FORM 1

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Diploma in Software Engineering

Programme: Diploma in Software Engineering (Group: 1)

Assignment

AMSE1003 SOFTWARE ENGINEERING

Name (Block Letters)	Registration No.	Signature	Marks
1. Shafina Siauye Lim Binti Abdul Malek Lim	24SMD07518	shafina	
2. Pang Jia Yie	24SMD00687	jiayie	
3. Phoebe Lo Yin Yue	24SMD07095	phoebe	
4. Rowan Yee Xiao Peng	24SMD07554	rowan	
5. Jasper Wong Yeu	24SMD01683	jasper	

Lecturer's Name: Ms. Surayaini Binti Basri

Date of Submission: 29 September 2024

FORM 2

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Plagiarism Statement and Guideline for Late Submission of Coursework

Read, complete, and sign this statement to be submitted with the written report.

We confirm that the submitted works are all our own work and are in our own words.

Name (Block Letters)	Registration No.	Signature	Date
1. Shafina Siauye Lim Binti Abdul Malek Lim	24SMD07518	shafina	
2. Pang Jia Yie	24SMD00687	jiayie	
3. Phoebe Lo Yin Yue	24SMD07095	phoebe	
4. Rowan Yee Xiao Peng	24SMD07554	rowan	
5. Jasper Wong Yeu	24SMD01683	jasper	

Table of Contents

1. Part 1

1.1 Problems of the existing system

1.1.1 Description of Menu Lee:

Our chosen organisation is Menu Lee, a popular food stall located in our university campus's canteen, frequented by students, lecturers, university staff, and occasional visitors. The stall offers a variety of food and beverage options, catering primarily to the campus community. Despite its popularity, Menu Lee continues to operate its business manually, which leads to several challenges.

1.1.2 Major problems of the manual process:

1. **Inefficiency and slow service**

The manual process of taking orders and payment handling slows down service significantly. Orders are written by hand and customers are given number tags, which the staff must call out when the order is ready. Staff typically shout out numbers multiple times until the customer comes to them and hands back the number tag to them in order to get their ordered item. Additionally, online payments require customers to show proof of payment to staff, which adds to wait times when the staff are busy. This inefficiency can lead to long wait times and customer dissatisfaction.

2. **Human error**

The reliance on manual order-taking and cash handling increases the potential for human error. Staff may mishear orders, leading to incorrect items being served, or they may miscalculate bills, resulting in financial discrepancies and customer complaints. Besides, paper orders can be misplaced or lost.

3. **Limited data analysis**

The current system cannot effectively track key performance indicators such as peak hours, popular menu items, and least popular menu items. This limitation prevents management from making informed decisions about inventory, staffing, and menu offerings.

4. **Excessive workload**

The current system relies heavily on human labor when taking orders and serving customers. Staff have trouble when taking orders due to the noisy environment which can cause miscommunication. This workload can lead to burnout and reduced service quality, ultimately impacting customer satisfaction.

5. **Lack of customer feedback**

Customers are having trouble sharing their experiences due to Menu Lee does not hava a structured feedback system for collecting customer feedback. This causes the management to have difficulty identifying areas for improvement, resulting in missed opportunities to enhance service.

1.2 Software quality attributes of the project

1. Acceptability

The software must be acceptable, understandable, and usable for users to do their tasks. We anticipate that our system will be easy for users to use with only a few clicks. For instance, users only need to click the “Order” button then “Add Order” and finally “Confirm Order” to order their food. This reduces complexity and ensures that even users with limited technical skills can easily use the system. Besides, all the terms used within the system are simple and familiar to users, ensuring they can easily understand how to use the system without additional training.

Assumption: The target users (students, staff, and visitors) are familiar with basic point-and-click systems and prefer minimal interactions when ordering food.

2. Dependability and security

The new system must be dependable, meaning it should operate reliably without frequent failures or downtime. Security is also important, especially when handling sensitive data such as user data and financial details. To ensure security, the system will implement Multi-Factor Authentication (MFA) which requires users to log in with both a password and a One-Time Password (OTP). This prevents unauthorized access and helps protect users' financial data.

