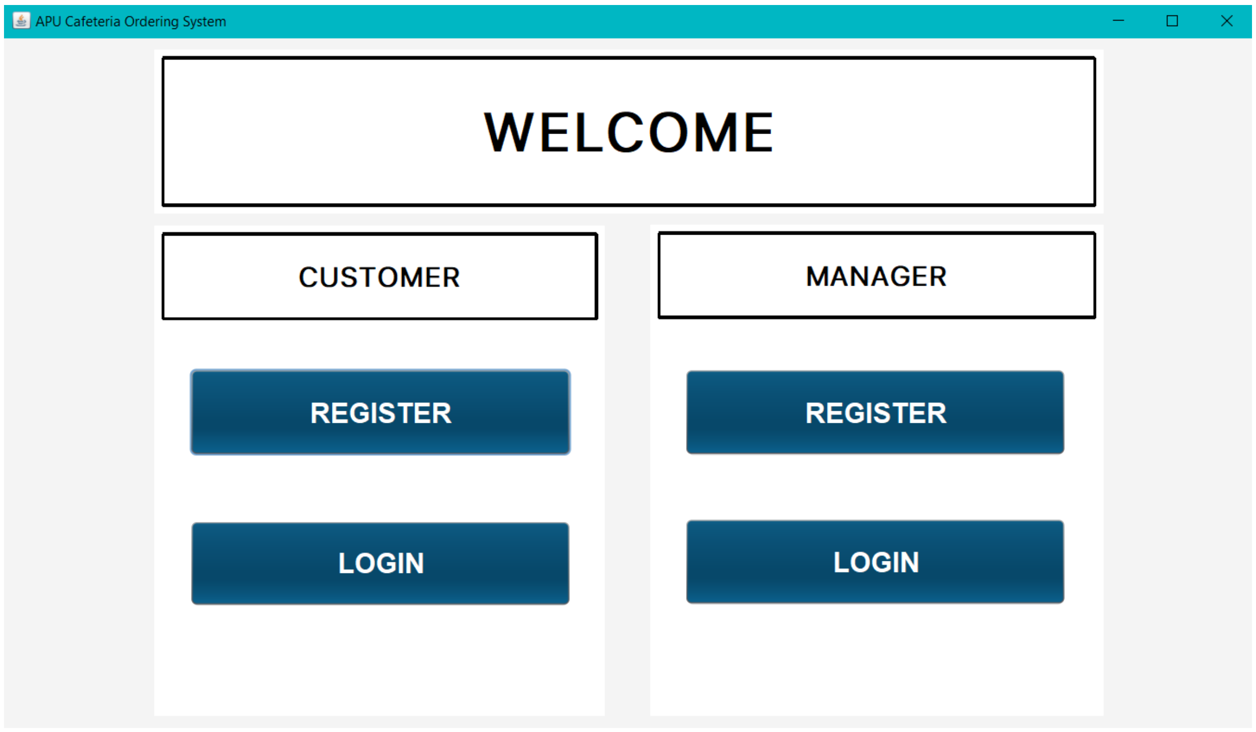
```java
private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {
    ManagerHome mgrHome = new ManagerHome(mgr.getUserID(), mgr.getUserPassword());
    mgrHome.setVisible(true);
    this.dispose();
    logger.info("Manager " + mgr.getUserID() + " has attempted to view Manager Home page.");
}
```

Figure 3.6.2.3 - getUserID() implementation

Using getters and setter allows one to access and update the value of a private variable, giving a better control of class attributes and methods. In this way, the programmer can also change a part of the code without affecting other parts as class attributes can be made read-only by using only the get method, or write-only by using only the set method.

4.0 User Interface

4.1 Welcome Page

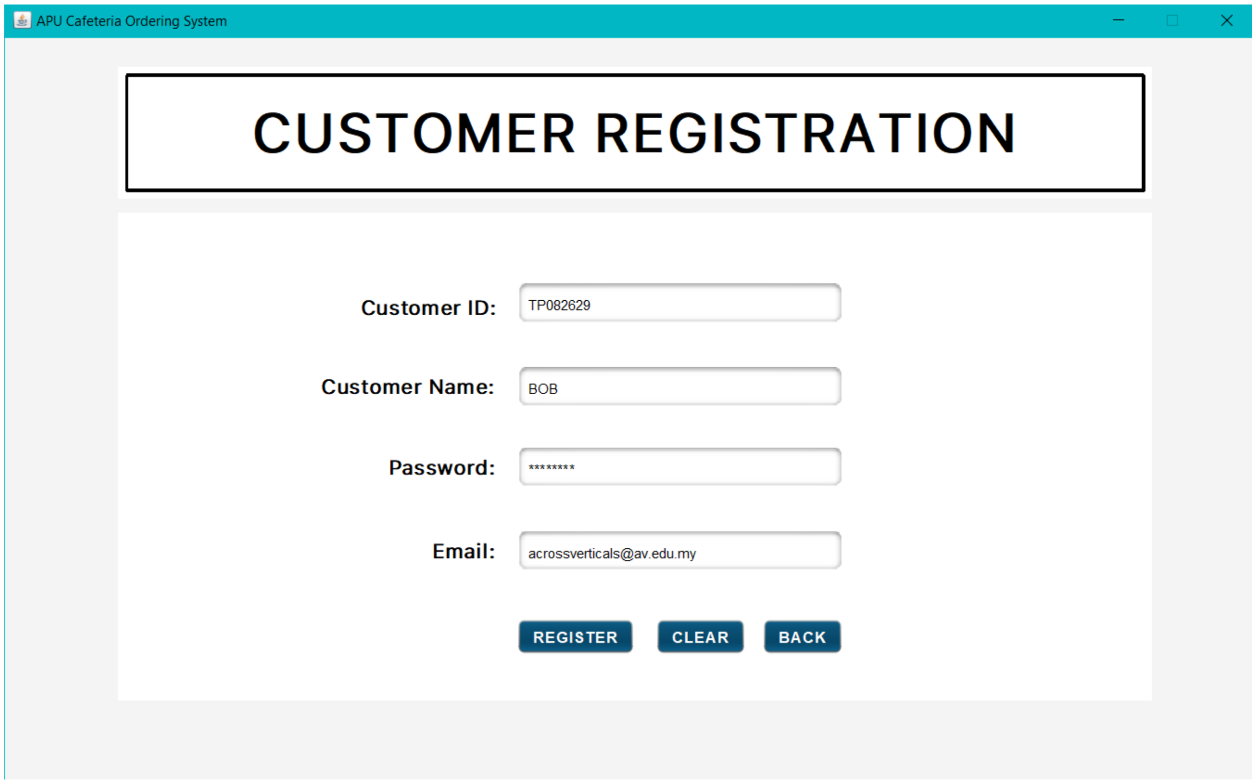


This is the first page the user will see when they start the APU Cafeteria Ordering System. The user can choose to register or login as a customer or manager by clicking on the respective buttons. Each button will lead to a different page.

4.2 Customer Pages

Pages that are related to the customer role in the APU Cafeteria Ordering System.

4.2.1 - Customer Registration



Customer ID: TP082629

Customer Name: BOB

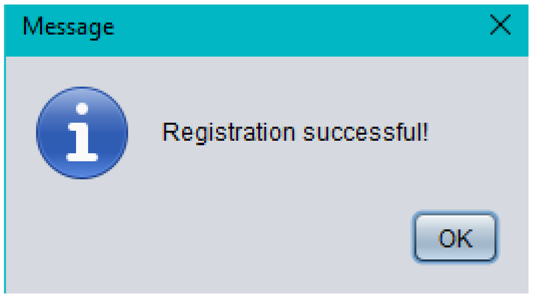
Password: ********

Email: acrossverticals@av.edu.my

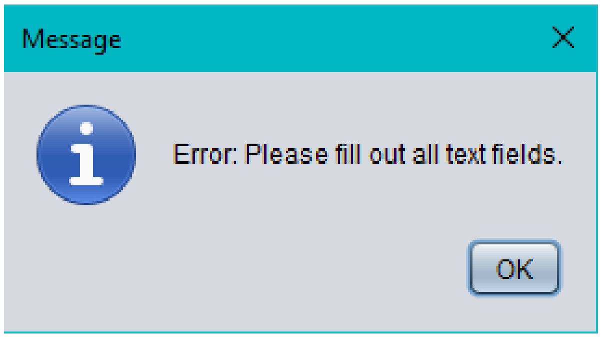
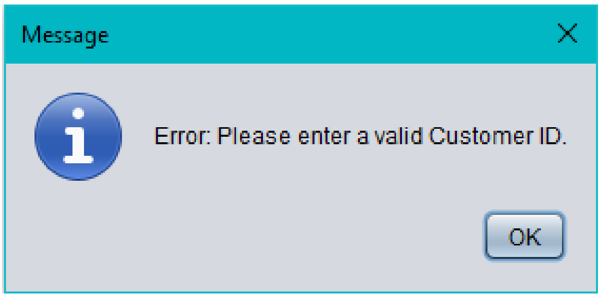
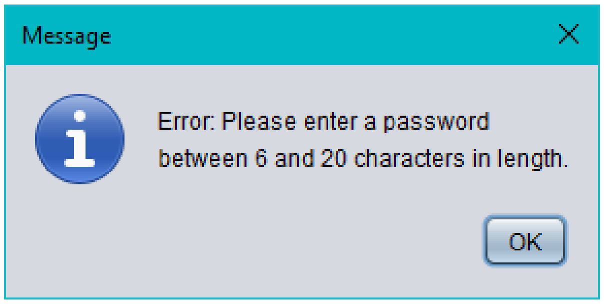
Figure 4.2.1 - Customer Registration Page

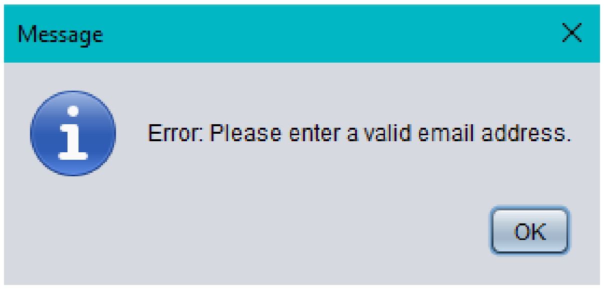
This page will show when the user clicks on the "REGISTER" button on the "CUSTOMER" section in the Welcome Page. The user can input their Customer ID, name, password and email in the text fields shown. The password typed will be shown in asterisks (*) to enhance security. The user can choose to register by clicking on the "REGISTER" button or clear the text fields by clicking on the "CLEAR" button. If the user wishes to return to the Welcome Page, they could click on the "BACK" button. Once the user has successfully registered, a prompt will show up, and the details would be appended to the custAccount.txt file.

Below is the scenario of the prompt message:

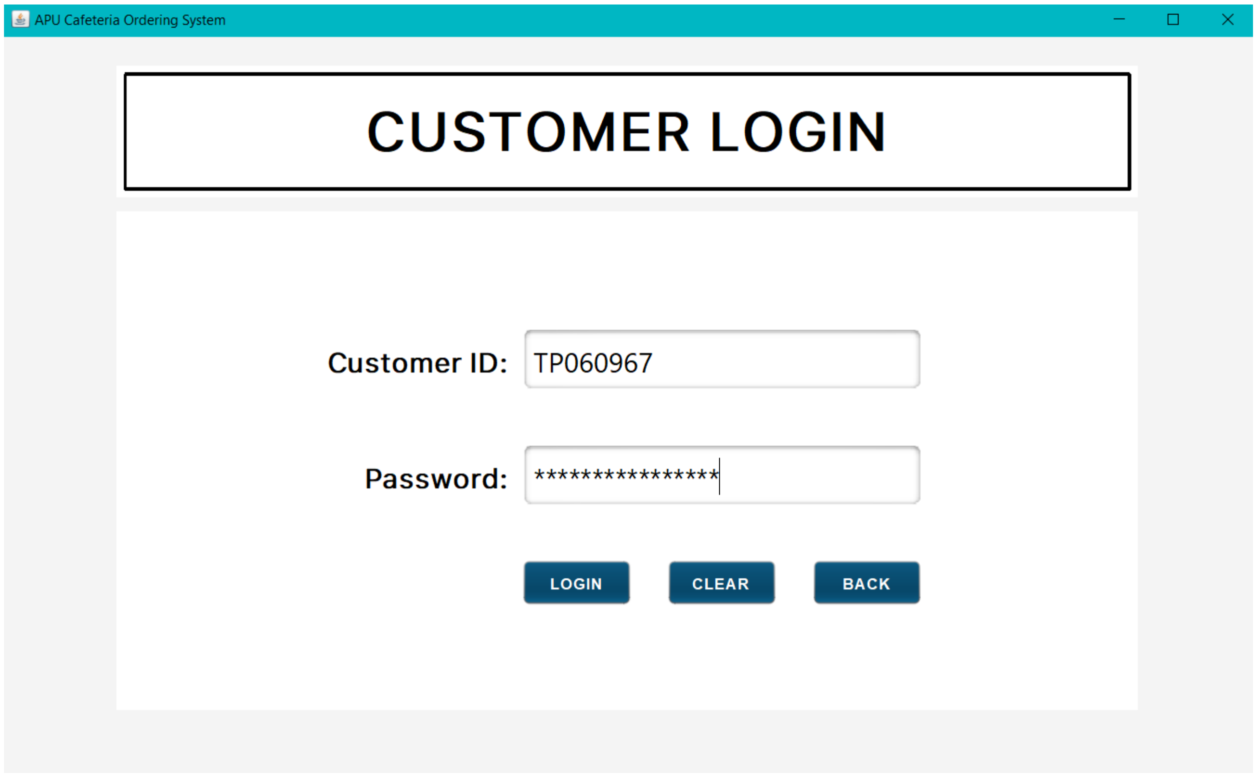
Prompt Messages	**Trigger Conditions**
 Message Registration successful!	When all the conditions are met accordingly after the "Register" button is clicked.

Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	**Trigger Conditions**
 Message Error: Please fill out all text fields.	When any of the registration text fields are empty upon pressing the "Register" button.
 Message Error: Please enter a valid Customer ID.	When the Customer ID does not match the correct format after clicking the "Register" button. (E.g. TP012345)
 Message Error: Please enter a password between 6 and 20 characters in length.	The password entered is not in between 6 and 20 characters in length after the "Register" button is clicked.

 Message Error: Please enter a valid email address. OK	The email address entered by the user does not match the standard email pattern. (E.g. myemail@mail.com)

4.2.2 - Customer Login



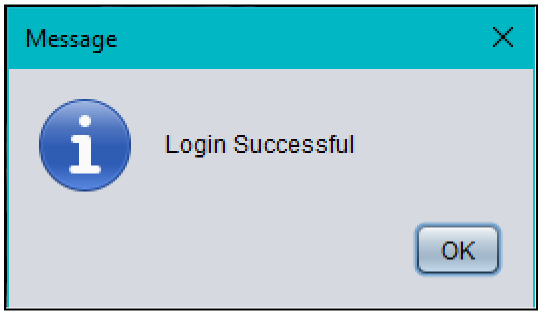
Customer ID: TP060967

Password: ****************

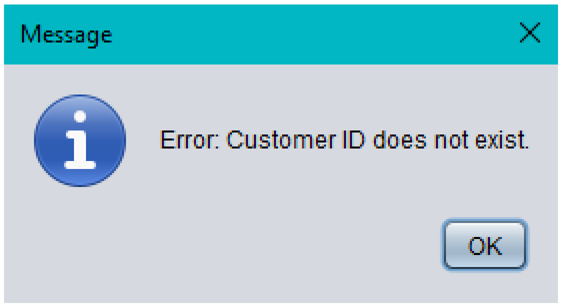
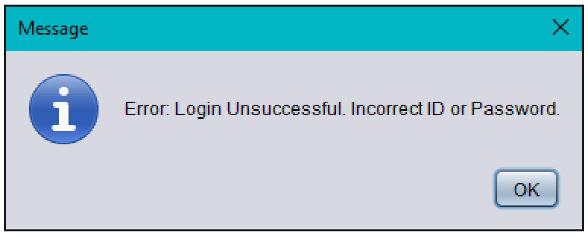
Figure 4.2.2 - Customer Login Page

This page will show when the user clicks on the “LOGIN” button on the “CUSTOMER” section in the Welcome Page. The user can input their registered Customer ID and their password that is stored in custAccount.txt in the text fields shown. The password typed will be shown in asterisks (*) to enhance security. The user can choose to login by clicking on the “LOGIN” button or clear the text fields by clicking on the “CLEAR” button. If the user wishes to return to the Welcome Page, they could click on the “BACK” button.

Below is the scenario of the prompt message:

Prompt Messages	**Trigger Conditions**
 Message Login Successful 	When the Customer ID and password entered by the user matches one of the lines within the text file after the "Login" button is clicked.

Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	**Trigger Conditions**
 Message Error: Customer ID does not exist. 	When the Customer ID keyed in by the user does not exist within the text file.
 Message Error: Login Unsuccessful. Incorrect ID or Password. 	Occurs when the password entered by the user does not match the Customer ID within the text file.

4.2.3 - Customer Home

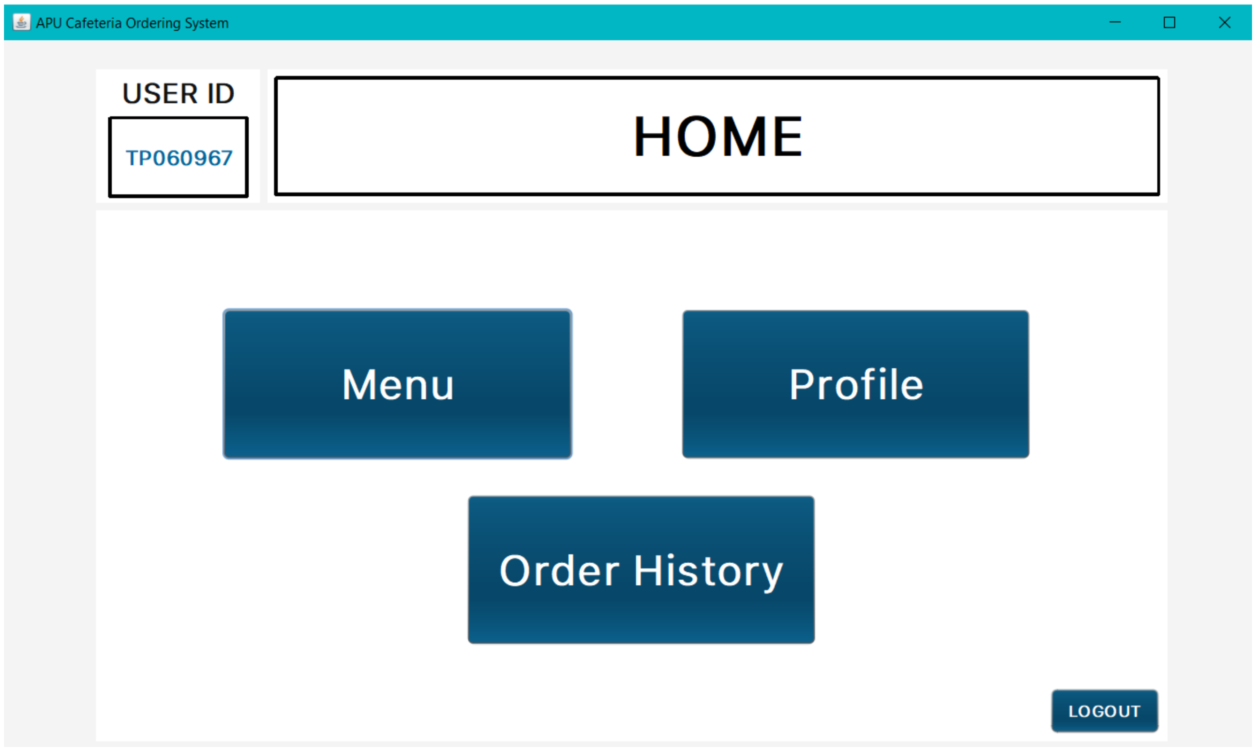
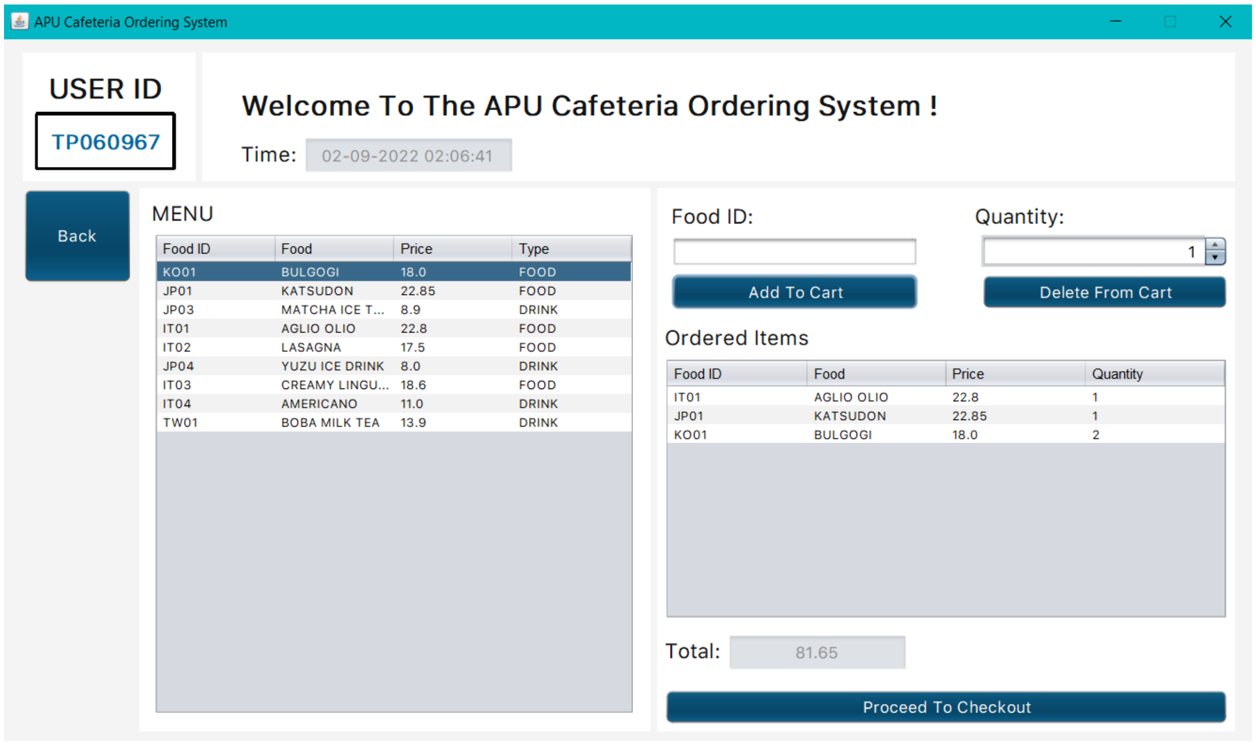


Figure 4.2.3 - Customer Home Page

This is the first page the user will see when they have successfully logged in as a user. The user can choose to order something from the menu by clicking on the "Menu" button, view their profile details and top up using the "Profile" button, and view their order history using the "Order History" button. Each button will lead to a different page. The user can also log out to the Welcome Page by clicking on the "LOGOUT" button at the bottom right of the page.