Assumption: The system will be handling sensitive customer information, such as payment details, that need protection from cyber threats.

3. Efficiency

Efficiency is critical for both performance and resource usage. The system should maximize its output while minimizing the resources it consumes (memory, processing power, etc.). In this case, the online ordering process is designed to be fast and lightweight, allowing customers to complete orders within approximately 1 minute, compared to the 5 to 10 minutes taken by the manual process. This reduction in time not only improves customer satisfaction but also increases the food stall's throughput. Additionally, the software will be optimized to avoid any unnecessary background processes or memory usage, ensuring the system runs smoothly even during peak hours.

Assumption: The current manual process is slow due to human interaction, and the new system will handle many customers simultaneously during peak times.

4. Maintainability

The system should be written in such a way that it can evolve to meet the changing needs of customers. Therefore, the system can be easily modified to fix defects, add new features, or improve performance. For example, the system could easily evolve to accommodate new features based on customer feedback and client requests, such as personalized recommendations or real-time tracking of orders. Regular updates should be simple to implement without disrupting the user experience, ensuring that the system stays relevant and continues to meet user needs.

Assumption: As customer preferences evolve, the system will need to adapt by adding new features or making improvements based on feedback.

1.3 Software Process Model

1.3.1 Recommendation and Explanation:

We recommend using Extreme Programming (XP) as the software process model for developing the new ordering system for Menu Lee. This is because Extreme Programming is an Agile software development framework that focuses on customer satisfaction, continuous feedback, and adaptability. It uses practices like frequent releases, pair programming, test-driven development (TDD), and continuous integration, to ensure the system is developed iteratively, with regular feedback from stakeholders. With XP, the development team can break down the project into smaller tasks and deliver them incrementally, ensuring that the most critical features—like online ordering and payment—are implemented early. The iterative nature of XP means that any issues with functionality or usability can be addressed immediately, resulting in a more reliable and user-friendly system.

1.3.2 Justification:

We chose XP to develop our system because it promotes rapid development cycles and continuous feedback, which are crucial for Menu Lee's needs. Frequent releases will allow the food stall to start using parts of the system early, such as order placement, and progressively add new features, like menu analysis and customer feedback. Pair programming improves code quality and speeds up problem-solving, while test-driven development ensures that each part of the system is thoroughly tested before deployment, minimizing errors and enhancing reliability.

Furthermore, incremental development allows the team to prioritize critical features like the order placement function while gradually adding other modules like customer feedback or inventory tracking. This model also supports constant communication with Menu Lee's staff and customers, ensuring that their evolving needs are met in real time. XP's adaptability is particularly beneficial in a food stall environment where user expectations and operational needs may shift.

1.3.3 Assumptions to support this suggestion:

To successfully implement Extreme Programming, several key assumptions are heavily relied on. Firstly, active and ongoing feedback from Menu Lee's staff and customers is assumed to be available throughout the development process. Regular feedback will ensure that the development team can prioritize features and make necessary adjustments quickly, ensuring that the system is always up to par with the needs of staff and customers. Additionally, a development team with expertise in software development, user interface design, and operations management is assumed to be available. Pair programming requires collaboration, while TDD requires team members familiar with writing automated tests. Finally, access to the necessary hardware, software tools, and network infrastructure is also crucial. The team will need the tools to support XP practices such as continuous integration, automated testing, and version control. Additionally, a stable network infrastructure is required to ensure the smooth operation of the ordering and payment system.

2. Part 2

2.1 Project Plan and Schedule

2.1.1 Task Allocation

AMSE1003 SOFTWARE ENGINEERING

Tasks Allocation

Indicate (√) in member name column if he/she have involved in that task.

Tasks		Pang Jia Yie	Phoebe	Shafina	Rowan	Jasper
1.	Problems of existing system	✓	✓	✓	✓	✓
2.	Software quality attributes of the project	✓	✓	✓	✓	✓
3.	Software Process Model		✓	✓	✓	
4.	Project Plan and Schedule	✓			✓	
5.	Software Requirements Specification	✓	✓	✓	✓	✓
6.	An architectural design	✓	✓	✓		
7.	Test cases	✓	✓	✓	✓	✓
8.	Software Configuration Management			✓		
9.	Reference		✓	✓		
10.	Appendices section			✓		
11.	Editing and Proofreading		✓	✓		
12.	Formatting	✓		✓	✓	
13.	Table of contents		✓			

2.1.2 Gantt Chart

2.2 Software Requirements Specification

Ordering module:

Functional requirements:

1.1 The system shall allow users to easily search the menu by category (e.g., drinks, snacks, meals) or by specific items using keywords.

1.2 The system shall facilitate order placement from smartphones, tablets, and laptops, ensuring a responsive and user-friendly interface across all devices.

1.3 The system shall display a visually appealing menu that includes clear item names, high-quality images, and prices in the menu section.

1.4 The system shall automatically calculate the total amount and total cost of the order, dynamically updating as users modify items.

1.5 The system shall allow users access to view their past orders, including detailed itemized receipts with date, time, total price, and payment method used.

1.6 The system shall allow users to customize their orders, providing options to add extra toppings, alter ingredients, and submit special requests (e.g., vegetarian, allergy alerts).

Payment module:

Functional requirement:

2.1 The system shall allow users to choose various payment methods upon checkout.

2.2 The system should be integrated with the banking API.

2.3 The system shall allow users to complete the checkout process independently.

Customer feedback module:

Functional requirements:

3.1 The system shall allow users to submit reviews through the feedback section.

3.2 The system shall allow users to upload photos of their food through the comments section.

3.3 The system shall allow users to rate by giving stars.

Reporting and analytics module:

Functional requirements:

4.1 The system shall generate daily sales reports in PDF or CSV format for easy review by management.

4.2 The system shall track sales by individual items, showing which dish is the most popular.

4.3 The system shall generate the ranking of the least popular food monthly.

User module:

Functional requirements:

5.1 The system should enable users to create accounts.

5.2 The system shall enable users to log in and log out of accounts.

5.3 The system should notify the user of the progress of the food such as the received order, in the kitchen, and completed.

Non-functional requirements:

1.1 The system must be available during business hours which are from 7 a.m. until 3.30 p.m. on weekdays and should provide notifications of any planned maintenance or downtime outside of these hours.

1.2 The software should be portable across Windows, macOS, and Linux operating systems.

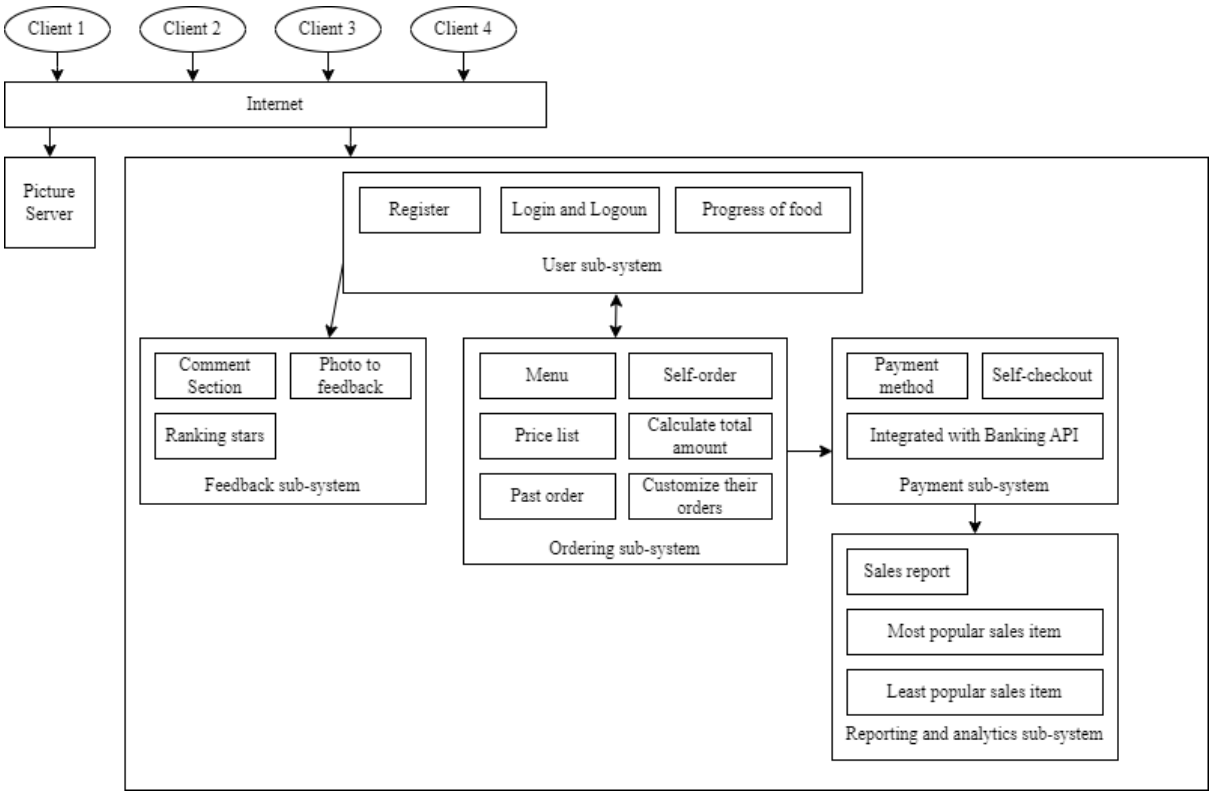
1.3 The system shall not disclose any personal information of system users apart from their usernames, and all user data should be protected using encryption and secure communication protocols.