4.2.4 - Customer Menu


APU Cafeteria Ordering System

USER ID

TP060967

Welcome To The APU Cafeteria Ordering System !

Time: 02-09-2022 02:06:41

Back

MENU

Food ID	Food	Price	Type
KO01	BULGOGI	18.0	FOOD
JP01	KATSUDON	22.85	FOOD
JP03	MATCHA ICE T...	8.9	DRINK
IT01	AGLIO OLIO	22.8	FOOD
IT02	LASAGNA	17.5	FOOD
JP04	YUZU ICE DRINK	8.0	DRINK
IT03	CREAMY LINGU...	18.6	FOOD
IT04	AMERICANO	11.0	DRINK
TW01	BOBA MILK TEA	13.9	DRINK

Food ID:

Quantity: 1

Add To Cart

Delete From Cart

Ordered Items

Food ID	Food	Price	Quantity
IT01	AGLIO OLIO	22.8	1
JP01	KATSUDON	22.85	1
KO01	BULGOGI	18.0	2

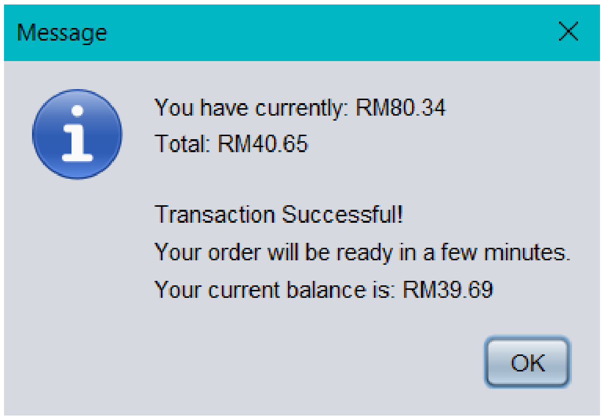
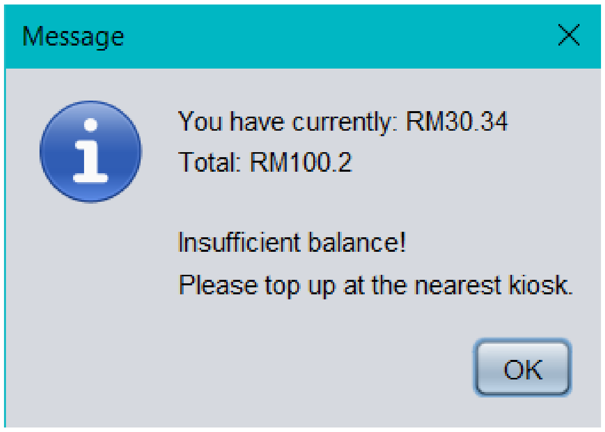
Total: 81.65

Proceed To Checkout

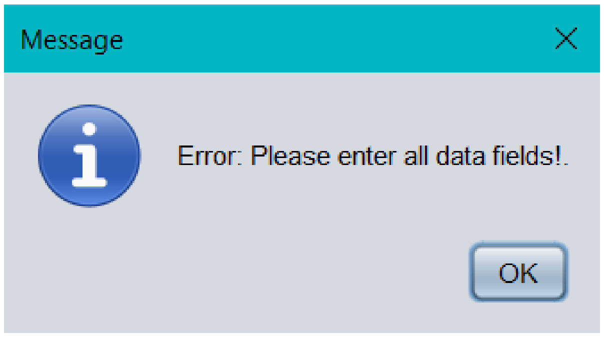
Figure 4.2.4 - Customer Menu Page

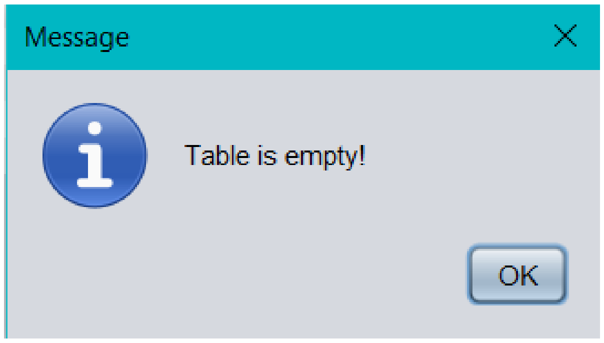
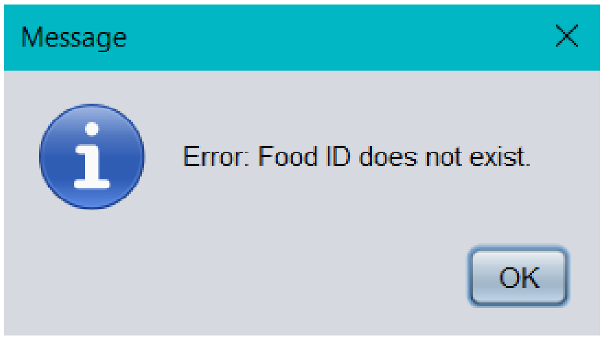
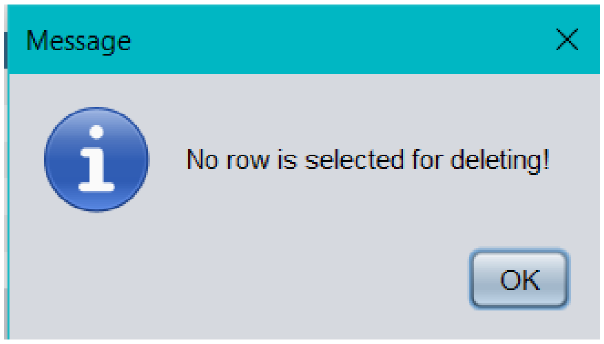
The top left of the page shows the User ID of the logged in user, while the top panel shows the time that the user has viewed the Customer Menu page. The Menu table will only show food and drinks that were marked as “AVAILABLE” in the menu.txt file. To add any item into the cart, the user could click on the items on the table or manually typing its Food ID to the “Food ID” text field. After picking a quantity, the user can choose to add the item to cart by clicking on the “Add To Cart” button or delete that item from the “Ordered Items” table by clicking on “Delete From Cart” button. The total of the current items in the cart will be generated when the user adds or deletes items in the “Ordered Items” table. Once the user is done with their order, they can click on “Proceed To Checkout”, where a prompt will inform the user if their transaction is successful or not. If the user wishes to return to the Customer Home Page, they could click on the “BACK” button.

Below are the 2 scenarios of the prompt message:

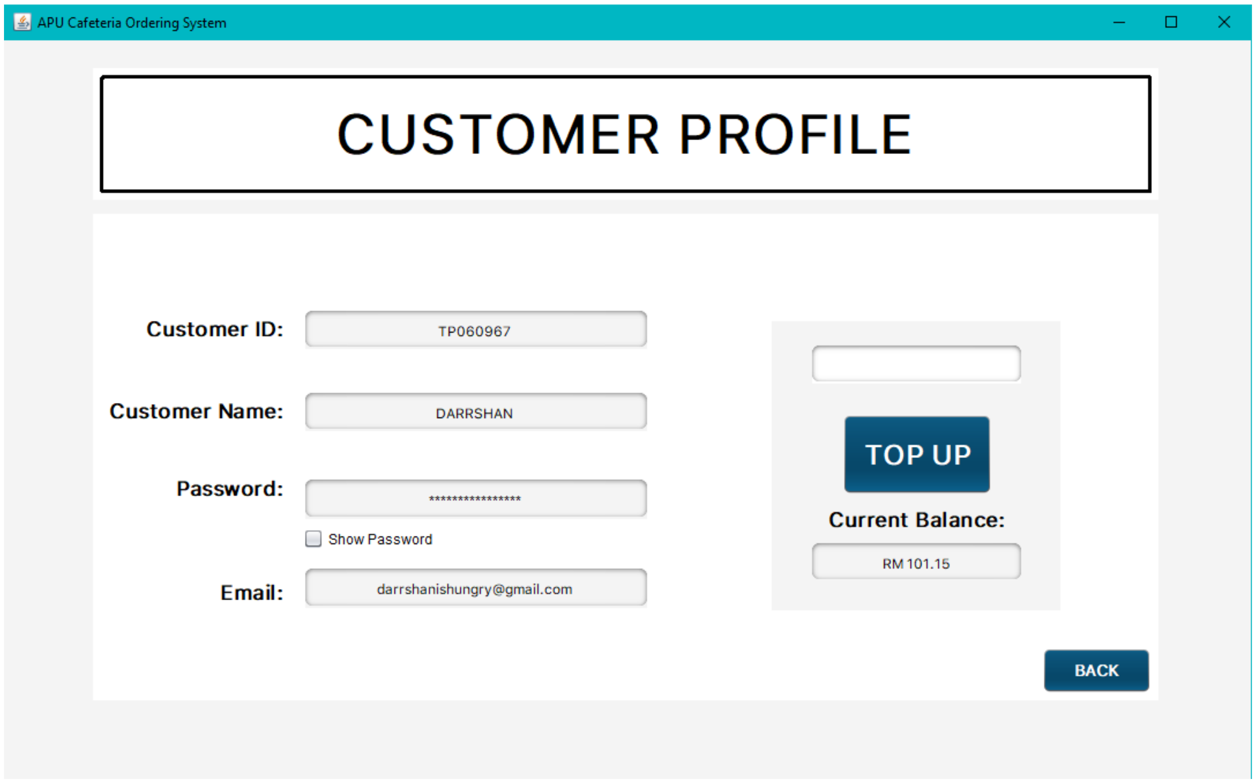
Prompt Messages	Trigger Conditions
 Message You have currently: RM80.34 Total: RM40.65 Transaction Successful! Your order will be ready in a few minutes. Your current balance is: RM39.69	When the user account balance is more than the total price of the item.
 Message You have currently: RM30.34 Total: RM100.2 Insufficient balance! Please top up at the nearest kiosk.	When the user account balance is less than the total price of the item.

Below are the error messages that could pop up due to the errors triggered by the user (Non-exception errors):

Error Messages	Trigger Conditions
 Message Error: Please enter all data fields!.	Data fields are empty when “Add To Cart” is clicked.

 Message Table is empty! OK	Ordered items are empty when “Delete To Cart” is clicked.
 Message Error: Food ID does not exist. OK	When the user inputs an invalid Food ID that is not in the “Menu” table.
 Message No row is selected for deleting! OK	When the user tries to delete a Food ID that is not in the “Ordered Items” table.

4.2.5 - Customer Profile



Customer ID: TP060967

Customer Name: DARRSHAN

Password: ****************

Show Password

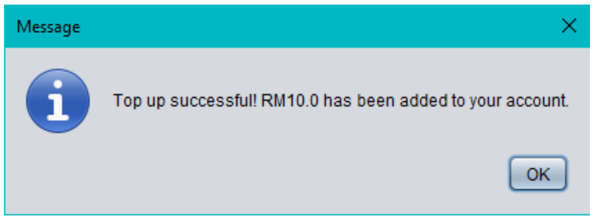
Email: darrshanishungry@gmail.com

Current Balance: RM 101.15

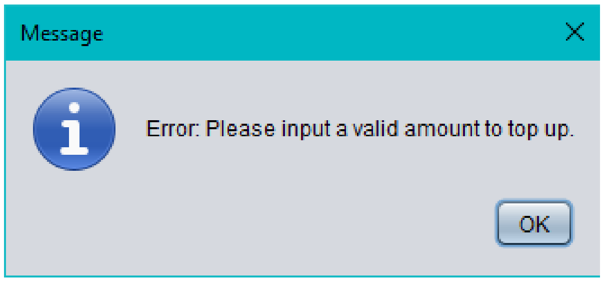
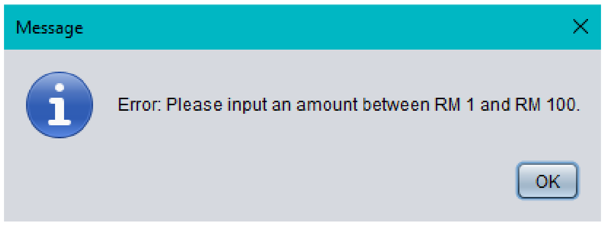
Figure 4.2.5 - Customer Profile Page

The customer profile page shows all the credentials of the logged in user. The credentials include the Customer ID, Customer Name, Password and Email address. There is also a small checkbox called "Show Password" that can be clicked to display the password of the user which is hidden by default. Aside from that there is also a small top up section that allows the user to update their account balance. In order to top up, the user will first need to key in the amount they would like to top up and click on the "TOP UP" button. If there are no errors, the user will be able view their updated balance right below the button. Finally, at the bottom right corner of the page is a "BACK" button which takes the Customer back to the Customer Home page.

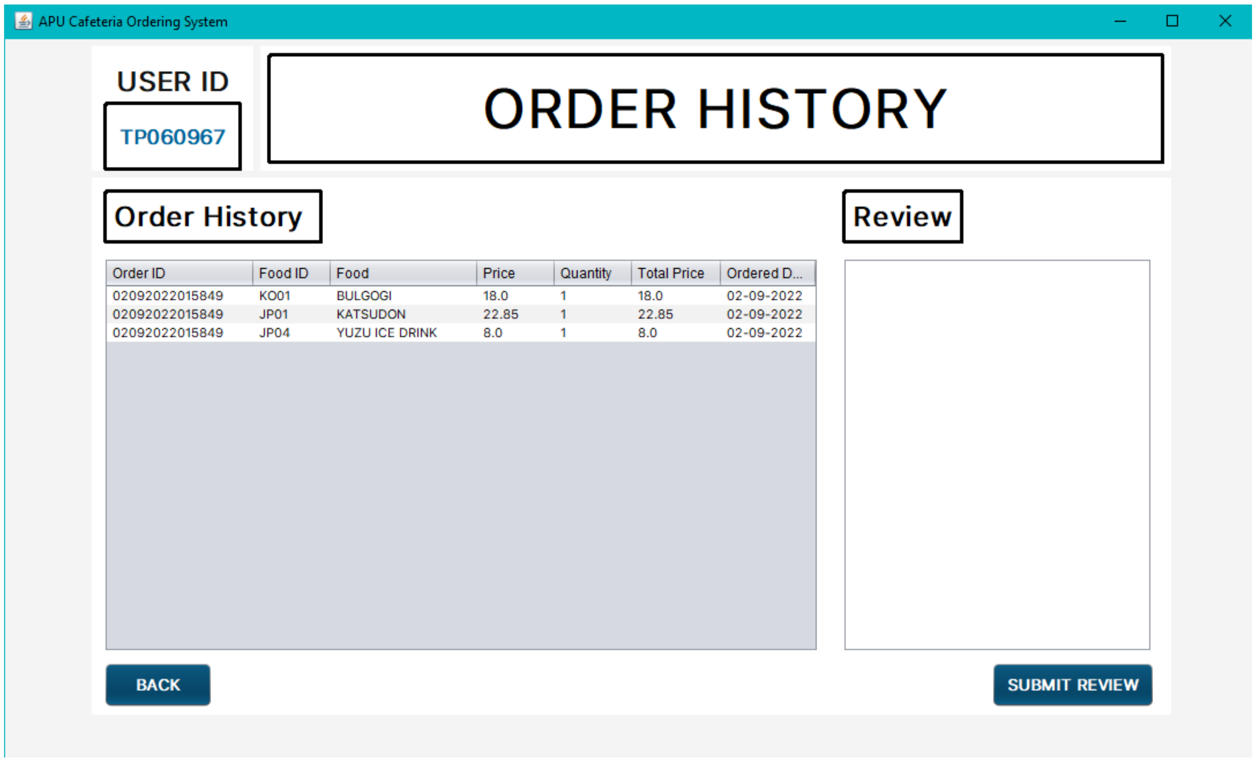
Below is the scenario of the prompt message:

Prompt Messages	Trigger Conditions
 Message Top up successful! RM10.0 has been added to your account. OK	Entering the top up amount and clicking the "TOP UP" button.

Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	Trigger Conditions
 Message Error: Please input a valid amount to top up. OK	When the top up amount keyed in by the user does not include is not a proper amount or includes characters aside from numbers and a decimal point.
 Message Error: Please input an amount between RM 1 and RM 100. OK	The top up amount entered by the user is not within the range of RM 1 and RM 100.

4.2.6 - Customer Order History

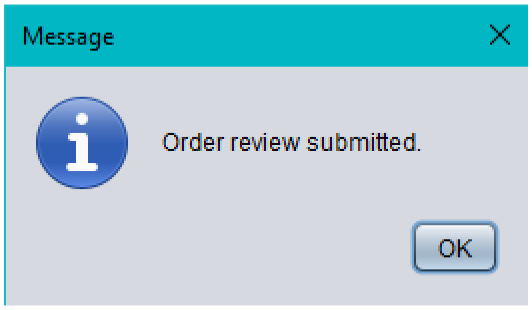


Order ID	Food ID	Food	Price	Quantity	Total Price	Ordered D...
02092022015849	KO01	BULGOGI	18.0	1	18.0	02-09-2022
02092022015849	JP01	KATSUDON	22.85	1	22.85	02-09-2022
02092022015849	JP04	YUZU ICE DRINK	8.0	1	8.0	02-09-2022

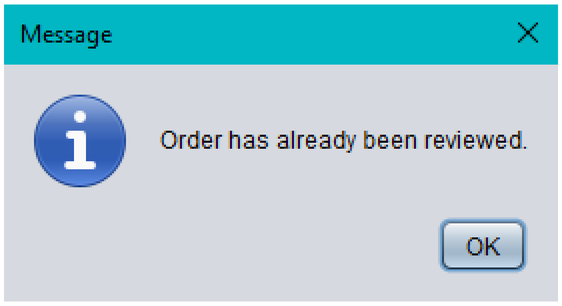
Figure 4.2.6 - Customer Order History Page

The top left of the page displays the ID of the Customer that is currently logged in and below are 2 sections which are the Order History and Review sections. The Order History section consists of a table with an Order History table that displays all the past orders of the user line by line. The columns within the table consist of the Order ID, Food ID, Food, Price, Quantity, Total Price and Order Date. The Review section on the other hand has a text area for the user to enter their review of a particular order from the Order History table. In order to write a review on the order, the user will first need to select an order within the table by clicking on one of the rows. Next, the user will have to key in their review and click on the “SUBMIT REVIEW” button for the review to go through. If the user wishes to return back to the Customer Home page, they may click the “BACK” button to be redirected.