1.4 The user interface shall use modern web technologies like HTML5 and CSS for simple and responsive design.

1.5 The system shall occupy at most 200MB of storage on the user's device, including all necessary resources and data files.

2.3 An architectural design

2.3.1 Client-server model



2.3.2 Explanation, justification and assumption:

We suggest using the client-server model for the proposed system. The Client-Server model is a well-established architecture in which the system is divided into two main components: clients (the front-end for users) and servers (the back-end for processing and storage). We chose this model because it separates the client and the server, allowing the system to distribute the workload efficiently.

This division makes it easier to manage different aspects of the system. For example, user interactions such as getting orders and feedback are handled smoothly on the client side, while the server can focus on calculating totals, processing payments, and generating reports for staff.

We assumed that there would be a stable and reliable network connection between the client and server components, allowing them to communicate effectively. Also, the server must have sufficient capacity (in terms of processing power and storage) to handle tasks such as payment processing, generating reports, and storing data. Since customer data (like orders and payment information) will be stored on the server, we assume that appropriate security measures, such as encryption and authentication protocols, will be in place to protect sensitive information.

In conclusion, this model really ensures that the system is both user-friendly for customers and provides valuable business insights to the staff, which is really suitable for the proposed food ordering system for Menu Lee.

3. Part 3

3.1 Test cases

1.1.1

Test Case Name	1.1.1 To search for valid and non-existing menu items.	**Test Case Description**	Verify that users can search the menu by category or specific items using keywords and verify that the system can handle invalid search queries properly.
Pre-conditions:		**Test Data:**	
1	The user is logged into the system.	1	Category: "Drinks"
2	The menu categories should exist in the database. (e.g., drinks, snacks, meals)	2	Keyword: "Coffee"
3	Items have keywords associated with them (e.g., coffee, meatball).	3	Invalid keyword: “RandomFood123”

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	The user selects the "Drinks" category on the menu page.	The system displays all items under the "Drinks" category.		
2	The user enters the keyword "Coffee" in the search bar at the top of the menu page.	The system displays all items with the keyword "Coffee".		
3	The user enters the keyword “RandomFood123” in the search bar at the top of the menu page.	The system shows a message “No items found”.		