Below is the scenario of the prompt message:

Prompt Messages	Trigger Conditions
 Message Order review submitted. OK	Prompts after the user selects an order from the table, writes a review and clicks on the "SUBMIT REVIEW" button.

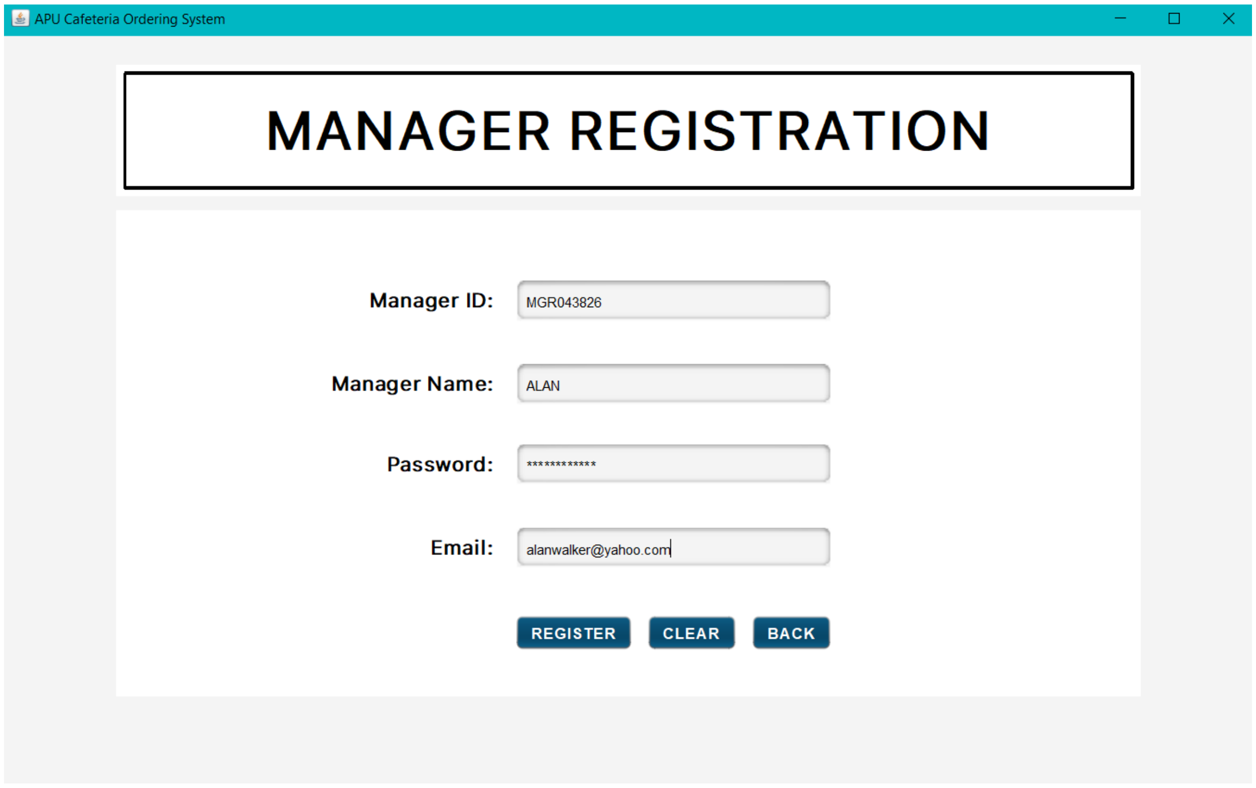
Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	Trigger Conditions
 Message Order has already been reviewed. OK	When a review is submitted for an order that has already been reviewed before.

4.3 Manager Pages

Pages that are related to the manager role in the APU Cafeteria Ordering System.

4.3.1 - Manager Registration

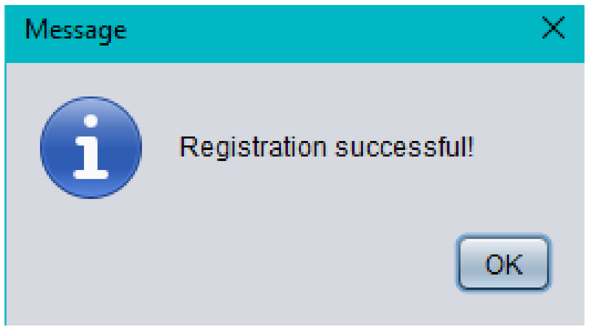


Manager ID:	MGR043826
Manager Name:	ALAN
Password:	************
Email:	alanwalker@yahoo.com

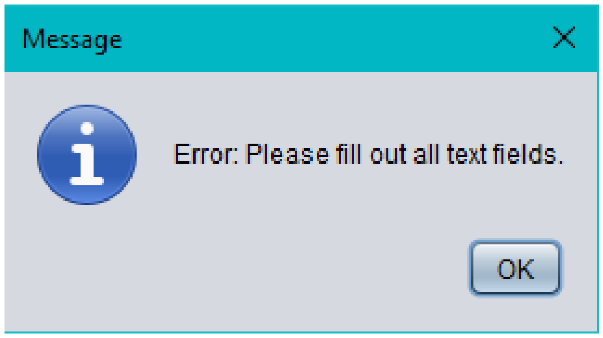
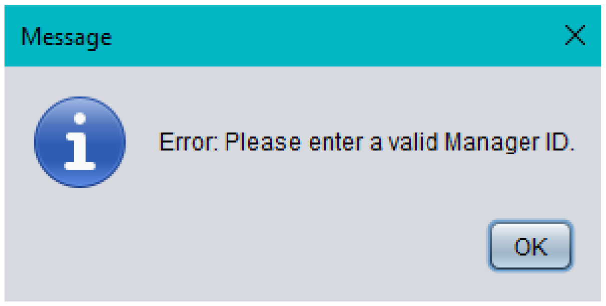
Figure 4.3.1 - Manager Registration Page

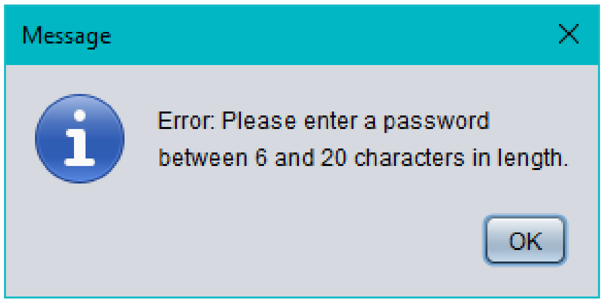
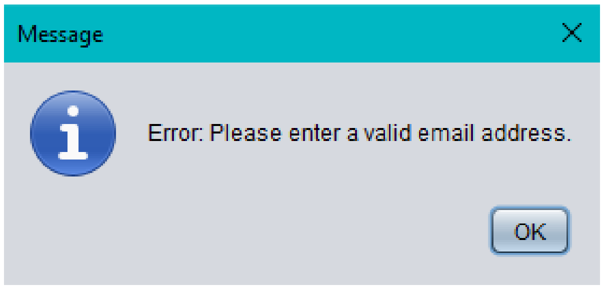
This page will show when the user clicks on the “REGISTER” button on the “MANAGER” section in the Welcome Page. The user can input their Manager ID, name, password and email in the text fields shown. The password typed will be shown in asterisks (*) to enhance security. The user can choose to register by clicking on the “REGISTER” button or clear the text fields by clicking on the “CLEAR” button. If the user wishes to return to the Welcome Page, they could click on the “BACK” button. Once the user has successfully registered, a prompt will show up, and the details would be appended to the mgrAccount.txt file.

Below is the scenario of the prompt message:

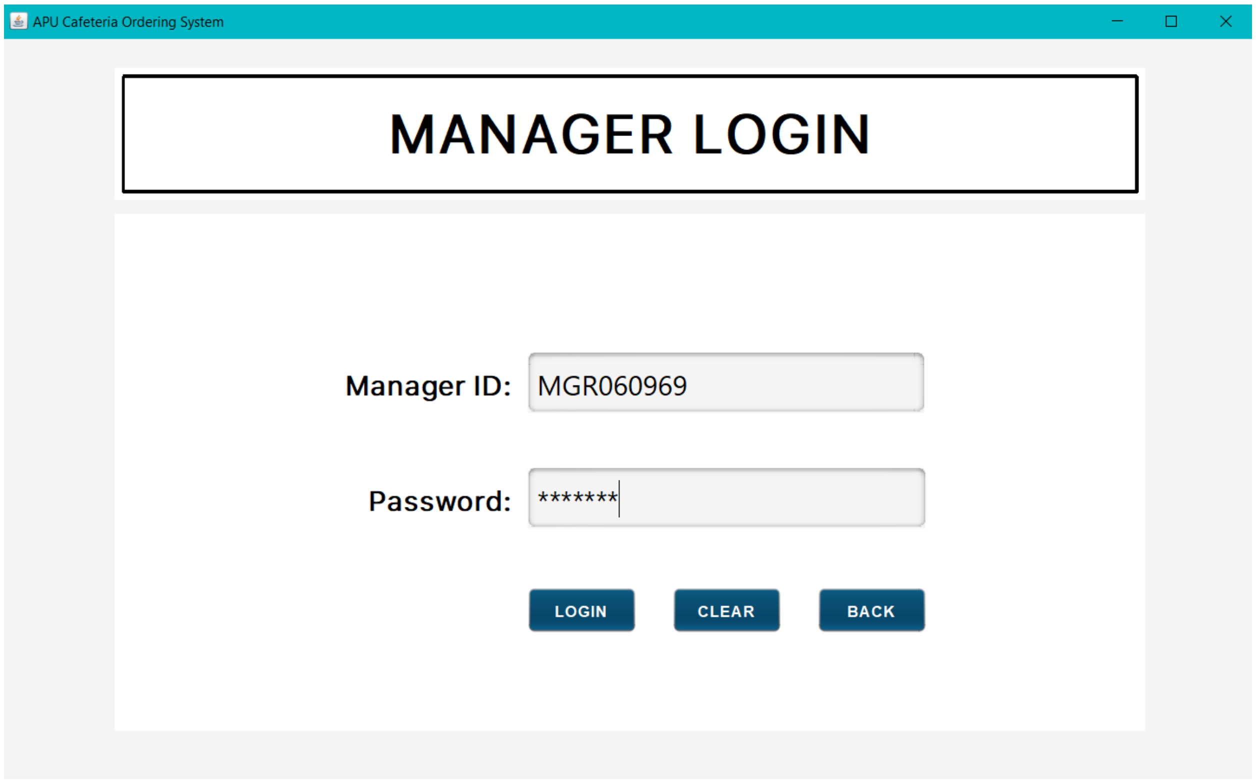
Prompt Messages	Trigger Conditions
 Message Registration successful!	When all the conditions are met accordingly after the "Register" button is clicked.

Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	Trigger Conditions
 Message Error: Please fill out all text fields.	When any of the registration text fields are empty upon pressing the "Register" button.
 Message Error: Please enter a valid Manager ID.	When the Manager ID does not match the correct format after clicking the "Register" button. (E.g. TP012345)

 Message Error: Please enter a password between 6 and 20 characters in length. OK	The password entered is not in between 6 and 20 characters in length after the “Register” button is clicked.
 Message Error: Please enter a valid email address. OK	The email address entered by the user does not match the standard email pattern. (E.g. manageremail@mail.com)

4.3.2 - Manager Login



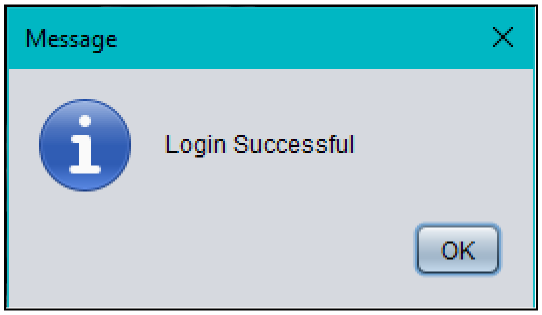
Manager ID: MGR060969

Password: *******

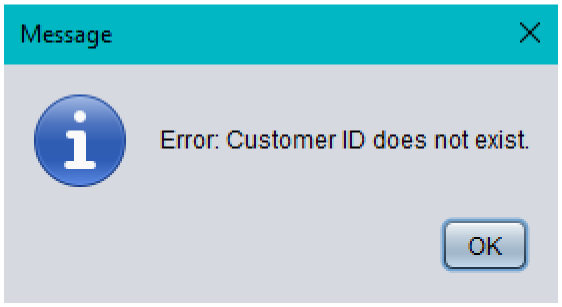
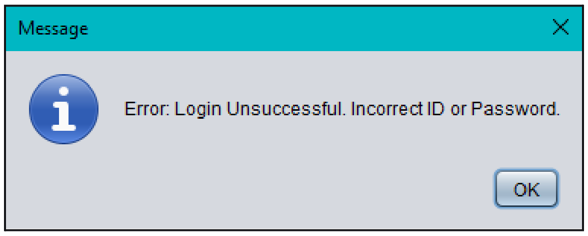
Figure 4.3.2 - Manager Login Page

This page will show when the user clicks on the “LOGIN” button on the “MANAGER” section in the Welcome Page. The user can input their registered Manager ID and their password that is stored in mgrAccount.txt in the text fields shown. The password typed will be shown in asterisks (*) to enhance security. The user can choose to login by clicking on the “LOGIN” button or clear the text fields by clicking on the “CLEAR” button. If the user wishes to return to the Welcome Page, they could click on the “BACK” button.

Below are a few scenarios of the prompt message:

Prompt Messages	**Trigger Conditions**
 Message Login Successful OK	When the Manager ID and password entered by the user matches one of the lines within the text file after the "Login" button is clicked.

Below are some of the error messages that could pop up from the errors triggered by the user (Non-exception errors):

Error Messages	**Trigger Conditions**
 Message Error: Customer ID does not exist. OK	When the Manager ID keyed in by the user does not exist within the text file.
 Message Error: Login Unsuccessful. Incorrect ID or Password. OK	Occurs when the password entered by the user does not match the Manager ID within the text file.

4.3.3 - Manager Home

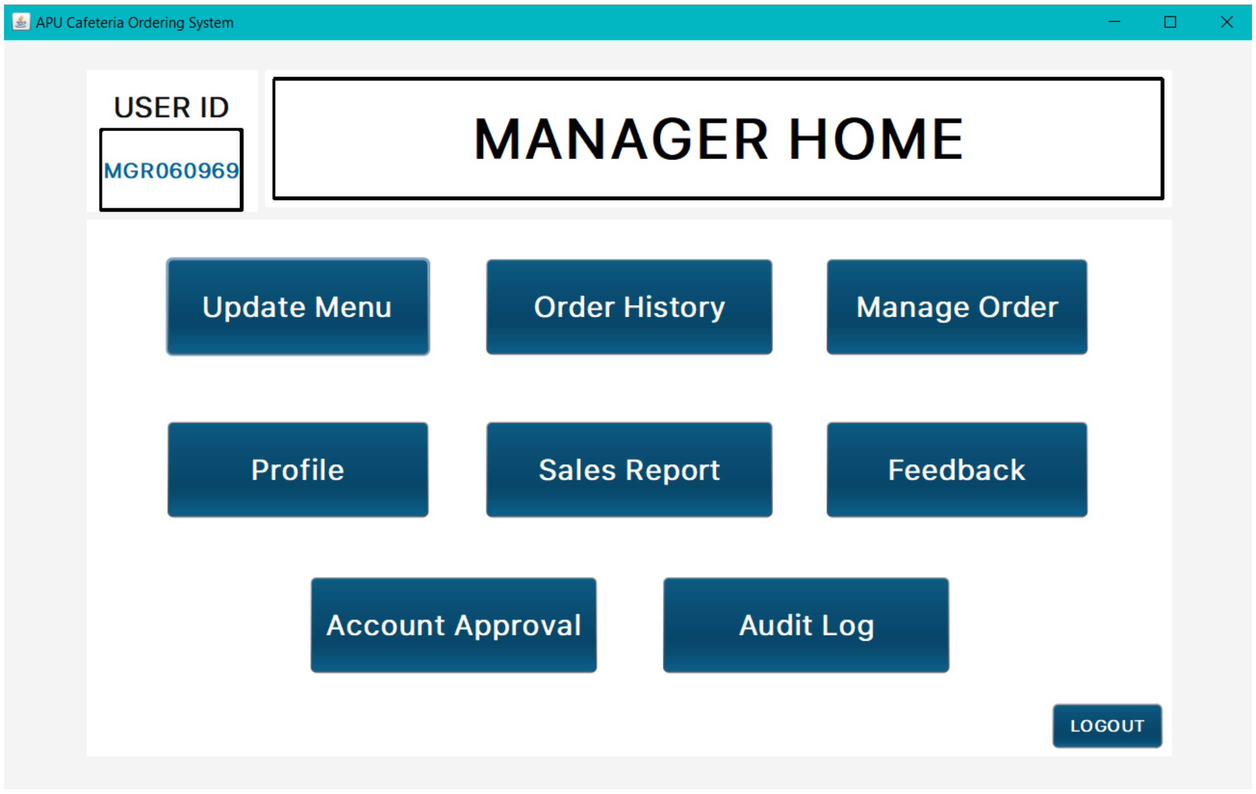
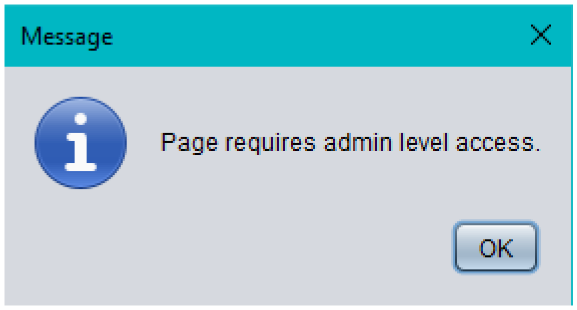


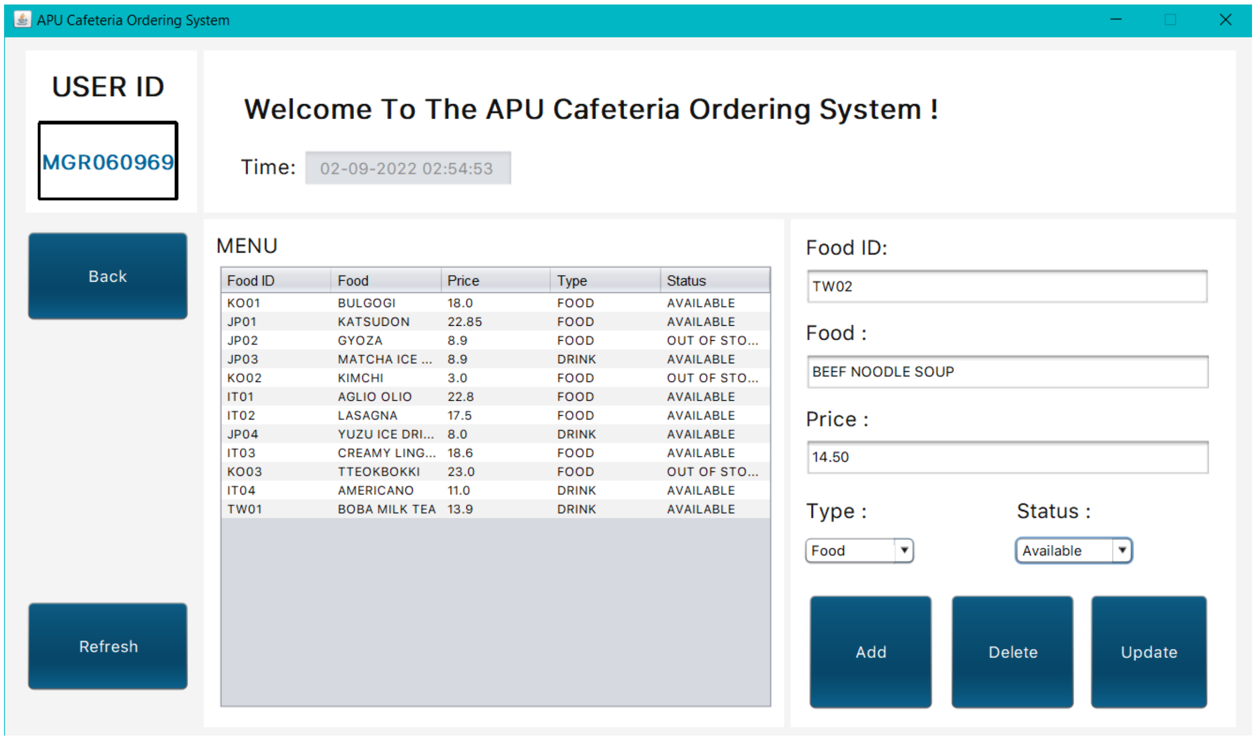
Figure 4.3.3 - Manager Home Page

This is the first page the user will see when they have successfully logged in as a manager or admin. The user can choose to update the menu by clicking on the “Update Menu” button, view the order history of all users using the “Order History” button, manage all customers’ order using the “Manage Order” button, view their profile using the “Profile” button, printing and checking the sales report using the “Sales Report” button, and view customers’ feedback using the “Feedback” button. If the user was logged in as an admin, they would be able to approve and reject managers accounts using the “Account Approval” button, as well as view the audit log using the “Audit Log” button. Each button will lead to a different page. The user can also log out to the Welcome Page by clicking on the “LOGOUT” button at the bottom right of the page.