1.2.1

Test Case Name	1.2.1 To place an order using a smartphone.	**Test Case Description**	Verify that users can place orders using smartphones.
Pre-conditions:		**Test Data:**	
1	The user has a smartphone.	1	Selected item: “Coffee”
2	The system interface is accessible via browser or app.	2	Quantity: 2
3	The user is logged into their account.	3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	The user selects "Coffee" from the menu page by clicking the “order” button to go to the order page.	The system adds "Coffee" to the order list. The order page that displays the menu is displayed.		
2	The user selects the quantity as "2" on the order page.	The system updates the order to show 2 coffees.		
3	The user clicks the “confirm order” button to place an order on the order page.	The system shows a message “The order is successfully placed”.		

1.3.1

Test Case Name	1.3.1 To display menu with item names, prices, and pictures	**Test Case Description**	Verify that the system displays the menu clearly with names, prices, and images.
Pre-conditions:		**Test Data:**	
1	The user is logged into the system.	1	Item: “Coffee”
2	Menu items have been configured with names, prices, and images in the database.	2	Price: RM 2
3	The menu interface is functional.	3	Image: coffee.jpg (an image of a coffee)

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	The user navigates to the menu section.	The system displays all available items with clear names, prices, and corresponding images.		
2	The user checks the name, price, and picture of the item "Coffee” on the menu page.	The system displays "Coffee" with a price of RM2 and the correct image (coffee.jpg).		
3	The user scrolls through the menu on the menu page.	The system continuously displays items with names, prices, and images.		

2.1.1

Test Case Name	2.1.1 To choose various payment methods upon checkout.	**Test Case Description**	Verify that users can choose from different payment methods (e.g., credit/debit card, cash) when checking out.
Pre-conditions:		**Test Data:**	
1	The user is logged in to the system.	1	Valid credit/debit card information.
2	The user has added items to the shopping cart.	2	Cash payment option.
3	The checkout page is accessible.	3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Navigate to the checkout page after adding items to the cart.	The checkout page is displayed with available payment methods.		
2	Select "Credit/Debit Card" as a payment method and enter valid card details.	The system processes the card payment and confirms the order.		
3	Select "Cash" as a payment method.	The system accepts the cash option and displays order confirmation with instructions for paying with cash.		

2.2.1

Test Case Name	2.2.1 Verify the system's integration with the banking API.	**Test Case Description**	Verify the system's integration with the banking API, allowing payments to be processed via the bank's API.
Pre-conditions:		**Test Data:**	
1	The user has a valid account in the system.	1	Valid user bank account details.
2	The banking API is functional and accessible.	2	Valid API credentials (API key, token, etc.).
3	The user has a bank account with sufficient balance for the payment.	3	Transaction amount (within allowable limits).

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Navigate to the payment page after adding items to the cart.	The payment page is displayed with the option to select a bank for processing the payment via API.		
2	Select a bank and initiate payment.	The system sends a payment request to the banking API. The API processes the transaction and returns a success response.		
3	Confirm the payment was successful.	The system displays a confirmation of the payment and generates a receipt.		

2.3.1

Test Case Name	2.3.1 Verify the system allows users to independently complete the self-checkout process.	**Test Case Description**	Verify that users can independently complete the checkout process without assistance.
Pre-conditions:		**Test Data:**	
1	The user is logged in to the system.	1	Valid user account credentials.
2	The user has added items to the shopping cart.	2	Item(s) in the shopping cart.
3	The checkout page is accessible and operational.	3	Valid payment information (e.g., credit/debit card, cash).

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Navigate to the shopping cart page.	The shopping cart page is displayed with a summary of items.		
2	Proceed to checkout and review order details.	The user can review their order and proceed to the payment section.		
3	Complete the payment by entering valid payment details.	The system processes the payment and confirms the order. The user is directed to an order confirmation page.		

3.1.1

Test Case Name	3.1.1 Give review with invalid data through feedback section	**Test Case Description**	To test the feedback section by writing review within 300 characters
Pre-conditions:		**Test Data:**	
1	The user must already login to the system	1	Feedback = “ ”
2		2	
3		3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Go to Feedback section	The feedback of Menu Lee system is displayed.		
2	Enter reviews as specified in the test data	Review is entered.		
3	Click submit button	The page with sentence “Please enter a valid review.” is displayed.		