Below are a few scenarios of the prompt message:

Prompt Messages	**Trigger Conditions**
 Message Page requires admin level access. 	When a Manager selects the “Account Approval Button” or Audit Log instead of the Admin.

4.3.4 - Manager Menu



MGR060969

Welcome To The APU Cafeteria Ordering System !

Time: 02-09-2022 02:54:53

MENU

Food ID	Food	Price	Type	Status
KO01	BULGOGI	18.0	FOOD	AVAILABLE
JP01	KATSUDON	22.85	FOOD	AVAILABLE
JP02	GYOZA	8.9	FOOD	OUT OF STO...
JP03	MATCHA ICE ...	8.9	DRINK	AVAILABLE
KO02	KIMCHI	3.0	FOOD	OUT OF STO...
IT01	AGLIO OLIO	22.8	FOOD	AVAILABLE
IT02	LASAGNA	17.5	FOOD	AVAILABLE
JP04	YUZU ICE DRI...	8.0	DRINK	AVAILABLE
IT03	CREAMY LING...	18.6	FOOD	AVAILABLE
KO03	TTEOKBOKKI	23.0	FOOD	OUT OF STO...
IT04	AMERICANO	11.0	DRINK	AVAILABLE
TW01	BOBA MILK TEA	13.9	DRINK	AVAILABLE

Food ID: TW02

Food : BEEF NOODLE SOUP

Price : 14.50

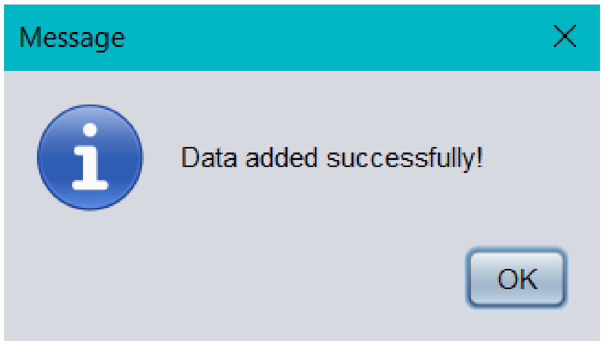
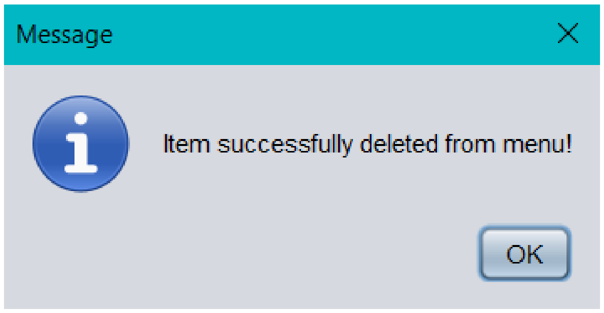
Type : Food

Status : Available

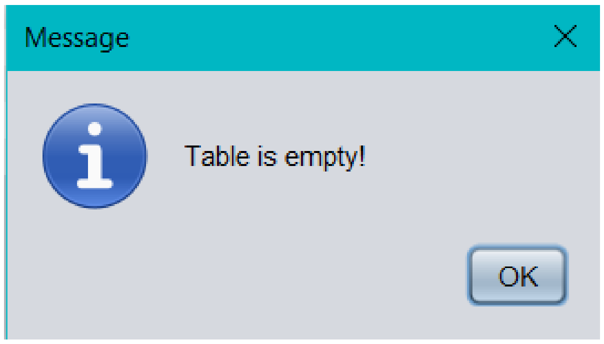
Figure 4.3.4 - Manager Menu Page

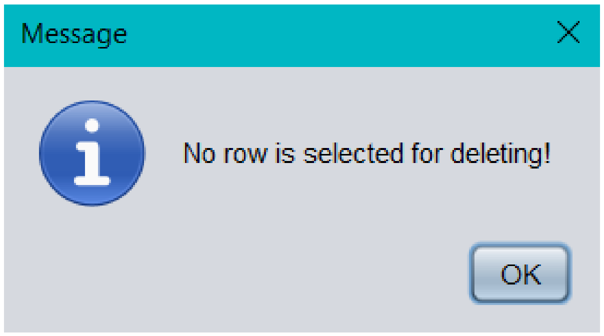
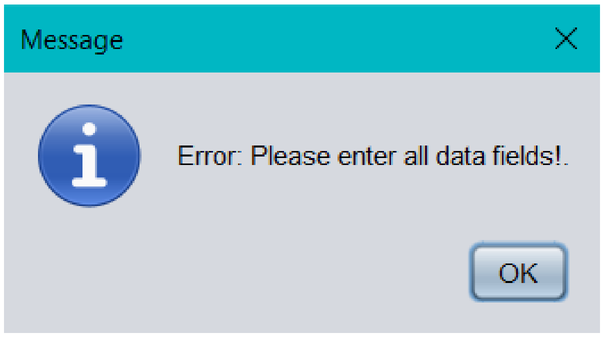
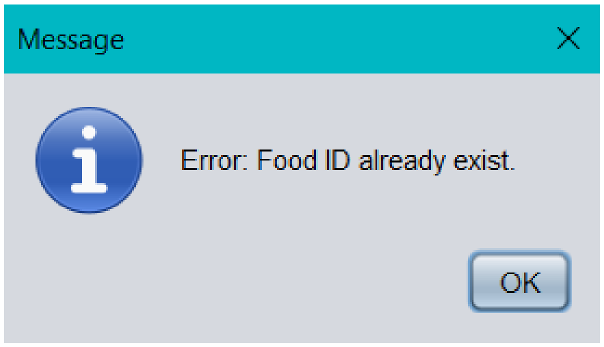
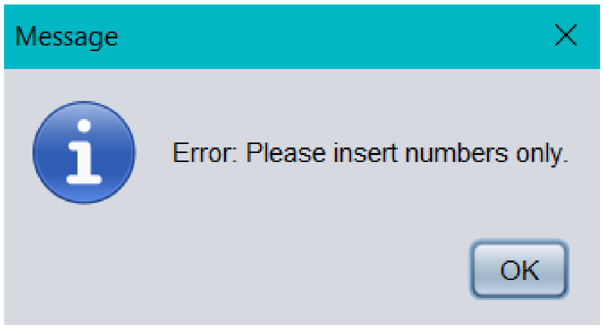
The top left of the page shows the User ID of the logged in user, while the top panel shows the time that the user has viewed the Manager Menu page. The “Menu” table will show food and drinks that were marked as both “AVAILABLE” and “OUT OF STOCK” in the menu.txt file. To add any item into the menu, manually type a new Food ID, Food, Price, Type and Status to the respective text fields. After that, the user can choose to add the item to the menu by clicking on the “Add” button. If the user would like to delete an item from the “Menu” table, they could do so by clicking on the items on the table or manually typing its details to the respective text fields and clicking on the “Delete” button. All items in the “Menu” table will be printed to the menu.txt file once the user clicks on the “Update” button. If the user did some adjustments by modifying certain details directly on the table itself, they could click on the “Refresh” button to revert the “Menu” table to its original state or click on the “Update” button to print the adjustments to the menu.txt file. If the user wishes to return to the Manager Home Page, they could click on the “Back” button.

Below are a few scenarios of the prompt message:

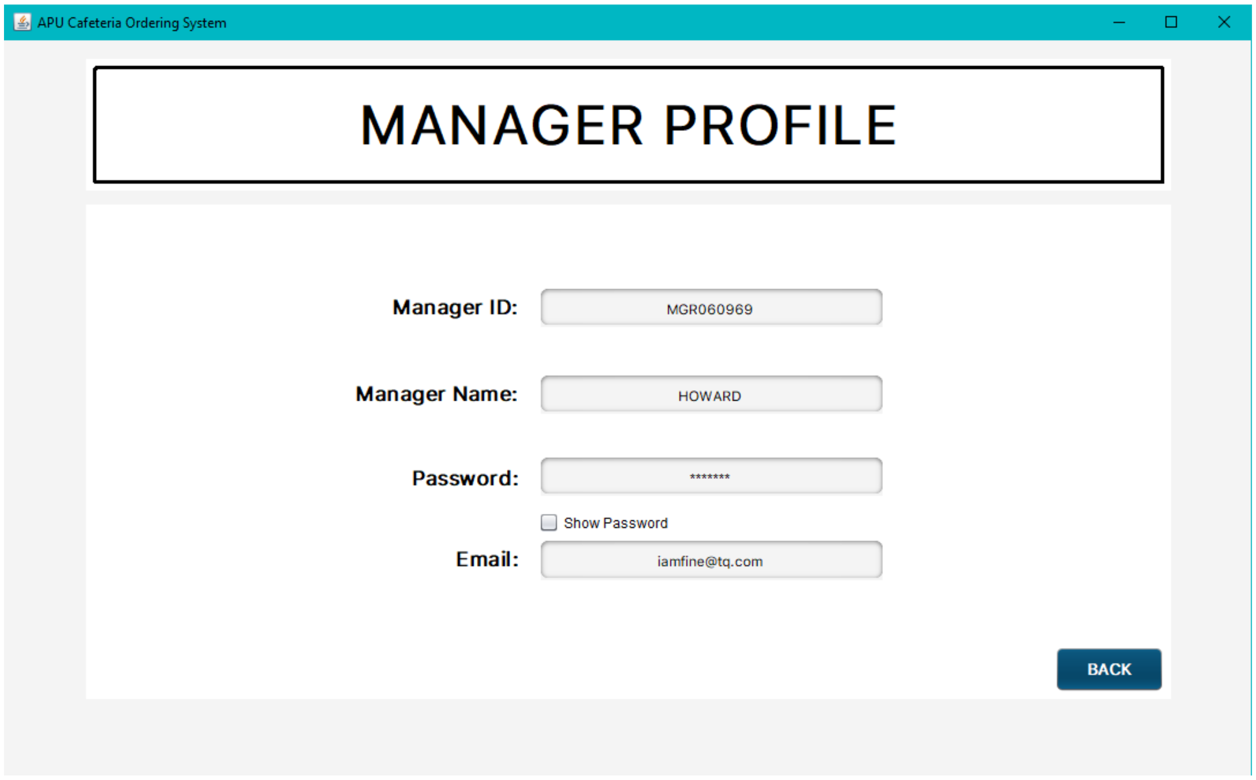
Prompt Messages	Trigger Conditions
 Message Data added successfully!	When an item is successfully added to the "Menu" table.
 Message Item successfully deleted from menu!	When an item is successfully deleted from the "Menu" table.

Below are the error messages that could pop up due to the errors triggered by the user (Non-exception errors):

Error Messages	Trigger Conditions
 Message Table is empty!	"Menu" table is empty when "Delete" is clicked.

 Message No row is selected for deleting! OK	Data fields are empty when "Delete" is clicked.
 Message Error: Please enter all data fields!. OK	Data fields are empty when "Add" is clicked.
 Message Error: Food ID already exist. OK	Food ID already exists in the menu.txt file when "Add" is clicked.
 Message Error: Please insert numbers only. OK	Users entered non-numeric characters in the "Price" text field.

4.3.5 - Manager Profile



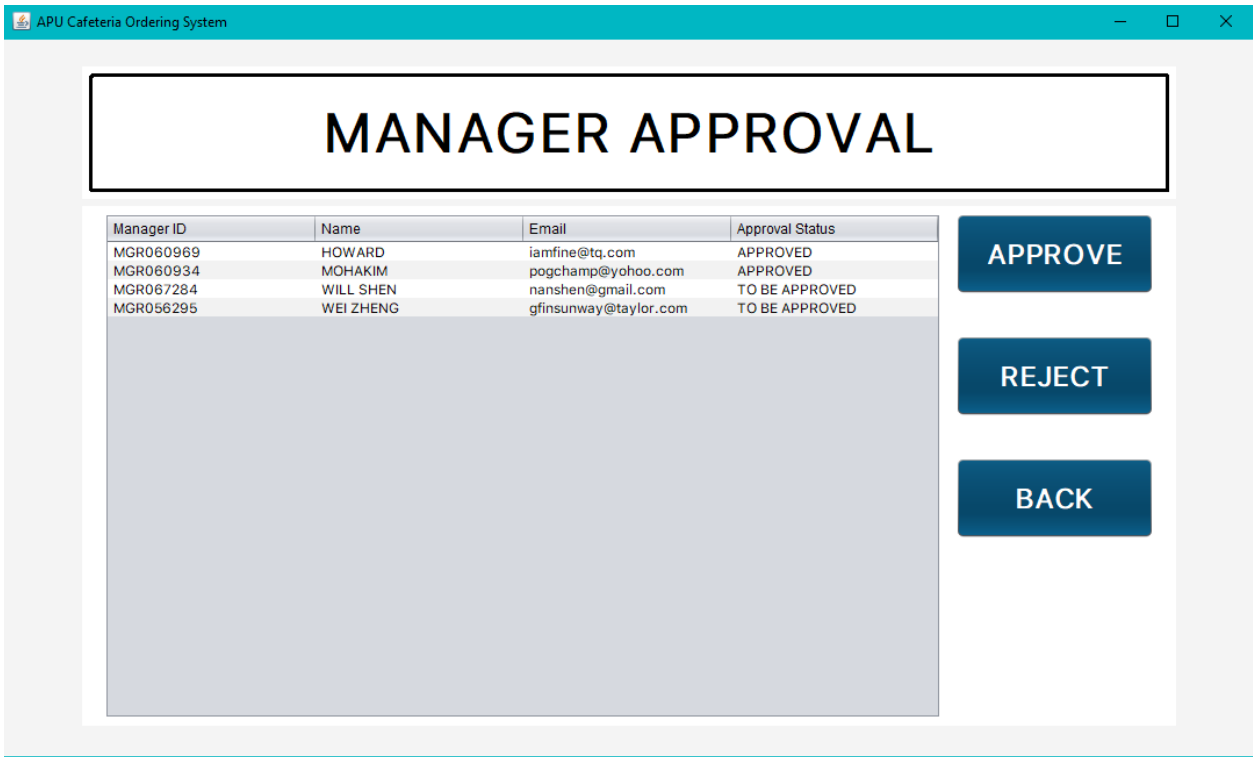
MANAGER PROFILE

Manager ID:	MGR060969
Manager Name:	HOWARD
Password:	*******
	Show Password
Email:	iamfine@tq.com

Figure 4.3.5 - Manager Profile Page

The manager profile page shows all the credentials of the manager that is currently logged in. The credentials include the Manager ID, Manager Name, Password and Email address. There is also a small checkbox called "Show Password" that can be clicked to display the password of the manager which is hidden by default. Lastly, at the bottom right corner of the page is a "BACK" button which takes the Customer back to the Customer Home page.

4.3.6 - Manager Approval

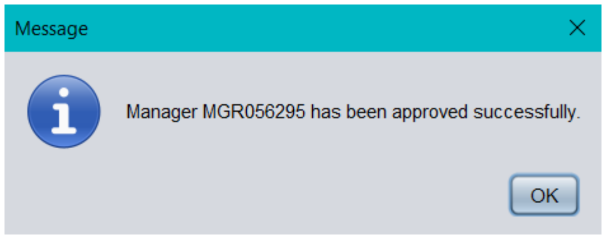
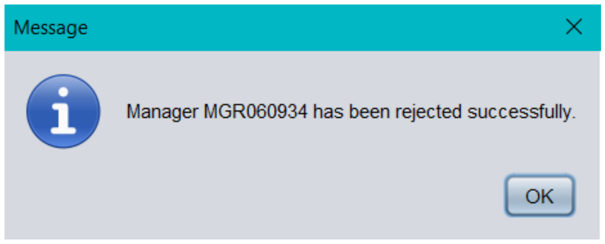


Manager ID	Name	Email	Approval Status
MGR060969	HOWARD	iamfine@tq.com	APPROVED
MGR060934	MOHAKIM	pogchamp@yohoo.com	APPROVED
MGR067284	WILL SHEN	nanshen@gmail.com	TO BE APPROVED
MGR056295	WEI ZHENG	gfinsunway@taylor.com	TO BE APPROVED

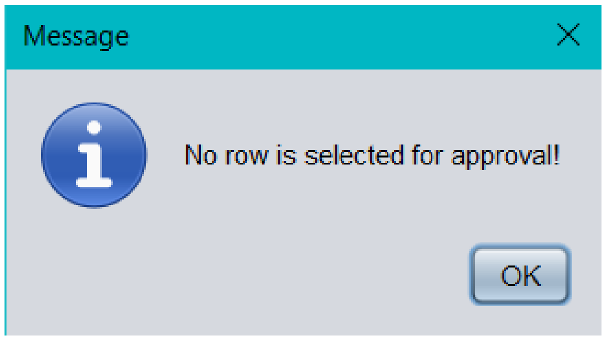
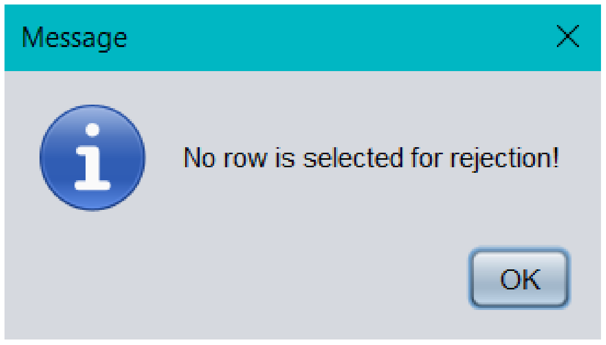
Figure 4.3.6 - Manager Approval Page

The manager approval page shows all the manager accounts in the system in a table and it is used to decide whether to approve them or reject them. By default, the manager accounts registered is set to “TO BE APPROVED” by default and the only user who can access this page and use its features is the Administrator. In order to approve a manager, a row has to be selected from the table to specify which manager to approve. Once a manager has been selected from the table, the Administrator has the choice of either clicking the “Approve” or “Reject” buttons. The “Approve” button will update the manager’s Approval Status in the table to “APPROVED”. The “Reject” button on the other hand will simply delete the row from the table and in the text file as well to permanently remove the manager should they choose not to remove them. There is also a “Back” button within the page to return to the Manager Home Page.

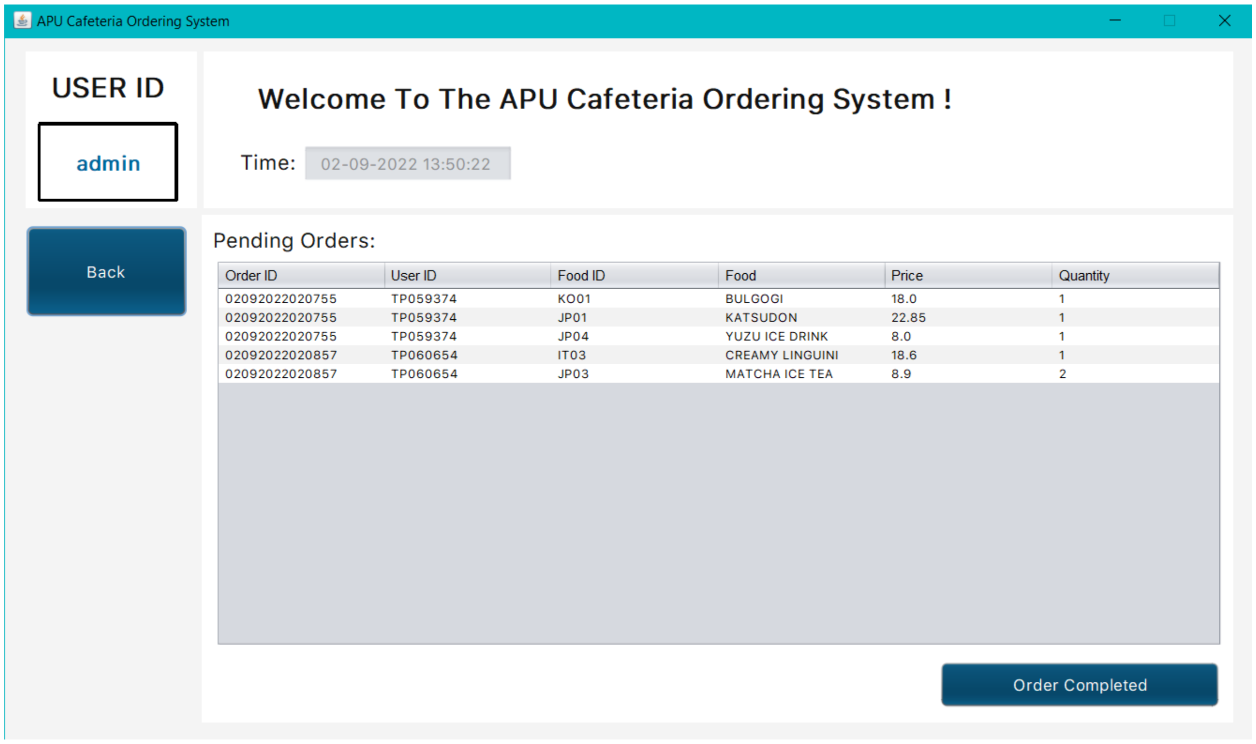
Below is the only scenario of the prompt message:

Prompt Messages	Trigger Conditions
 Message Manager MGR056295 has been approved successfully. OK	When the Admin clicks on the “Approve” button after selecting a row from the table.
 Message Manager MGR060934 has been rejected successfully. OK	When the Admin clicks on the “Reject” button after selecting a row from the table.

Below are the error messages that could pop up caused by errors triggered by the user. (Non-exception errors):

Error Messages	Trigger Conditions
 Message No row is selected for approval! OK	A row in the table is not selected before clicking the “Approve” button.
 Message No row is selected for rejection! OK	A row in the table is not selected before clicking the “Reject” button.

4.3.7 - Manager Manage Order


APU Cafeteria Ordering System
USER ID
admin
Back
Welcome To The APU Cafeteria Ordering System !
Order Completed

Time: 02-09-2022 13:50:22

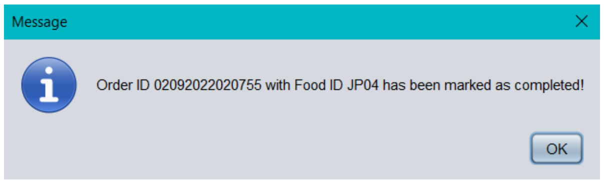
Pending Orders:

Order ID	User ID	Food ID	Food	Price	Quantity
02092022020755	TP059374	KO01	BULGOGI	18.0	1
02092022020755	TP059374	JP01	KATSUDON	22.85	1
02092022020755	TP059374	JP04	YUZU ICE DRINK	8.0	1
02092022020857	TP060654	IT03	CREAMY LINGUINI	18.6	1
02092022020857	TP060654	JP03	MATCHA ICE TEA	8.9	2

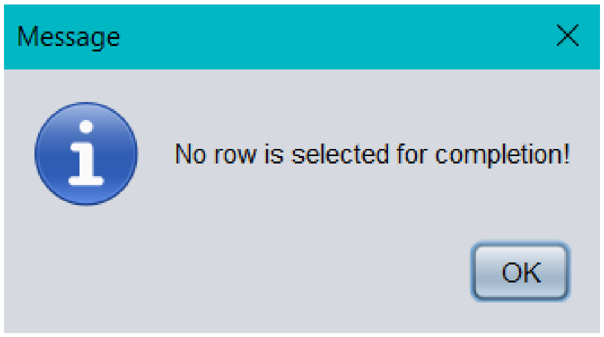
Figure 4.3.7 - Manager Manage Order Page

The top left of the page shows the User ID of the logged in user, while the top panel shows the time that the user has viewed the Manager Manage Order page. The "Pending Order" table will show all the pending orders in the pendingOrders.txt file that were made by the users in the system. Once the user selects a row in the table and clicks on the "Order Completed" button, that particular order will be sent to the completedOrders.txt file, allowing the user to leave a review in the Customer Order History page. There is also a "Back" button within the page to return to the Manager Home Page.

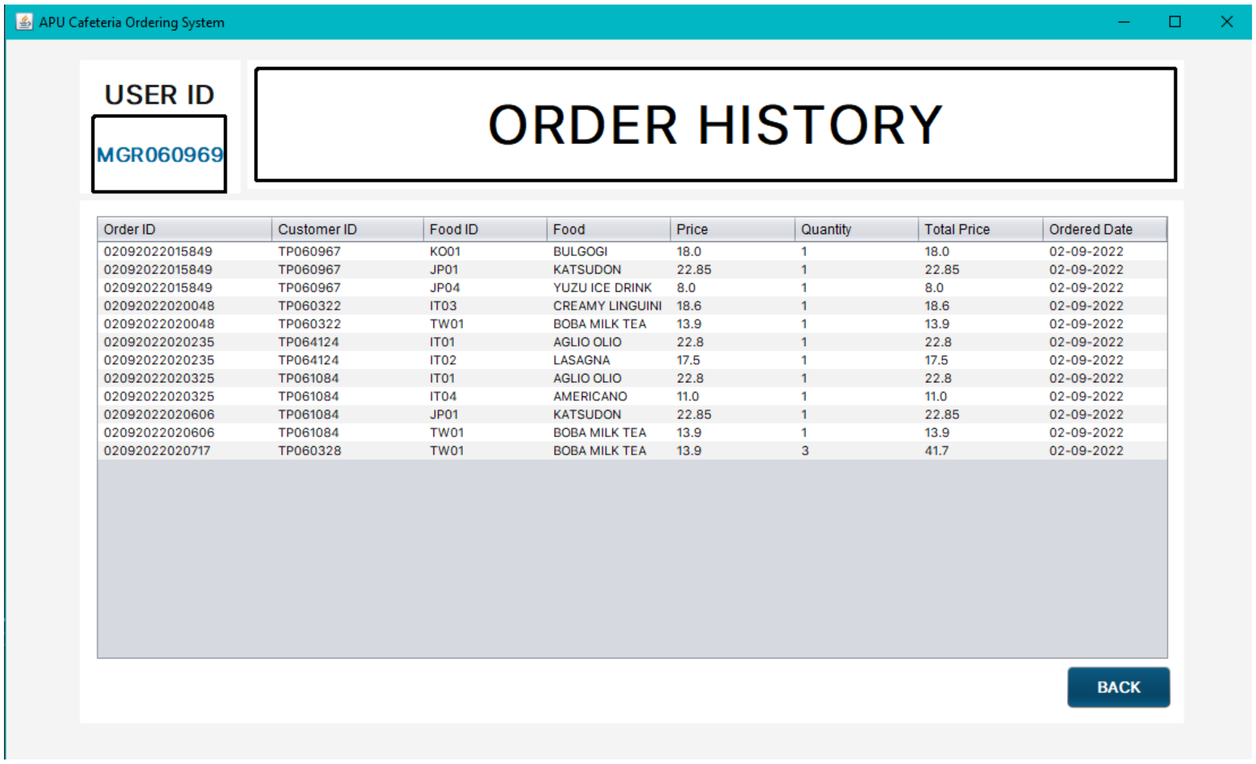
Below is the only scenario of the prompt message:

Prompt Message	**Trigger Conditions**
 Order ID 02092022020755 with Food ID JP04 has been marked as completed!	When the manager clicks on the "Order Completed" button after selecting a row from the table.

Below are the error messages that could pop up caused by errors triggered by the user. (Non-exception errors):

Error Messages	Trigger Conditions
 Message No row is selected for completion!	A row in the table is not selected before clicking the “Order Completed” button.

4.3.8 - Manager Order History

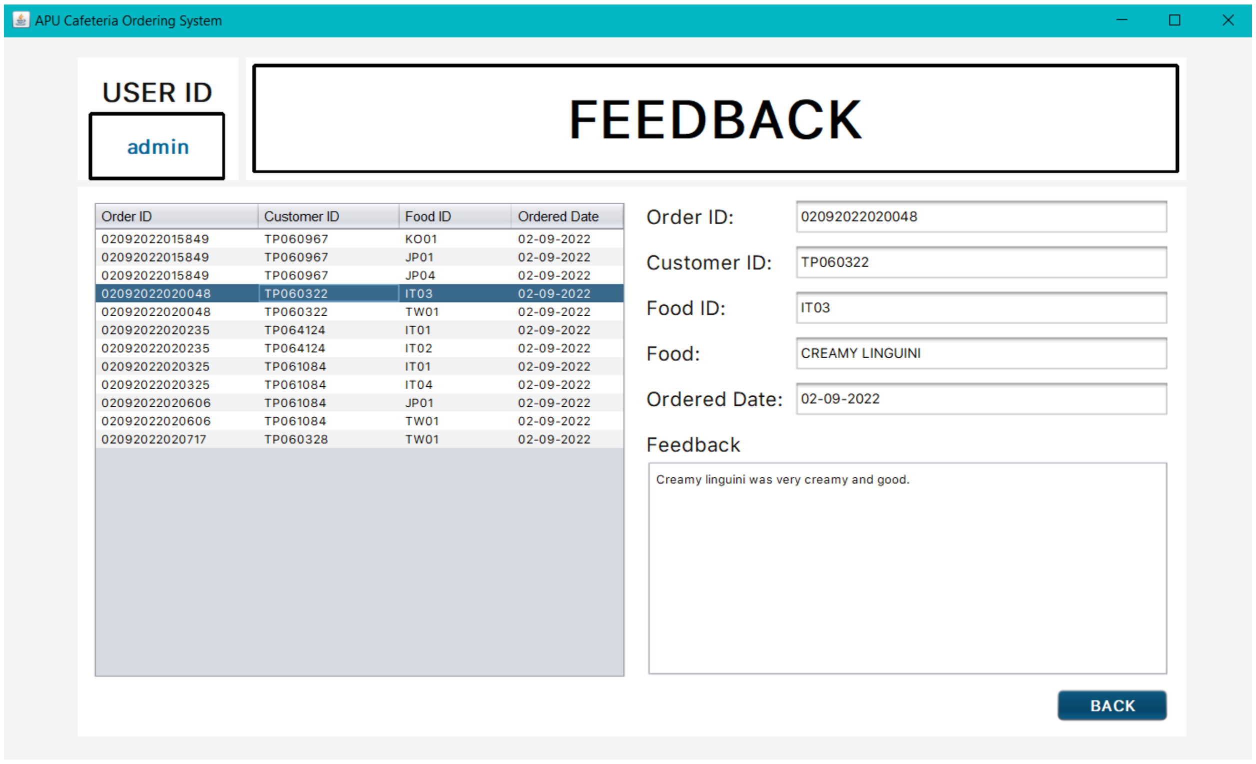


Order ID	Customer ID	Food ID	Food	Price	Quantity	Total Price	Ordered Date
02092022015849	TP060967	KO01	BULGOGI	18.0	1	18.0	02-09-2022
02092022015849	TP060967	JP01	KATSUDON	22.85	1	22.85	02-09-2022
02092022015849	TP060967	JP04	YUZU ICE DRINK	8.0	1	8.0	02-09-2022
02092022020048	TP060322	IT03	CREAMY LINGUINI	18.6	1	18.6	02-09-2022
02092022020048	TP060322	TW01	BOBA MILK TEA	13.9	1	13.9	02-09-2022
02092022020235	TP064124	IT01	AGLIO OLIO	22.8	1	22.8	02-09-2022
02092022020235	TP064124	IT02	LASAGNA	17.5	1	17.5	02-09-2022
02092022020325	TP061084	IT01	AGLIO OLIO	22.8	1	22.8	02-09-2022
02092022020325	TP061084	IT04	AMERICANO	11.0	1	11.0	02-09-2022
02092022020606	TP061084	JP01	KATSUDON	22.85	1	22.85	02-09-2022
02092022020606	TP061084	TW01	BOBA MILK TEA	13.9	1	13.9	02-09-2022
02092022020717	TP060328	TW01	BOBA MILK TEA	13.9	3	41.7	02-09-2022

Figure 4.3.8 - Manager Order History Page

The top left of the page displays the ID of the Manager that is currently logged in. The Manager Order History section consists of a table with an Order History table that displays all the past orders of every user line by line. The columns within the table consist of the Order ID, Food ID, Food, Price, Quantity, Total Price and Order Date. Aside from that, there is also the “Back” button at the bottom right corner of the page that will redirect the user back to the Manager Home page when it is clicked.

4.3.9 - Manager Feedback



Order ID	Customer ID	Food ID	Ordered Date
02092022015849	TP060967	KO01	02-09-2022
02092022015849	TP060967	JP01	02-09-2022
02092022015849	TP060967	JP04	02-09-2022
02092022020048	TP060322	IT03	02-09-2022
02092022020048	TP060322	TW01	02-09-2022
02092022020235	TP064124	IT01	02-09-2022
02092022020235	TP064124	IT02	02-09-2022
02092022020325	TP061084	IT01	02-09-2022
02092022020325	TP061084	IT04	02-09-2022
02092022020606	TP061084	JP01	02-09-2022
02092022020606	TP061084	TW01	02-09-2022
02092022020717	TP060328	TW01	02-09-2022

Order ID:	02092022020048
Customer ID:	TP060322
Food ID:	IT03
Food:	CREAMY LINGUINI
Ordered Date:	02-09-2022

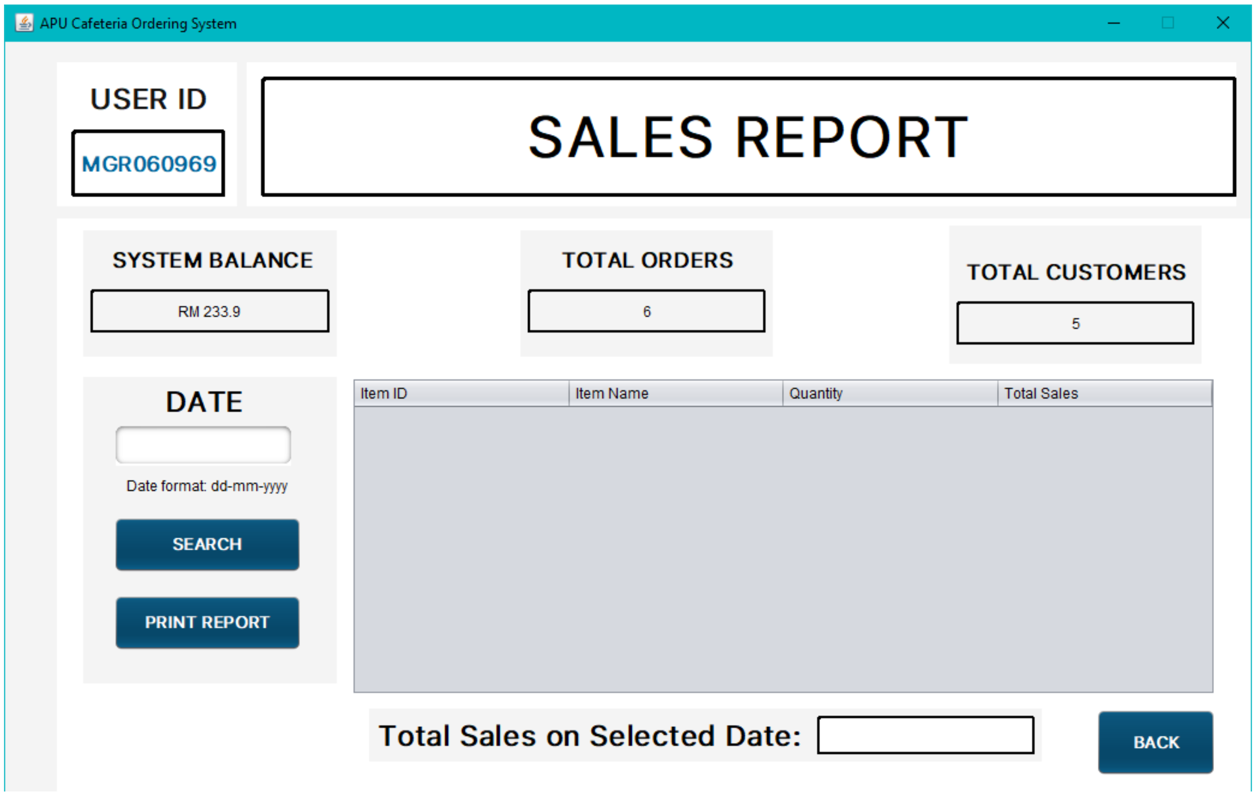
Feedback

Creamy linguini was very creamy and good.

Figure 4.3.9 - Manager Feedback Page

The top left of the page shows the User ID of the logged in user. The table at the left will show all the completed orders that contained feedback that were made by the users in the system. Once the user selects a row in the table, the feedback for that particular order will be shown on the respective text fields at the right, allowing the manager to view the feedback and order details. There is also a "Back" button within the page to return to the Manager Home Page.

4.3.10 - Manager Sales Report


APU Cafeteria Ordering System

USER ID: MGR060969

SALES REPORT

SYSTEM BALANCE: RM 233.9

TOTAL ORDERS: 6

TOTAL CUSTOMERS: 5

DATE

Date format: dd-mm-yyyy

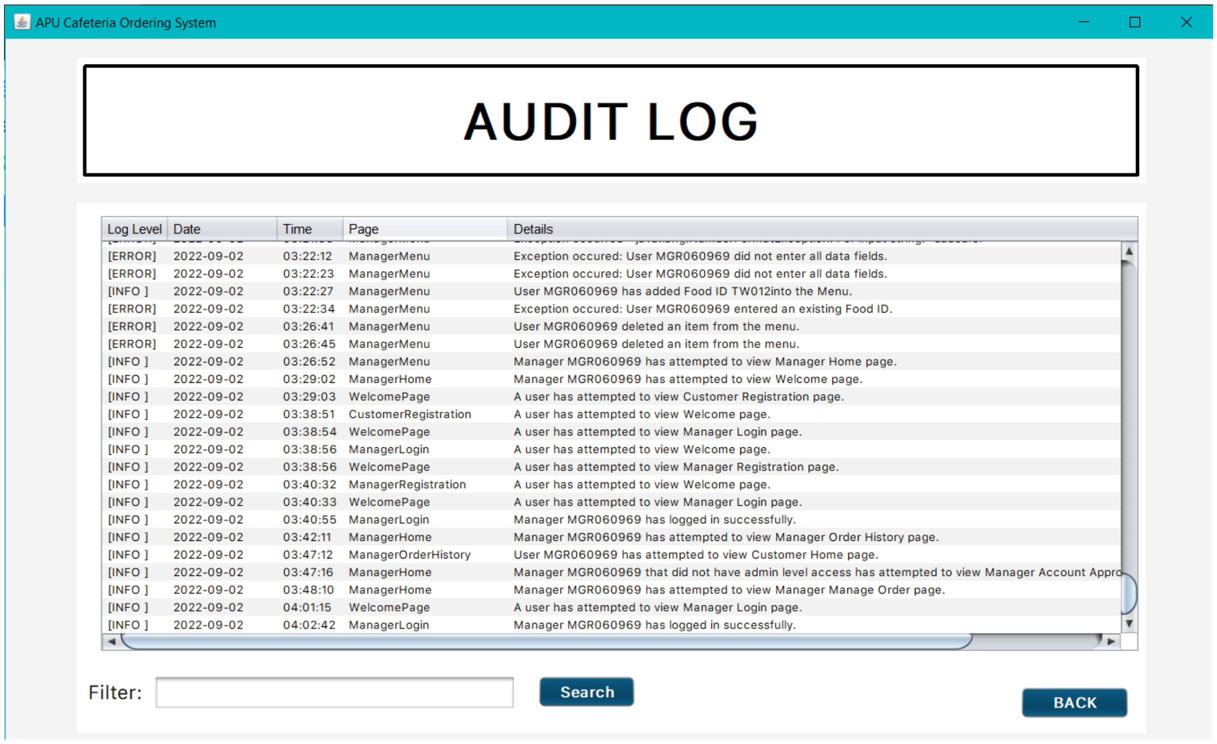
Item ID	Item Name	Quantity	Total Sales

Total Sales on Selected Date:

Figure 4.3.10 - Manager Sales Report Page

The manager sales report page is used to view the overall daily sales for a specific date selected. Within the page, there is the User ID of the logged in user at the top left corner. Below there are three boxes which show the System Balance, Total Number of Orders and the Total Number of Customers unique customers who have ordered from the system. Next, there is also a search section to specify the date to view the daily sales. For this, the user will first need to key in a date they would like to search for according to the example provided below the text field and clicking the "Search" button. By doing so, they will be able to view all the sales recorded in the system based on the sales for each food/beverage item sold. Aside from that, the total amount of money made in sales for the selected date is displayed below the table. If the user would like to print a PDF file of the report, they can do so by clicking the "Print Report" button. Finally, if they choose to return back to the Manager Home page, they can simply do so by clicking the "Back" button at the bottom right corner.