3.2.1

Test Case Name	3.2.1 Upload valid photo through comments section	**Test Case Description**	To check the comments section by uploading one photo
Pre-conditions:		**Test Data:**	
1	The user must already login in the system	1	Comments = “photo of food”
2	The user must already placed order the food	2	
3		3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Go to menu	The menu of Menu Lee system is displayed		
2	Find and click the picture of food that is already placed order	The food details are displayed.		
3	Upload photo of the food by clicking the camera icon	The photo is uploaded.		
4	Enter submit button	The page with sentence “Thanks you for your feedback” is displayed		

3.3.1

Test Case Name	3.3.1 Rating by giving valid star(s)	**Test Case Description**	To test the rating function by giving 1 to 5 star(s)
Pre-conditions:		**Test Data:**	
1	The user must already logged in the system	1	Rating = “2”
2		2	
3		3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Go to feedback section	The feedback of Menu Lee system is displayed.		
2	Click the star(s) based on experience	The star(s) that have been clicked turning grey colour to yellow colour.		
.	Enter submit button	The page with the sentence “Thank you for your rating and support” is displayed.		

4.1.1

Test Case Name	4.1.1 The system shall generate daily sales reports with a valid date.	Test Case Description	To validate that the system generates a daily sales report with accurate data for a given date.
Pre-conditions:		**Test Data:**	
1	User is logged in with access to sales reports.	1	Date: August 1, 2024
2	The system contains sales data for the date in question.	2	Total sales amount:RM699
3	The report generation feature is functioning correctly.	3	

Step #	Step Details	Expected Results	Actual Results	Remarks (Pass / Fail / Not executed / Suspended)
1	Choose the specific date for the sales report (e.g., "August 1, 2024")	System will generate a calendar that the user can choose for the sales reports.		
2	Click on the "Generate Report" button.	The system will display the sales report based on the date chosen.		

4.2.1

Test Case Name	4.2.1 The system shall track sales by individual items, showing which dishes are the most popular.	**Test Case Description**	Validate that the system tracks sales by individual items and accurately shows which dishes are the most popular based on sales data.
Pre-conditions:		**Test Data:**	
1	User is logged into the system.	1	Date:August 1,2024
2	There are sales transactions recorded in the system for various dishes.	2	Most Sales: Cold Dog
3	Sales data includes multiple items, with some items being sold more frequently than others.	3	Total amount of “Cold Dog” Sold: 38

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Navigate to the "Item sales" where individual item sales are displayed.	The system will display total sales of items.		
2	Choose a date(e.g., "August 1, 2024").	Users can choose a specific date based on the given calendar.		
3	Check the system highlights the most popular dishes based on the number of times they were sold.	The most popular dishes are correctly identified and displayed based on their sales count.		

4.3.1

Test Case Name	4.3.1 The system shall generate the ranking of the least popular food monthly.	**Test Case Description**	To validate that the system generates rankings of the least popular foods on a monthly basis, based on sales data.
Pre-conditions:		**Test Data:**	
1	User is logged into the system with access to sales reports.	1	Date: August-September
2	The system contains sales data for multiple food items over the course of at least one month.	2	Most sales: Fried Chicken Least Sales: Red bean Soup
3	The system includes functionality to rank food items by popularity.	3	Total amount of “Fried Chicken” sold: 336 Total amount of “Red Bean Soup” sold: 35

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Navigate to the "Sales Ranking" section.	The system will display the total amount of sales.		
2	Choose a full month for the report(e.g., "August 2024").	The system will let the user choose a specific month for the sales report.		
3	Click on the "Generate Report" button to create a report for the selected month.	The system will show the items are ranked from least popular (lowest sales) to more popular based on sales data.Items with zero sales are included in the ranking.		