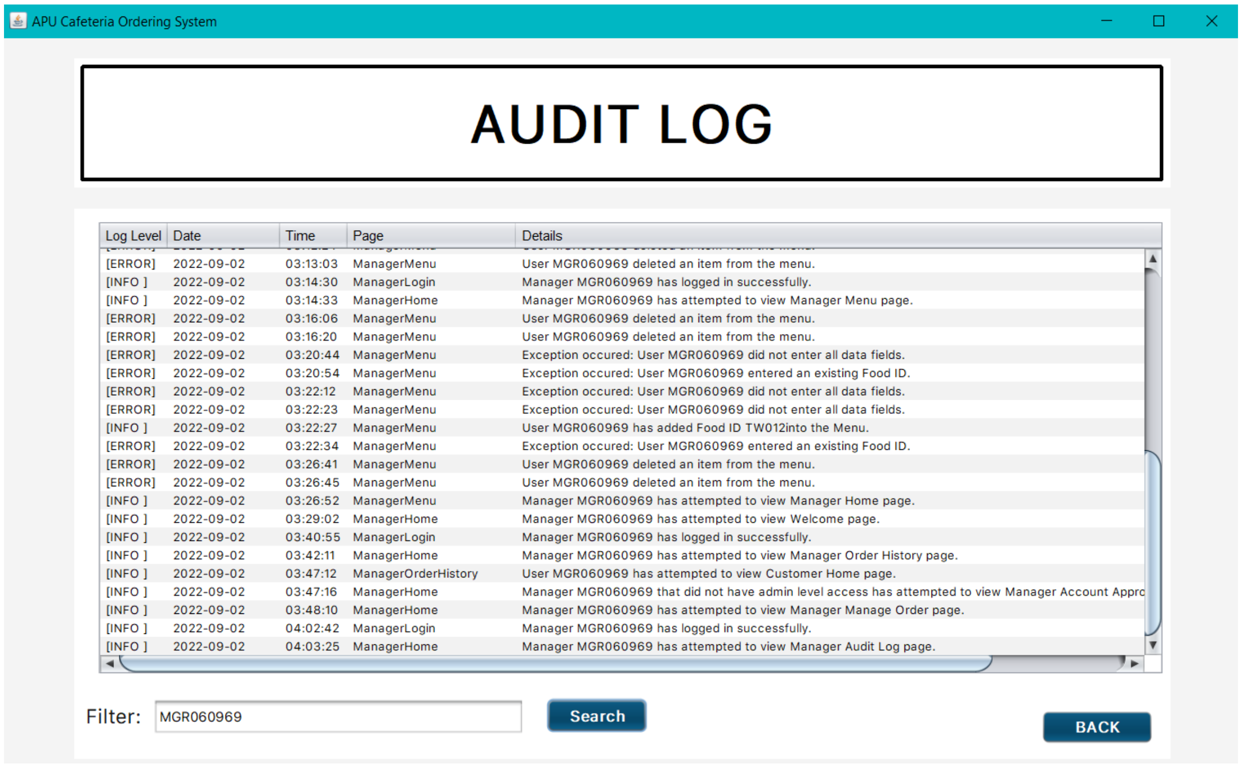
4.3.11 - Manager Audit Log



Log Level	Date	Time	Page	Details
[ERROR]	2022-09-02	03:22:12	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[ERROR]	2022-09-02	03:22:23	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[INFO]	2022-09-02	03:22:27	ManagerMenu	User MGR060969 has added Food ID TW012into the Menu.
[ERROR]	2022-09-02	03:22:34	ManagerMenu	Exception occured: User MGR060969 entered an existing Food ID.
[ERROR]	2022-09-02	03:26:41	ManagerMenu	User MGR060969 deleted an item from the menu.
[ERROR]	2022-09-02	03:26:45	ManagerMenu	User MGR060969 deleted an item from the menu.
[INFO]	2022-09-02	03:26:52	ManagerMenu	Manager MGR060969 has attempted to view Manager Home page.
[INFO]	2022-09-02	03:29:02	ManagerHome	Manager MGR060969 has attempted to view Welcome page.
[INFO]	2022-09-02	03:29:03	WelcomePage	A user has attempted to view Customer Registration page.
[INFO]	2022-09-02	03:38:51	CustomerRegistration	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:38:54	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	03:38:56	ManagerLogin	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:38:56	WelcomePage	A user has attempted to view Manager Registration page.
[INFO]	2022-09-02	03:40:32	ManagerRegistration	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:40:33	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	03:40:55	ManagerLogin	Manager MGR060969 has logged in successfully.
[INFO]	2022-09-02	03:42:11	ManagerHome	Manager MGR060969 has attempted to view Manager Order History page.
[INFO]	2022-09-02	03:47:12	ManagerOrderHistory	User MGR060969 has attempted to view Customer Home page.
[INFO]	2022-09-02	03:47:16	ManagerHome	Manager MGR060969 that did not have admin level access has attempted to view Manager Account Appro
[INFO]	2022-09-02	03:48:10	ManagerHome	Manager MGR060969 has attempted to view Manager Manage Order page.
[INFO]	2022-09-02	04:01:15	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	04:02:42	ManagerLogin	Manager MGR060969 has logged in successfully.

Figure 4.3.11.1 - Manager Audit Log Page

Only the admin role is allowed to access this page. This page shows the audit logs of every user in the system, which includes the severity level, date, time, page, and details. The user can filter any queries using by clicking on the “Search” button below.



Log Level	Date	Time	Page	Details
[ERROR]	2022-09-02	03:13:03	ManagerMenu	User MGR060969 deleted an item from the menu.
[INFO]	2022-09-02	03:14:30	ManagerLogin	Manager MGR060969 has logged in successfully.
[INFO]	2022-09-02	03:14:33	ManagerHome	Manager MGR060969 has attempted to view Manager Menu page.
[ERROR]	2022-09-02	03:16:06	ManagerMenu	User MGR060969 deleted an item from the menu.
[ERROR]	2022-09-02	03:16:20	ManagerMenu	User MGR060969 deleted an item from the menu.
[ERROR]	2022-09-02	03:20:44	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[ERROR]	2022-09-02	03:20:54	ManagerMenu	Exception occured: User MGR060969 entered an existing Food ID.
[ERROR]	2022-09-02	03:22:12	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[ERROR]	2022-09-02	03:22:23	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[INFO]	2022-09-02	03:22:27	ManagerMenu	User MGR060969 has added Food ID TW012into the Menu.
[ERROR]	2022-09-02	03:22:34	ManagerMenu	Exception occured: User MGR060969 entered an existing Food ID.
[ERROR]	2022-09-02	03:26:41	ManagerMenu	User MGR060969 deleted an item from the menu.
[ERROR]	2022-09-02	03:26:45	ManagerMenu	User MGR060969 deleted an item from the menu.
[INFO]	2022-09-02	03:26:52	ManagerMenu	Manager MGR060969 has attempted to view Manager Home page.
[INFO]	2022-09-02	03:29:02	ManagerHome	Manager MGR060969 has attempted to view Welcome page.
[INFO]	2022-09-02	03:40:55	ManagerLogin	Manager MGR060969 has logged in successfully.
[INFO]	2022-09-02	03:42:11	ManagerHome	Manager MGR060969 has attempted to view Manager Order History page.
[INFO]	2022-09-02	03:47:12	ManagerOrderHistory	User MGR060969 has attempted to view Customer Home page.
[INFO]	2022-09-02	03:47:16	ManagerHome	Manager MGR060969 that did not have admin level access has attempted to view Manager Account Appro
[INFO]	2022-09-02	03:48:10	ManagerHome	Manager MGR060969 has attempted to view Manager Manage Order page.
[INFO]	2022-09-02	04:02:42	ManagerLogin	Manager MGR060969 has logged in successfully.
[INFO]	2022-09-02	04:03:25	ManagerHome	Manager MGR060969 has attempted to view Manager Audit Log page.

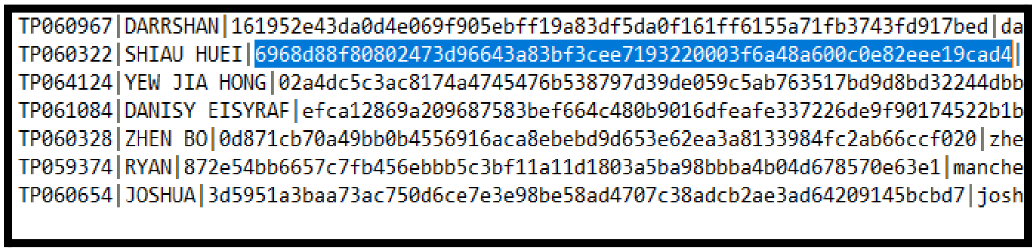
Filter: MGR060969

Figure 4.3.11.2 - Manager Audit Log Page with the Filter Query "MGR060969"

5.0 Extra Features

5.1 Hashing

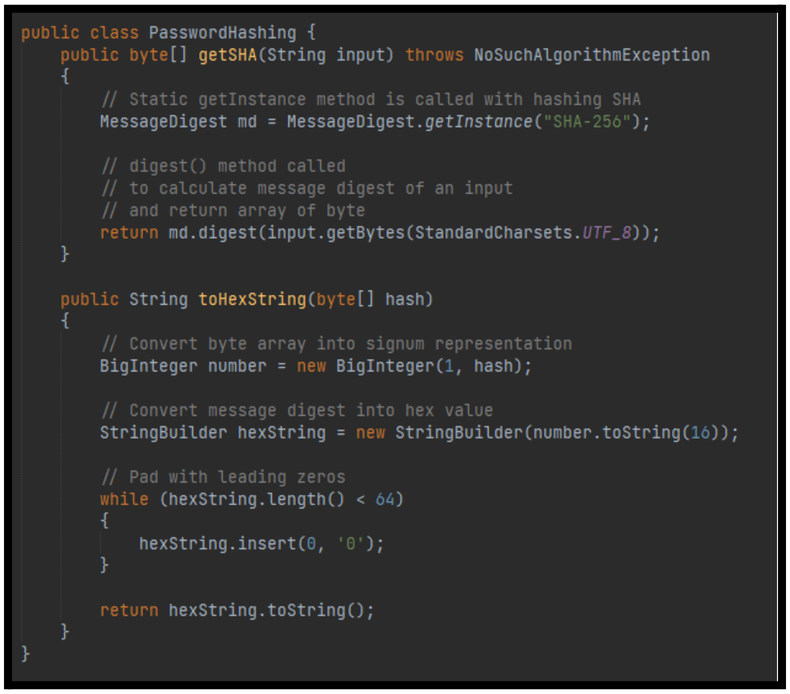
Hashing is used in the text file for the purpose of security. When registering for an account, the passwords of the users that will be stored in the text files are converted into a hash format of SHA-256 before being appended. By doing so, no one will be able to obtain the passwords of the users from the text file directly as seen in the image below.



```
TP060967|DARRSHAN|161952e43da0d4e069f905ebff19a83df5da0f161ff6155a71fb3743fd917bed|da
TP060322|SHIAU HUEI|6968d88f80802473d96643a83bf3cee7193220003f6a48a600c0e82eee19cad4|
TP064124|YEW JIA HONG|02a4dc5c3ac8174a4745476b538797d39de059c5ab763517bd9d8bd32244dbb
TP061084|DANISY EISYRAF|efca12869a209687583bef664c480b9016dfeafe337226de9f90174522b1b
TP060328|ZHEN BO|0d871cb70a49bb0b4556916aca8ebebd9d653e62ea3a8133984fc2ab66ccf020|zhe
TP059374|RYAN|872e54bb6657c7fb456ebbb5c3bf11a11d1803a5ba98bbba4b04d678570e63e1|manche
TP060654|JOSHUA|3d5951a3baa73ac750d6ce7e3e98be58ad4707c38adcb2ae3ad64209145bcbd7|josh
```

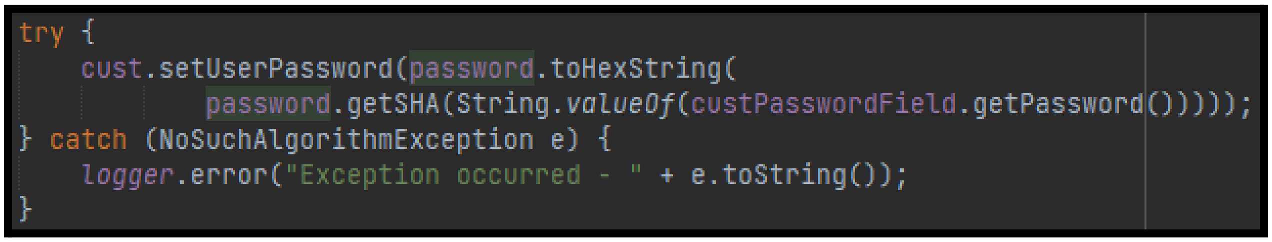
Figure 5.1.1 - Stored Passwords in Customer Account text file

As the passwords are hashed, the program will also need to hash the password entered by the user in the login pages to compare the hash of the user password and the one within the text file for validation. Below is a class called PasswordHashing which is dedicated to hashing a string input.



```java
public class PasswordHashing {
    public byte[] getSHA(String input) throws NoSuchAlgorithmException
    {
        // Static getInstance method is called with hashing SHA
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        // digest() method called
        // to calculate message digest of an input
        // and return array of byte
        return md.digest(input.getBytes(StandardCharsets.UTF_8));
    }

    public String toHexString(byte[] hash)
    {
        // Convert byte array into signum representation
        BigInteger number = new BigInteger(1, hash);

        // Convert message digest into hex value
        StringBuilder hexString = new StringBuilder(number.toString(16));

        // Pad with leading zeros
        while (hexString.length() < 64)
        {
            hexString.insert(0, '0');
        }

        return hexString.toString();
    }
}
```

Figure 5.1.2 - Class used for hashing passwords

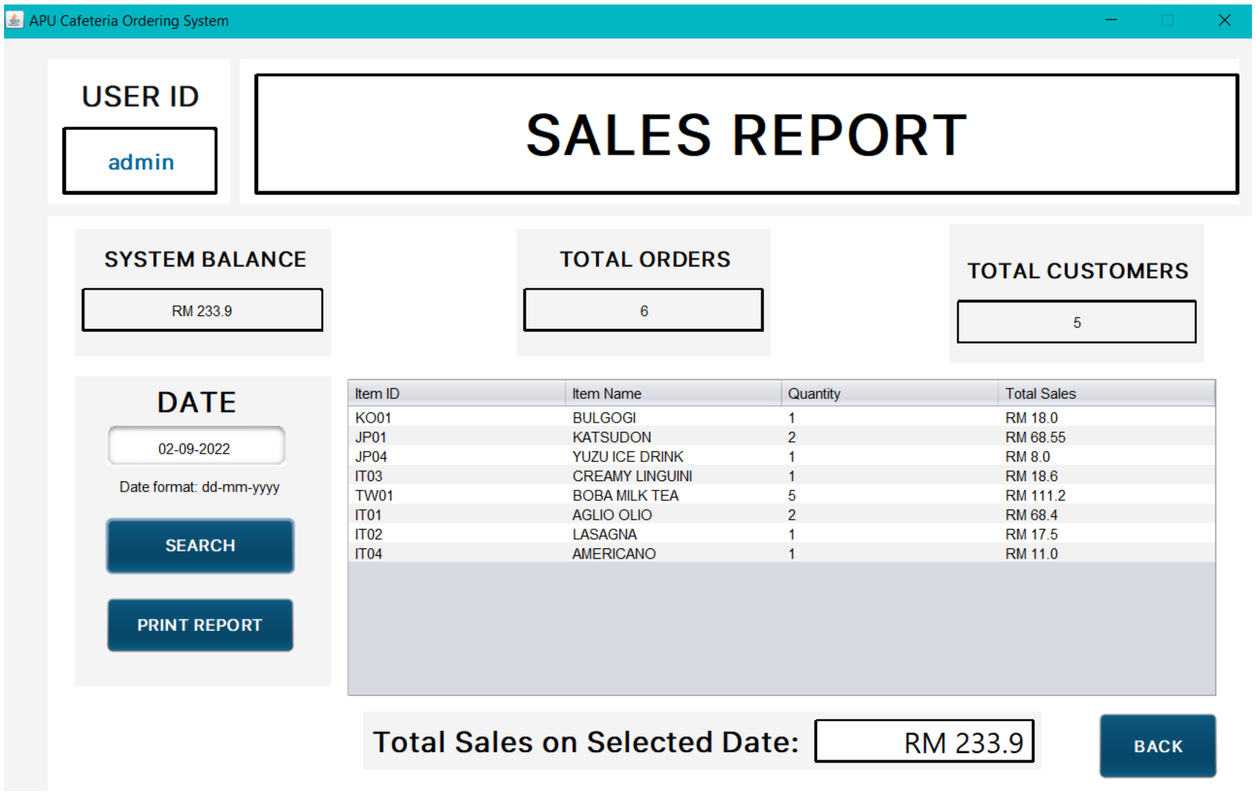


```
try {
    cust.setUserPassword(password.toHexString(
            password.getSHA(String.valueOf(custPasswordField.getPassword()))));
} catch (NoSuchAlgorithmException e) {
    logger.error("Exception occurred - " + e.toString());
}
```

Figure 5.1.3 - Converting plain text password into hash

5.2 Report Generation in PDF

The managers are able to print the sales report generated in the system by clicking on the "PRINT REPORT" button in the Manager Sales Report page.


APU Cafeteria Ordering System

USER ID: admin

SALES REPORT

SYSTEM BALANCE: RM 233.9

TOTAL ORDERS: 6

TOTAL CUSTOMERS: 5

DATE: 02-09-2022

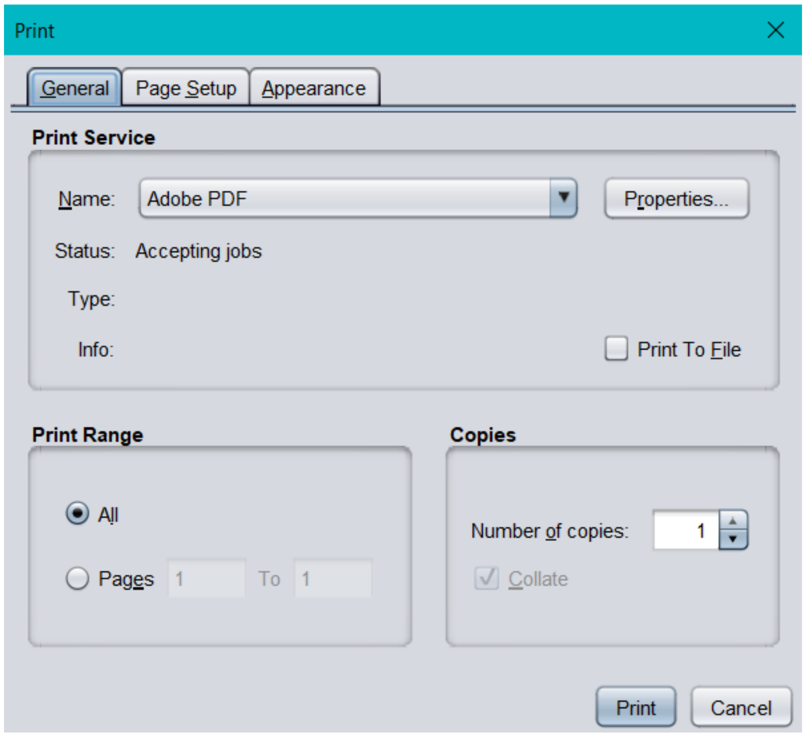
Date format: dd-mm-yyyy

Item ID	Item Name	Quantity	Total Sales
KO01	BULGOGI	1	RM 18.0
JP01	KATSUDON	2	RM 68.55
JP04	YUZU ICE DRINK	1	RM 8.0
IT03	CREAMY LINGUINI	1	RM 18.6
TW01	BOBA MILK TEA	5	RM 111.2
IT01	AGLIO OLIO	2	RM 68.4
IT02	LASAGNA	1	RM 17.5
IT04	AMERICANO	1	RM 11.0

Total Sales on Selected Date: RM 233.9

Figure 5.2.1 - Manager Sales Report Page on 02-09-2022

Once the "PRINT REPORT" button is clicked, the user can select which format should the file be outputted in the "Print" window. But for this example, the "Adobe PDF" text is selected in order to print the sales report in .pdf format.


Print

General | Page Setup | Appearance

Print Service

Name: Adobe PDF

Status: Accepting jobs

Type:

Info:

Print To File

Print Range

All

Pages 1 To 1

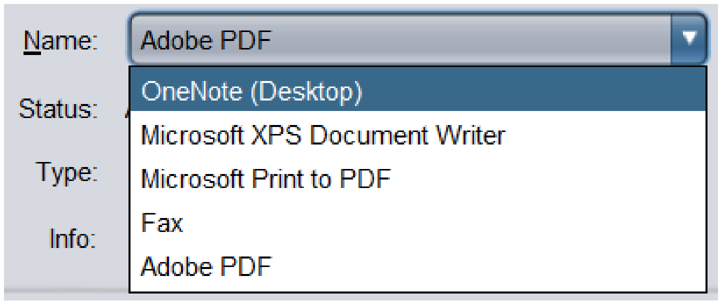
Copies

Number of copies: 1

Collate

Print | Cancel

Figure 5.2.2 - Print window


Name: Adobe PDF

Status:

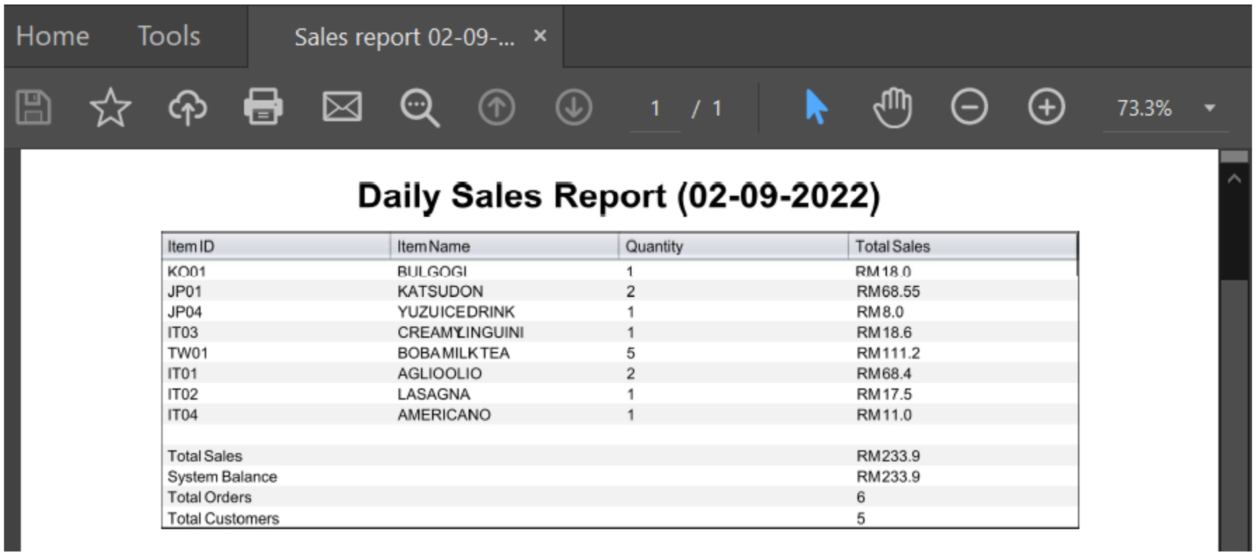
Type:

Info:

OneNote (Desktop)
Microsoft XPS Document Writer
Microsoft Print to PDF
Fax
Adobe PDF

Figure 5.2.3 - List of available formats

The output of the .pdf file will contain all of the sales made in the date chosen and it will contain the total sales, current system balance, total orders and total customers. It will look like the figure below:



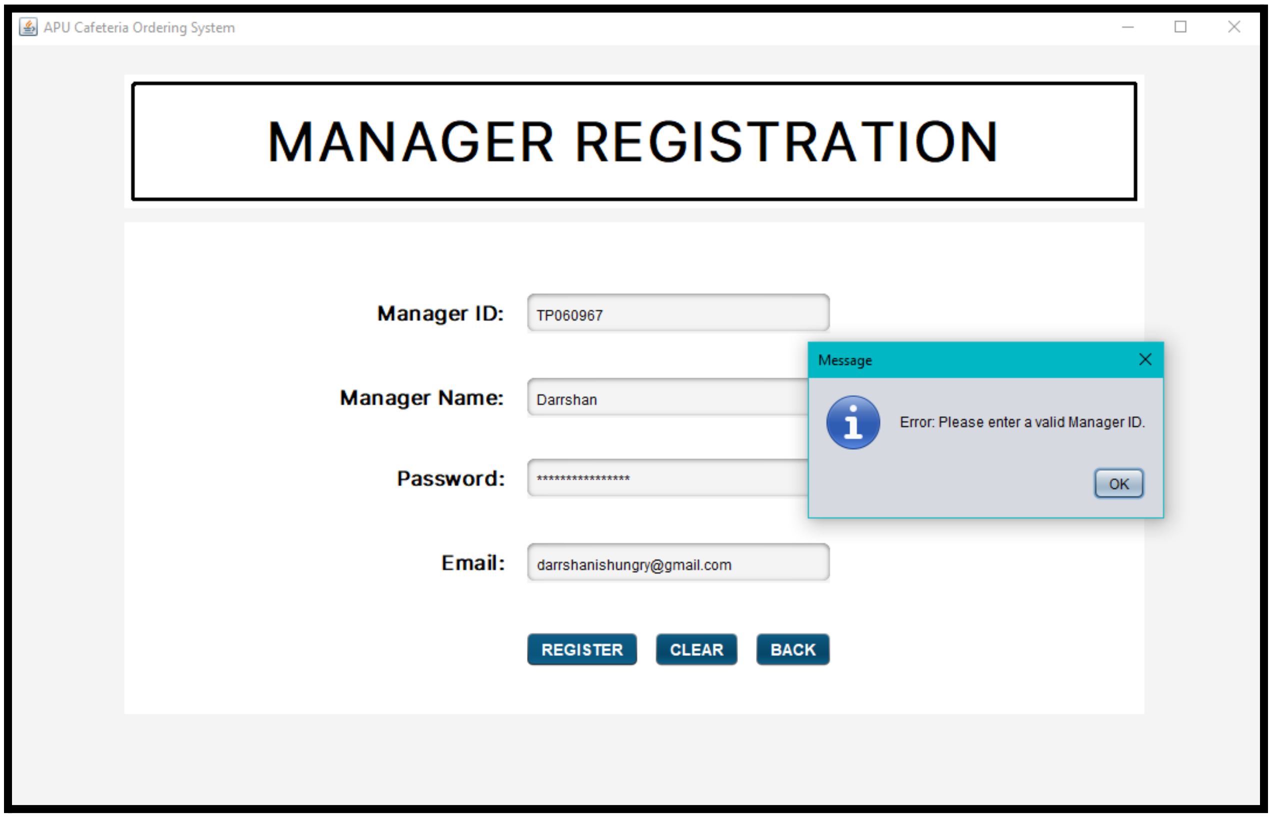
Daily Sales Report (02-09-2022)

Item ID	Item Name	Quantity	Total Sales
KO01	BULGOGI	1	RM18.0
JP01	KATSUDON	2	RM68.55
JP04	YUZUICEDRINK	1	RM8.0
IT03	CREAMYLINGUINI	1	RM18.6
TW01	BOBAMILKTEA	5	RM111.2
IT01	AGLIOOLIO	2	RM68.4
IT02	LASAGNA	1	RM17.5
IT04	AMERICANO	1	RM11.0
Total Sales			RM233.9
System Balance			RM233.9
Total Orders			6
Total Customers			5

Figure 5.2.4 - Daily Sales Report on 02-09-2022 in .pdf format

5.3 Duplication/Unique Value Check

In order to ensure that no two users share the same ID, a duplication check is done in the text file to validate it. If an identical ID is found within the text file, an error message will be shown to prevent the user from registering with the same ID. Figure 5.3.1 is an example of a duplication check done on the Manager Registration page for the Manager ID.


APU Cafeteria Ordering System

MANAGER REGISTRATION

Manager ID: TP060967

Manager Name: Darrshan

Password: ****************

Email: darrshanishungry@gmail.com

Message

Error: Please enter a valid Manager ID.

Figure 5.3.1 - Unique ID check on Manager Registration Page

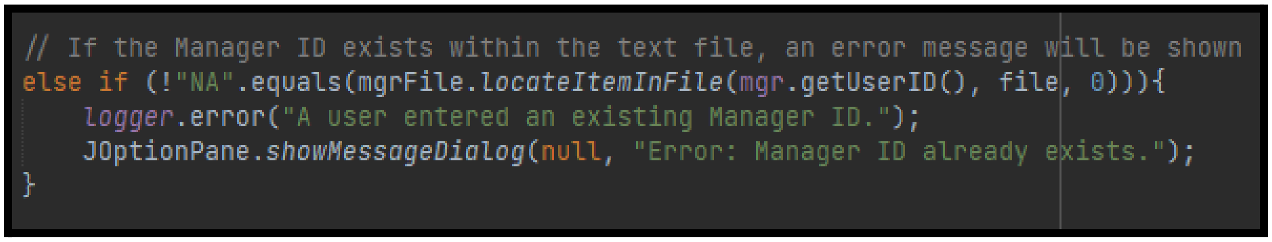
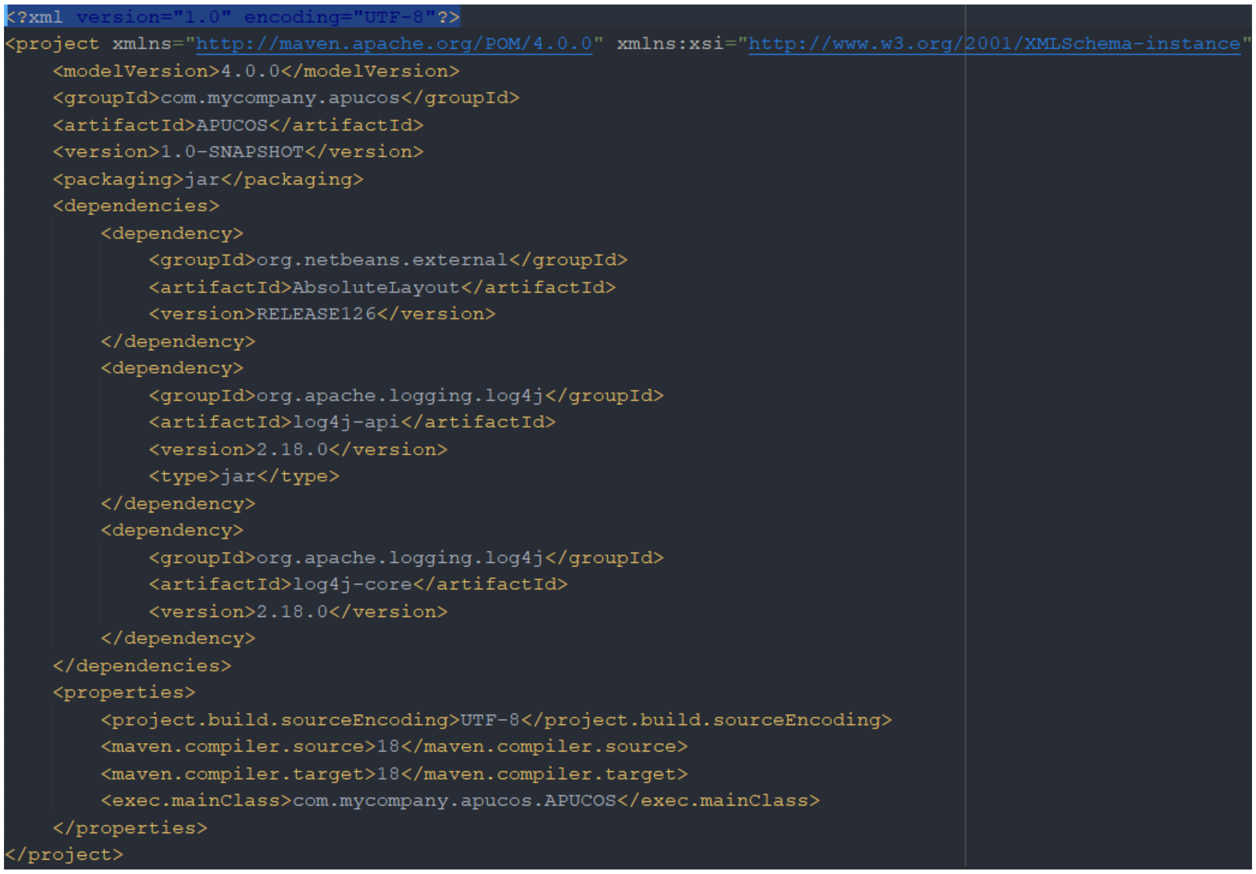

```
// If the Manager ID exists within the text file, an error message will be shown
else if (!"NA".equals(mgrFile.locateItemInFile(mgr.getUserID(), file, 0))){
    logger.error("A user entered an existing Manager ID.");
    JOptionPane.showMessageDialog(null, "Error: Manager ID already exists.");
}
```

Figure 5.3.2 - Code used to search for identical ID in the file

The code in Figure 5.3.2 uses a method to search for the file for an identical Manager ID to the one keyed in by the user. If the two are the same, then an error message is shown telling the user that they would need to enter a unique Manager ID as it already exists in the file.

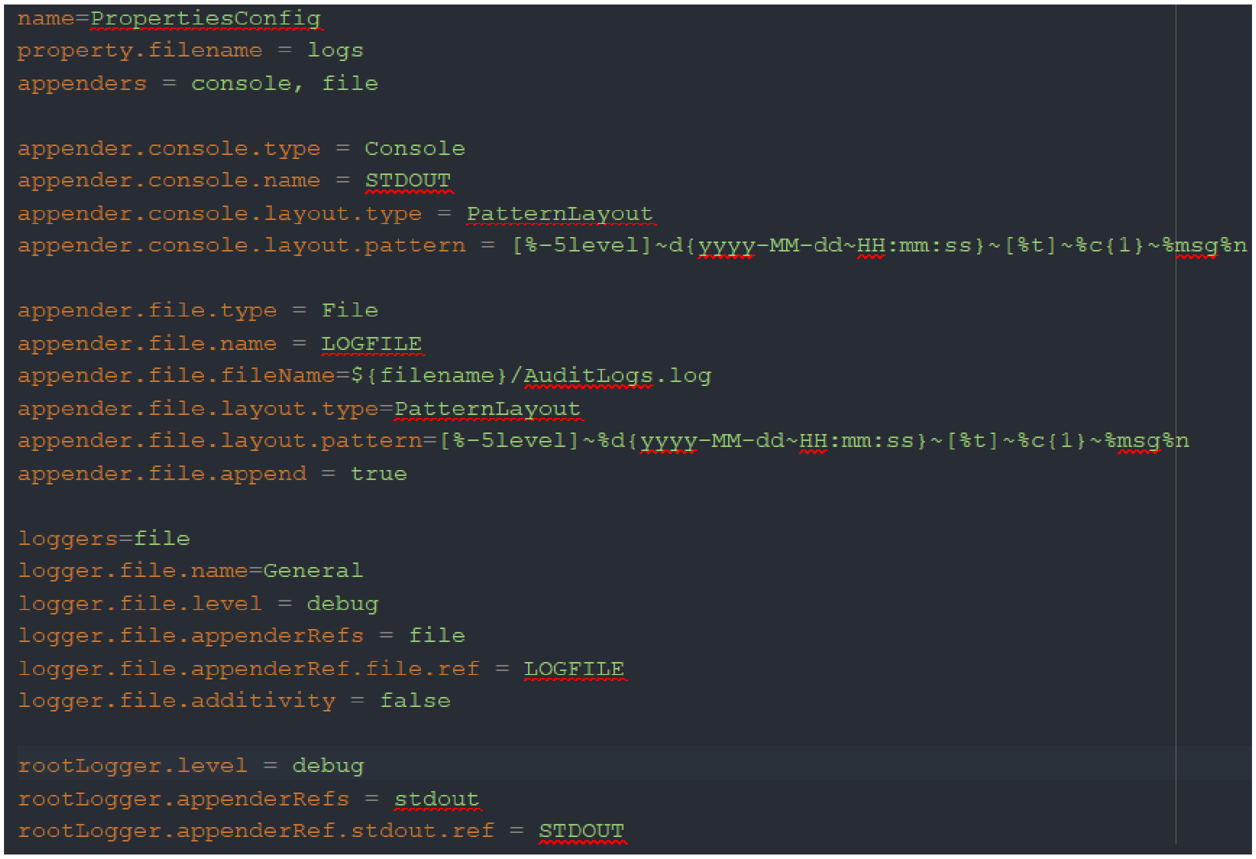
5.4 Logging using Log4j

Logging is done in the system for debugging and recording operation information. This feature is implemented with the help of an external library called log4j2. To implement it, the pom.xml file has to be altered, and a new log4j2.properties has to be added to the project.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.apucos</groupId>
    <artifactId>APUCOS</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <dependencies>
        <dependency>
            <groupId>org.netbeans.external</groupId>
            <artifactId>AbsoluteLayout</artifactId>
            <version>RELEASE126</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.18.0</version>
            <type>jar</type>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.18.0</version>
        </dependency>
    </dependencies>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <exec.mainClass>com.mycompany.apucos.APUCOS</exec.mainClass>
    </properties>
</project>
```

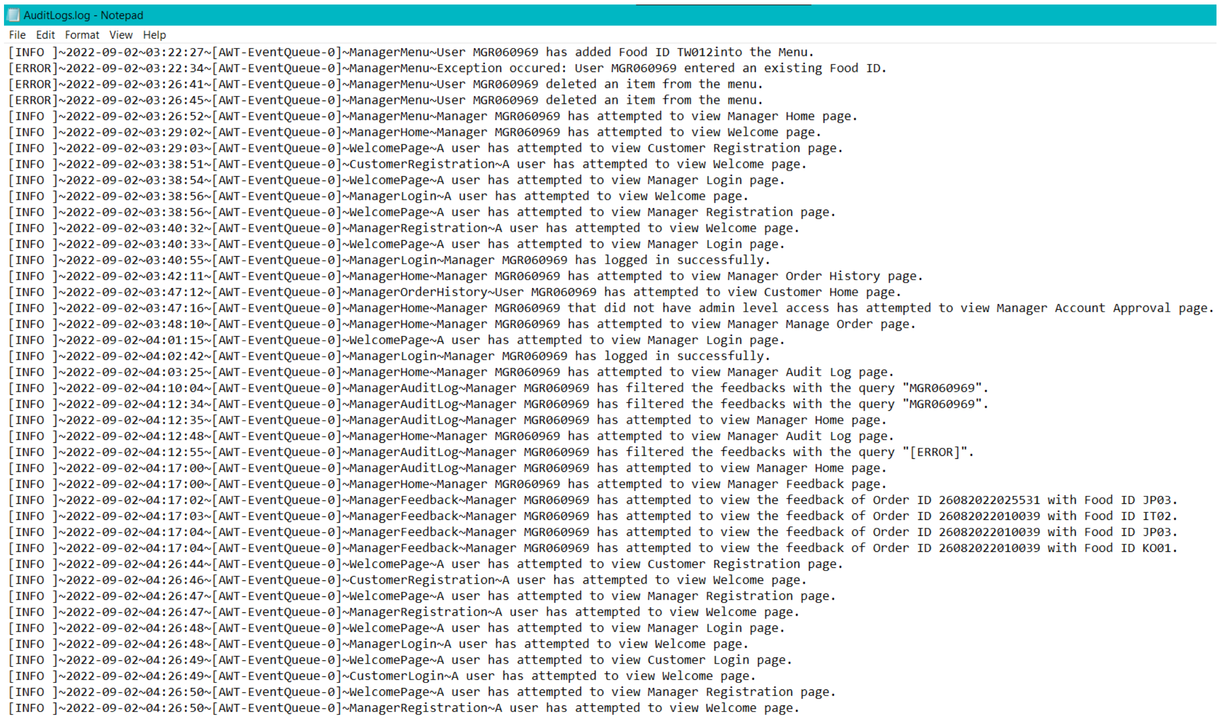
Figure 5.4.1 - pom.xml



```
name=PropertiesConfig
property.filename = logs
appenders = console, file

appender.console.type = Console
appender.console.name = STDOUT
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = [%-5level]~d{yyyy-MM-dd~HH:mm:ss}~[%t]~%c{1}~%msg%n

appender.file.type = File
appender.file.name = LOGFILE
appender.file.fileName=${filename}/AuditLogs.log
appender.file.layout.type=PatternLayout
appender.file.layout.pattern=[%-5level]~%d{yyyy-MM-dd~HH:mm:ss}~[%t]~%c{1}~%msg%n
appender.file.append = true

loggers=file
logger.file.name=General
logger.file.level = debug
logger.file.appenderRefs = file
logger.file.appenderRef.file.ref = LOGFILE
logger.file.additivity = false

rootLogger.level = debug
rootLogger.appenderRefs = stdout
rootLogger.appenderRef.stdout.ref = STDOUT
```

Figure 5.4.2 - log4j2.properties

These logs can be found in the AuditLogs.log file, which includes the severity level, date, time, page, and more details. The logs are created for every action done in the system and shows which user does it, for instance, when a user has attempted to access a page and view the feedback of an order.



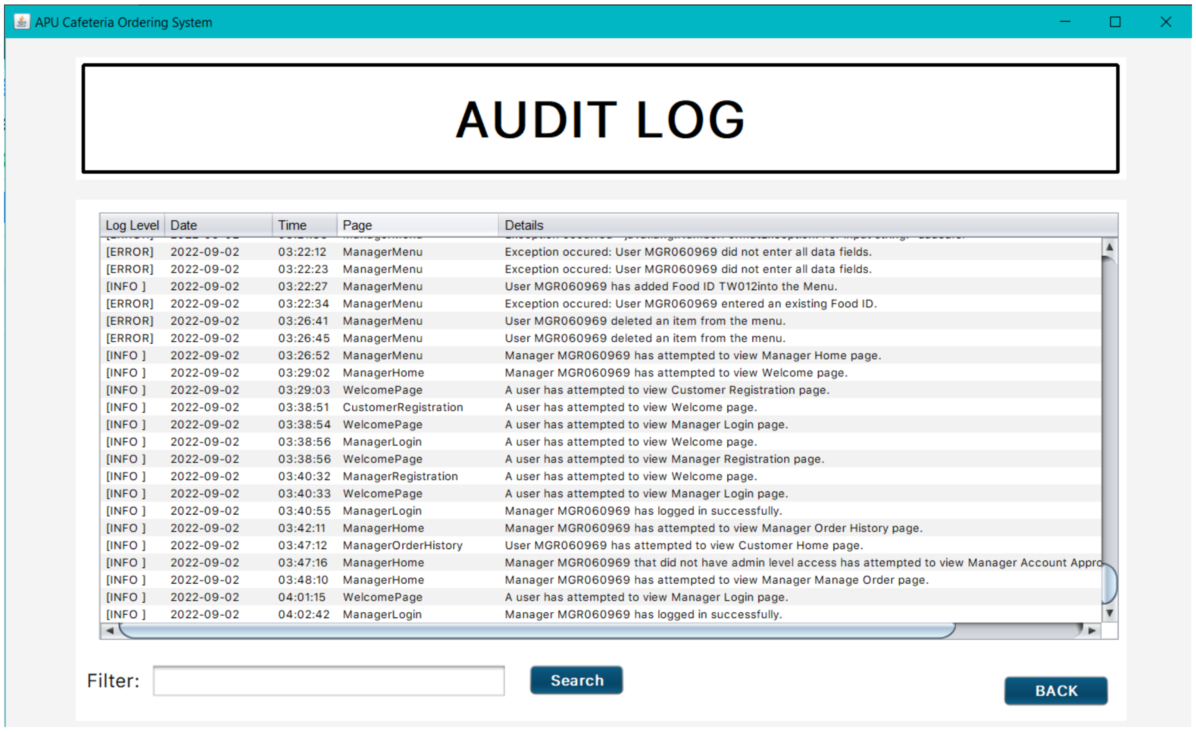
AuditLogs.log - Notepad

File Edit Format View Help

```
[INFO ]~2022-09-02~03:22:27~[AWT-EventQueue-0]~ManagerMenu~User MGR060969 has added Food ID TW012into the Menu.
[ERROR]~2022-09-02~03:22:34~[AWT-EventQueue-0]~ManagerMenu~Exception occured: User MGR060969 entered an existing Food ID.
[ERROR]~2022-09-02~03:26:41~[AWT-EventQueue-0]~ManagerMenu~User MGR060969 deleted an item from the menu.
[ERROR]~2022-09-02~03:26:45~[AWT-EventQueue-0]~ManagerMenu~User MGR060969 deleted an item from the menu.
[INFO ]~2022-09-02~03:26:52~[AWT-EventQueue-0]~ManagerMenu~Manager MGR060969 has attempted to view Manager Home page.
[INFO ]~2022-09-02~03:29:02~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Welcome page.
[INFO ]~2022-09-02~03:29:03~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Customer Registration page.
[INFO ]~2022-09-02~03:38:51~[AWT-EventQueue-0]~CustomerRegistration~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~03:38:54~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Login page.
[INFO ]~2022-09-02~03:38:56~[AWT-EventQueue-0]~ManagerLogin~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~03:38:56~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Registration page.
[INFO ]~2022-09-02~03:40:32~[AWT-EventQueue-0]~ManagerRegistration~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~03:40:33~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Login page.
[INFO ]~2022-09-02~03:40:55~[AWT-EventQueue-0]~ManagerLogin~Manager MGR060969 has logged in successfully.
[INFO ]~2022-09-02~03:42:11~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Manager Order History page.
[INFO ]~2022-09-02~03:47:12~[AWT-EventQueue-0]~ManagerOrderHistory~User MGR060969 has attempted to view Customer Home page.
[INFO ]~2022-09-02~03:47:16~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 that did not have admin level access has attempted to view Manager Account Approval page.
[INFO ]~2022-09-02~03:48:10~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Manager Manage Order page.
[INFO ]~2022-09-02~04:01:15~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Login page.
[INFO ]~2022-09-02~04:02:42~[AWT-EventQueue-0]~ManagerLogin~Manager MGR060969 has logged in successfully.
[INFO ]~2022-09-02~04:03:25~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Manager Audit Log page.
[INFO ]~2022-09-02~04:10:04~[AWT-EventQueue-0]~ManagerAuditLog~Manager MGR060969 has filtered the feedbacks with the query "MGR060969".
[INFO ]~2022-09-02~04:12:34~[AWT-EventQueue-0]~ManagerAuditLog~Manager MGR060969 has filtered the feedbacks with the query "MGR060969".
[INFO ]~2022-09-02~04:12:35~[AWT-EventQueue-0]~ManagerAuditLog~Manager MGR060969 has attempted to view Manager Home page.
[INFO ]~2022-09-02~04:12:48~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Manager Audit Log page.
[INFO ]~2022-09-02~04:12:55~[AWT-EventQueue-0]~ManagerAuditLog~Manager MGR060969 has filtered the feedbacks with the query "[ERROR]".
[INFO ]~2022-09-02~04:17:00~[AWT-EventQueue-0]~ManagerAuditLog~Manager MGR060969 has attempted to view Manager Home page.
[INFO ]~2022-09-02~04:17:00~[AWT-EventQueue-0]~ManagerHome~Manager MGR060969 has attempted to view Manager Feedback page.
[INFO ]~2022-09-02~04:17:02~[AWT-EventQueue-0]~ManagerFeedback~Manager MGR060969 has attempted to view the feedback of Order ID 26082022025531 with Food ID JP03.
[INFO ]~2022-09-02~04:17:03~[AWT-EventQueue-0]~ManagerFeedback~Manager MGR060969 has attempted to view the feedback of Order ID 26082022010039 with Food ID IT02.
[INFO ]~2022-09-02~04:17:04~[AWT-EventQueue-0]~ManagerFeedback~Manager MGR060969 has attempted to view the feedback of Order ID 26082022010039 with Food ID JP03.
[INFO ]~2022-09-02~04:17:04~[AWT-EventQueue-0]~ManagerFeedback~Manager MGR060969 has attempted to view the feedback of Order ID 26082022010039 with Food ID KO01.
[INFO ]~2022-09-02~04:26:44~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Customer Registration page.
[INFO ]~2022-09-02~04:26:46~[AWT-EventQueue-0]~CustomerRegistration~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~04:26:47~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Registration page.
[INFO ]~2022-09-02~04:26:47~[AWT-EventQueue-0]~ManagerRegistration~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~04:26:48~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Login page.
[INFO ]~2022-09-02~04:26:48~[AWT-EventQueue-0]~ManagerLogin~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~04:26:49~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Customer Login page.
[INFO ]~2022-09-02~04:26:49~[AWT-EventQueue-0]~CustomerLogin~A user has attempted to view Welcome page.
[INFO ]~2022-09-02~04:26:50~[AWT-EventQueue-0]~WelcomePage~A user has attempted to view Manager Registration page.
[INFO ]~2022-09-02~04:26:50~[AWT-EventQueue-0]~ManagerRegistration~A user has attempted to view Welcome page.
```

Figure 5.4.3 - AuditLogs.log

The logs are also outputted in a table format in the Manager Audit Log page.



APU Cafeteria Ordering System

AUDIT LOG

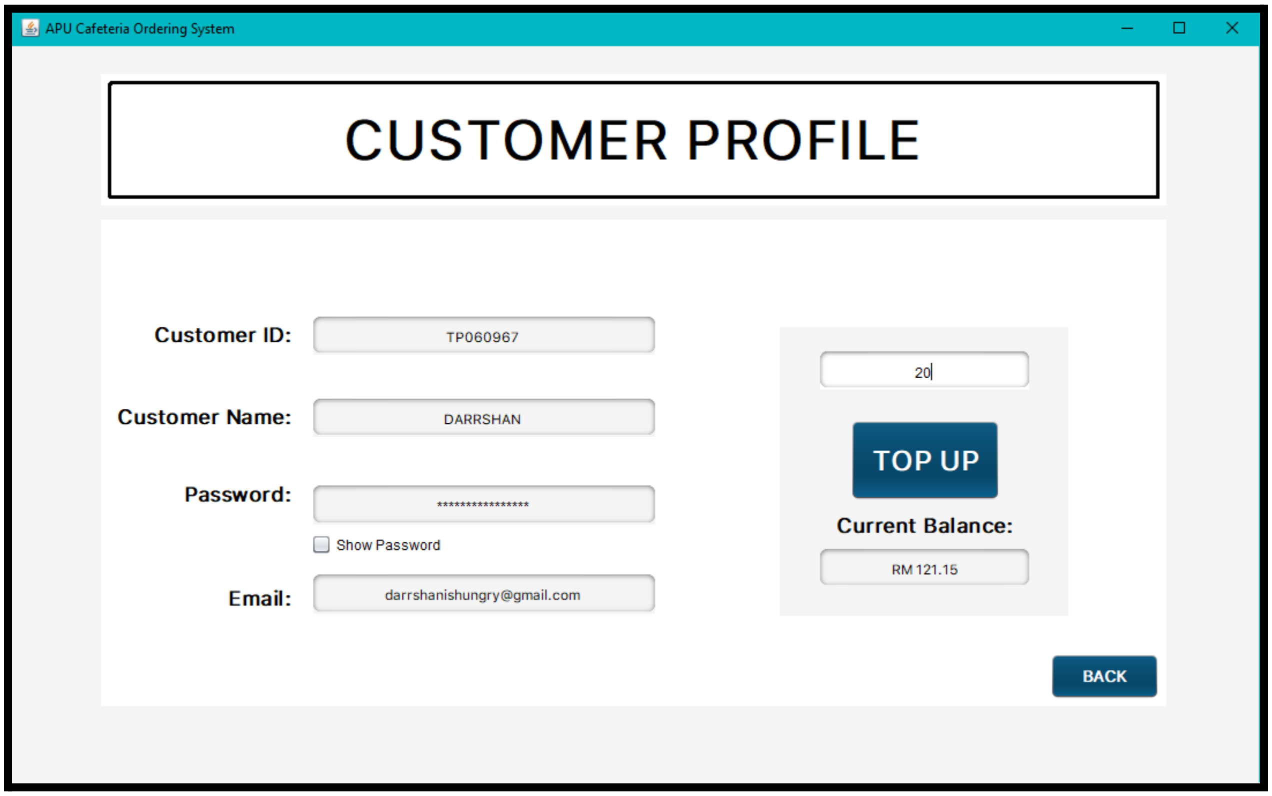
Log Level	Date	Time	Page	Details
[ERROR]	2022-09-02	03:22:12	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[ERROR]	2022-09-02	03:22:23	ManagerMenu	Exception occured: User MGR060969 did not enter all data fields.
[INFO]	2022-09-02	03:22:27	ManagerMenu	User MGR060969 has added Food ID TW012into the Menu.
[ERROR]	2022-09-02	03:22:34	ManagerMenu	Exception occured: User MGR060969 entered an existing Food ID.
[ERROR]	2022-09-02	03:26:41	ManagerMenu	User MGR060969 deleted an item from the menu.
[ERROR]	2022-09-02	03:26:45	ManagerMenu	User MGR060969 deleted an item from the menu.
[INFO]	2022-09-02	03:26:52	ManagerMenu	Manager MGR060969 has attempted to view Manager Home page.
[INFO]	2022-09-02	03:29:02	ManagerHome	Manager MGR060969 has attempted to view Welcome page.
[INFO]	2022-09-02	03:29:03	WelcomePage	A user has attempted to view Customer Registration page.
[INFO]	2022-09-02	03:38:51	CustomerRegistration	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:38:54	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	03:38:56	ManagerLogin	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:38:56	WelcomePage	A user has attempted to view Manager Registration page.
[INFO]	2022-09-02	03:40:32	ManagerRegistration	A user has attempted to view Welcome page.
[INFO]	2022-09-02	03:40:33	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	03:40:55	ManagerLogin	Manager MGR060969 has logged in successfully.
[INFO]	2022-09-02	03:42:11	ManagerHome	Manager MGR060969 has attempted to view Manager Order History page.
[INFO]	2022-09-02	03:47:12	ManagerOrderHistory	User MGR060969 has attempted to view Customer Home page.
[INFO]	2022-09-02	03:47:16	ManagerHome	Manager MGR060969 that did not have admin level access has attempted to view Manager Account Appro
[INFO]	2022-09-02	03:48:10	ManagerHome	Manager MGR060969 has attempted to view Manager Manage Order page.
[INFO]	2022-09-02	04:01:15	WelcomePage	A user has attempted to view Manager Login page.
[INFO]	2022-09-02	04:02:42	ManagerLogin	Manager MGR060969 has logged in successfully.

Filter: Search BACK

Figure 5.4.4 - Manager Audit Log Page

5.5 Top Up

The top up feature is done in the system for the purpose of allowing the Customers to update the balance in their accounts. The customers can access this feature in the Customer Profile pages where there is a section for them to add on to the balance within their account. The implementation of this feature enables the system to be fully cashless which is far more convenient.



Customer ID: TP060967

Customer Name: DARRSHAN

Password: ***************

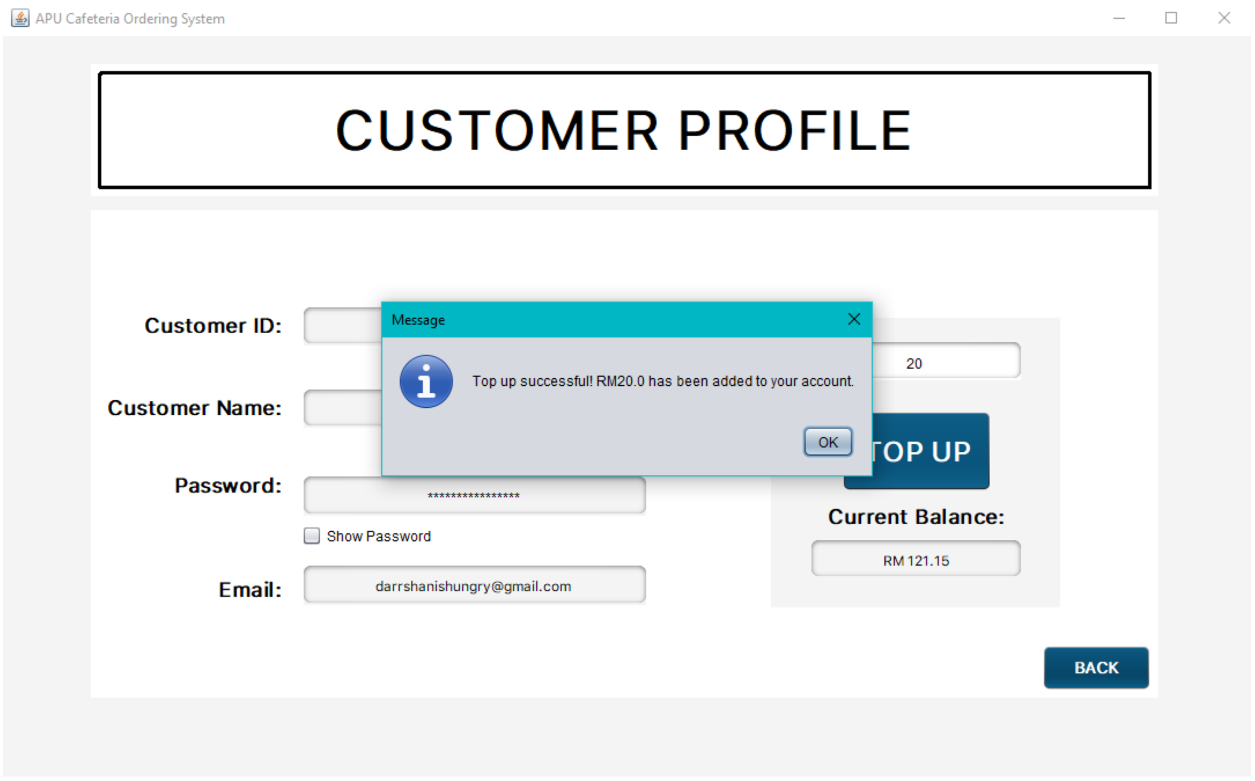
Email: darrshanishungry@gmail.com

20

Current Balance: RM 121.15

Figure 5.5.1 - Customer Profile Page with Top Up Feature

Within the page, the customer can input the amount of their choosing as long as it is within the range of RM 1 to RM 100 to top up. The system will validate their input and proceed to add the amount to their current balance within their account and display it to them below the "Top Up" button.



APU Cafeteria Ordering System

CUSTOMER PROFILE

Message

Top up successful! RM20.0 has been added to your account.

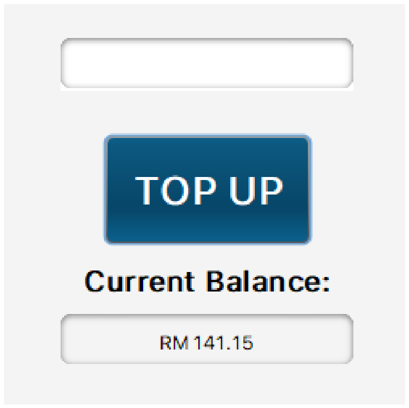
Password: ****************

Email: darrshanishungry@gmail.com

20

Current Balance: RM 121.15

Figure 5.5.2 - Customer Profile Page with Successful Top Up



Current Balance: RM 141.15

Figure 5.5.3 - Updated Balance

As seen in Figure 5.5.2, the top up was successful for this particular customer with the balance being added to their account. Figure 5.5.3 shows the new balance of the user which proves that the amount has been updated in the account.

5.0 Conclusion

As demonstrated by this project, Object Oriented Concepts have proven to be useful for complex system development. The implementation of these concepts has enabled our project to achieve code stability and flexibility. Consequently it has aided us in achieving our goal of creating a more convenient and efficient cafeteria ordering system for Asia Pacific University. Furthermore, by applying certain OOP Concepts it will also allow the code to be more adaptable and easier for maintenance and improvement in the future if new features were to be implemented.

6.0 References

Braunschweig, D. (2018). *Encapsulation – Programming Fundamentals*. Pressbooks.

https://press.rebus.community/programmingfundamentals/chapter/encapsulation/

Doherty, E. (2020). *What is object-oriented programming? OOP explained in depth*. Educative: Interactive Courses for Software Developers.

https://www.educative.io/blog/object-oriented-programming

GeeksforGeeks. (2022, July 11). *Classes and Objects in Java*.

https://www.geeksforgeeks.org/classes-objects-java/

Gupta, L. (2022). *Guide to Polymorphism*. HowToDoInJava.

https://howtodoinjava.com/java/oops/what-is-polymorphism-in-java/

Janssen, T. (2022). *OOP Concepts for Beginners: What is Polymorphism*. Stackify.

https://stackify.com/oop-concept-polymorphism/#:%7E:text=Polymorphism%20is%20one%20of%20the,types%20through%20the%20same%20interface.

Javatpoint. (n.d.) *Difference between object and class - javatpoint*.

https://www.javatpoint.com/difference-between-object-and-class

Janssen, T. (2021). *OOP Concept for Beginners: What is Inheritance?* Stackify.

https://stackify.com/oop-concept-inheritance/

Muldrow, L. (2020, December 9). *What is DRY Development?* DigitalOcean Community.

https://www.digitalocean.com/community/tutorials/what-is-dry-development

Programiz. (n.d.) *Java Inheritance (With Examples)*.

https://www.programiz.com/java-programming/inheritance

SHA-256 Hash in Java. (2018). GeeksforGeeks. https://www.geeksforgeeks.org/sha-256-hash-in-java/

Tutoralspoint. (2022). *Java String substring() Method example.* https://www.tutorialspoint.com/Java-String-substring-Method-example#:%7E:text=The%20substring(int%20beginIndex%2C%20int,the%20substring%20is%20endIndex%2Dbeg inIndex.

Vats, R. (2022). *Modularity in Java Explained With Step by Step Example [2022]*. upGrad Blog. https://www.upgrad.com/blog/modularity-in-java/#:%7E:text=good%20web%20architecture%3F-,What%20is%20Modularity%20in%20Java%3F,than%20a%20single%20legacy%20architecture.

W3schools. (n.d.). *Java Interface*. https://www.w3schools.com/java/java_interface.asp

W3schools. (2022). *Java Modifiers*. https://www.w3schools.com/java/java_modifiers.asp