5.1.1

Test Case Name	5.1.1 Create accounts in the system	**Test Case Description**	To test the account creation function using valid user data.
Pre-conditions:		**Test Data:**	
1	The user must not already have an account in the system.	1	Full Name: Abcd
2		2	Email: abcd@example.com
3		3	Password: *****
		4	Confirm Password: *****

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Go to www.MenuLee.com and navigate to the "Sign Up" page.	The "Create Account" page is displayed.		
2	Enter full name:Abcd.	Full name is entered.		
3	Enter email: abcd@example.com.	Email is entered.		
4	Enter password and confirm password.	Password and confirm password fields are filled.		
5	Click the "Create Account" button.	A confirmation message is shown, and the account is created. The user is redirected to the login page.		

5.2.1

Test Case Name	5.2.1 Login and logout with valid data	**Test Case Description**	To test the login function using valid username and password.
Pre-conditions:		**Test Data:**	
1	The user must already be registered to the system.	1	Email: abcd@example.com
2		2	Password: *****
3		3	

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Go to www.MenuLee.com	The Menu Lee system web page is displayed.		
2	Enter the username and password as specified in the user data	The username and password were entered.		
3.	Click the "Login" button	The menu homepage is displayed.		
4.	Click the "Logout" button to log out of the system.	The user is logged out.		

5.3.1

Test Case Name	5.3.1 notify the user of the progress of the food	**Test Case Description**	To verify that the system correctly notifies the user about the status of their food order from preparation to be served.
Pre-conditions:		**Test Data:**	
1	The user must have placed a food order in the system.	1	User Email: abcd@example.com
2	The system should have order tracking functionality	2	Order ID: ORD12345
3		3	Status updates: "Order Received", "Preparing", "Ready for Pickup", "Served at the Restaurant"

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Remarks (Pass / Fail / Not executed / Suspended)**
1	Place order at the system www.MenuLee.com	The order is placed, and an initial notification "Order Received" is sent.		
2	As the restaurant starts preparing the food, the user receives an update.	The user receives a notification: "Your food is now being prepared."		
3	When the food is ready for pickup, the user is notified.	The user receives a notification: "Your food is ready for pickup."		
4	Once the user picks up the food or it's served at the restaurant, the status updates to "Served".	The user receives a final notification: "Your food has been served at the restaurant."		

3.2 Software Configuration Management

Here is the link to the repository: https://github.com/Naslba/AMSE1003.git

Description:

Git is a powerful version control system that helps track changes and manage different versions of our work efficiently. By scanning for changes, we can stage the files we want to include in our next commit, allowing us to commit specific changes. If we realize there is something to amend after committing, we can modify the commit before pushing it to the repository. This helps maintain a clean and organized version history, ensuring we can revert to previous versions if needed. Additionally, Git makes it easy to manage branches and collaborate with others without overwriting each other's work, making it an essential tool for version control.

4. Reference section

Lecture notes:

- Bt Basri, Surayaini. (2024) Software Engineering [Lecture notes]. AMSE1003. Tunku Abdul Rahman University of Management and Technology.

Journal Articles:

- Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002). Agile software development methods: Review and analysis. *VTT Publications*, 478, pp.1-107.

Web Sources:

- Ordermentum Insights (2018) The Pitfalls of Manual Ordering in a Coffee Supply Chain. Available at: https://www.ordermentum.com/blog/the-pitfalls-of-manual-ordering-in-a-coffee-supply-chain (Accessed: 9 July 2024).
- Priyanka (2024). What Are The Software Quality Attributes? Available at: https://testsigma.com/blog/software-quality-attributes/ (Accessed: 16 July 2024)
- Agile Alliance (2024). *What is Extreme Programming (XP)?* Available at: https://www.agilealliance.org/glossary/xp (Accessed: 30 July 2024).
- ClickUp (2023) *What is XP in Agile?*. Available at: https://clickup.com/blog/what-is-xp-in-agile/ (Accessed: 30 July 2024).
- TeamGantt (n.d.) *How to use Gantt charts for your Agile project*. Available at: https://www.teamgantt.com/blog/how-to-use-gantt-charts-for-your-agile-project (Accessed: 6 August 2024).
- Chacon, S. and Straub, B. (2014) *Pro Git*. 2nd edn. Apress. Available at: https://git-scm.com/book/en/v2 (Accessed: 26 September 2024).

5. Appendices

Originality report:

Menu Lee: