



ASSESSMENT COVERSHEET

Attach this coversheet as the cover of your submission. All sections must be completed.

Section A: Submission Details

Programme : BACHELOR IN SOFTWARE ENGINEERING
Course Code & Name : ISB26504 SOFTWARE DESIGN AND INTEGRATION
Course Lecturer(s) : MADAM ROBIAH BINTI HAMZAH
Submission Title : MINI PROJECT REPORT – PIZZA ORDERING SYSTEM
Deadline : Day 7 Month FEB Year 2025 Time 6:00 PM
Penalties :

- 5% will be deducted per day to a maximum of four (4) working days, after which the submission will **not** be accepted.
- Plagiarised work is an Academic Offence in University Rules & Regulations and will be penalised accordingly.

Section B: Academic Integrity

Tick (✓) each box below if you agree:

<input checked="" type="checkbox"/>	I have read and understood the UniKL's policy on Plagiarism in University Rules & Regulations.
<input checked="" type="checkbox"/>	This submission is my own, unless indicated with proper referencing.
<input checked="" type="checkbox"/>	This submission has not been previously submitted or published.
<input checked="" type="checkbox"/>	This submission follows the requirements stated in the course.

Section C: Submission Receipt

(must be filled in manually)

Office Receipt of Submission

Date & Time of Submission (stamp)	Student Name(s)	Student ID(s)
7/2/2025 5.10 PM	NUR AFIKAH BINTI NOOR AZMAN NOR MIRZA AFIQAH BINTI ABDUL RAHMAN NUR NISA RAFIAH BINTI RAMLEE NUR HAMIZAH ZAHIRAH BINTI NOR'AZMAN NUR SAZAHAH BINTI SALAUDDIN	52213223167 52213223174 52213223209 52213223118 52213223035

Student Receipt of Submission

This is your submission receipt, the only accepted evidence that you have submitted your work. After this is stamped by the appointed staff & filled in, cut along the dotted lines above & retain this for your record.

Date & Time of Submission (stamp)	Course Code	Submission Title	Student ID(s) & Signature(s)
7/2/2025 5.10 PM	ISB26504	MINI PROJECT REPORT – PIZZA ORDERING SYSTEM	52213223167 AFIKAH

Table of Contents

1. INTRODUCTION	4
2. REQUIREMENT ANALYSIS.....	4
2.1 Background Of the Project.....	4
2.2 Aim of the Project	5
2.3 Scope – New Version.....	5
2.4 Technology Used	6
3. DESIGN	7
3.1 Use Case Diagram	7
3.2 Use Case Specification	8
3.3 System Architecture.....	50
4. IMPLEMENTATION	51
4.1 User Manual of Delicious Pizza System	51
4.2 Technical Specifications.....	57
4.3 Explanations For Every Function and Module	60
4.4 Discussions	64
5. TESTING.....	77
5.1 Table of Input and Output Process	77
6. MAINTENANCE.....	81
6.1 Discussion	81
6.2 Benefits of the Project	81
6.3 Limitations of the Project.....	82
6.4 Recommendations	83
6.5 Conclusion	83
7. REFERENCE	84
8. APPENDIX.....	85
8.1 Minutes of Meeting 1	85
8.2 Minutes of Meeting 2	86

Table of Figures

Figure 1. Use Case Diagram	7
Figure 2. Deli Pizza System Architecture.....	50
Figure 3. Delicious Pizza Homepage	51
Figure 4. Sign up page.....	51
Figure 5. Login page.....	52
Figure 6. Menu Selection page	52
Figure 7. Customize Selection page.....	53
Figure 8. User Cart page	54
Figure 9. Checkout page	54
Figure 10. Receipt page	55
Figure 11. User Profile page.....	55
Figure 12. My Rewards page.....	56
Figure 13. Feedback and Contact page	57
Figure 14. Sign up Input Error Validation	57
Figure 15. Exist User Account Validation	58
Figure 16. Login Input Error Validation.....	58
Figure 17. Incorrect Password Validation.....	59
Figure 18. Invalid User Account Validation	59
Figure 19. Sign up section	65
Figure 20. Sign up Validation Check.....	66
Figure 21. Login section	67
Figure 22. Login Validation	67
Figure 23. Menu Selection.....	68
Figure 24. Promotion Notification	71
Figure 25. Header page.....	76
Figure 26. Footer of the page.....	76

1. INTRODUCTION

The Delicious Pizza System is a comprehensive digital platform that simplifies the online pizza ordering experience for users. At its core, the system offers a user-friendly interface that allows customers to connect with the pizza business via a variety of features, beginning with a simple sign-up and login process. Users can use intuitive interfaces to traverse menu selections, select pizza variations, personalize orders, and manage their personal information.

The system's architecture enables a complete end-to-end ordering workflow, including essential features like cart management, delivery mode selection, and payment processing. Customers may check their order history, update their profiles, and even provide feedback, resulting in a comprehensive digital experience that extends beyond transaction processing. The platform's design promotes flexibility by allowing customers to add or remove goods from their cart, choose preferred delivery methods, and enter specific delivery addresses.

Administratively, the system has strong management tools that allow for backend control and monitoring. Administrators can log in to view user information, track sales, and ensure system integrity. The use case diagram demonstrates a comprehensive approach to digital service delivery that balances user convenience and operational efficiency. The Delicious Pizza System stands out as a modern, user-centric solution for online food ordering, thanks to features such as input validation, order placement, and receipt generation.

2. REQUIREMENT ANALYSIS

2.1 Background Of the Project

The Delicious Pizza Ordering System was initially a web-based application using C# as its main language. Users could browse the menu, choose pizzas, and place orders with this version's minimal capabilities. However, modern online ordering systems require characteristics like validation input, smooth ordering process and online transactions, as well as additional features like user profile, points system and feedback, which the current version lacked. To implement these improvements, the project was transitioned to PHP to create an even better, more robust version of the system. The PHP version will significantly improve the user experience, offering a more responsive design and dynamic content updates.

2.2 Aim of the Project

This project aims to enhance the existing Pizza Ordering System by addressing current limitations, improving user navigation, integrating essential features like menu selection, checkout, payment, and flexible delivery options, implementing new features like user profile, rewards and points redemption as well as optimizing the system for both customer satisfaction and operational efficiency.

2.3 Scope – New Version

Module	Description
Homepage	This module displays an introduction to the system with menus and next-page buttons. It also includes navigation bar for key functionalities like sign-up and login, along with a drop-down menu for quick access to various categories.
Sign Up	This module allows users to register for the system by entering personal details such as a username, email, phone number, address and password.
Sign Up (Error Messages)	Shows error messages if the required fields are left empty or the input is invalid.
Login	Provides secure authentication by requiring users to enter their username and password to access their accounts.
Login (Error Message)	Displays error messages for incorrect passwords or if the account is disabled, like the error handling in the Sign-Up module.
Add to Cart	Enables users to place orders by adding items to their cart and confirming sales. Users can choose their preferred crust and quantity before adding to cart. Users can review their cart's contents before finalizing their purchase.
Checkout (Review payment, Receipt)	Enables user to check their total price, delivery/pickup selection and rewards redemption before processing to place order. Users will receive receipt that is printable.
Points Redemption (My Rewards)	Users will receive points for every purchase and the accumulated points is available in their rewards page. Users can use the points to redeem rewards during checkout process.

Admin (Admin, Edit Users)	Allows administrators to view a monthly sales summary, sales by product, and total sales. Additionally, it lets administrators can see list of users registered..
User Profile	Enable user to view their information along with their purchased history for easy tracking.
Contact & Feedback	Enable user to leave their feedback or message to the customer service. The page also displays the contact information for users to contact the shop for further assistance.

2.4 Technology Used

Technology	Descriptions
PHP	Used to develop the server-side logic of the Pizza Ordering System, handling functionalities like order processing, database interaction, and user authentication.
HTML	Provides the structure for web pages, including sections like the menu display, customer registration forms, and the checkout page. This ensures a well-organized layout for the web application
CSS	Styles the web pages, ensuring they are visually appealing and consistent. It manages layout, colours, fonts, and responsive design to ensure the site works well on different devices.
SQL Server	A robust database management system used to store and manage structured data, such as menu items, customer orders, and payment details. It supports fast and reliable data retrieval and updates.
Visual Studio Code	The integrated development environment (IDE) used for writing, testing, and debugging the Pizza Ordering System's codebase.
Laragon / Xampp	A software packages used to run web applications locally on computers. It included a variety of components like Apache web server, MySQL database and PHP scripting language.

3. DESIGN

3.1 Use Case Diagram

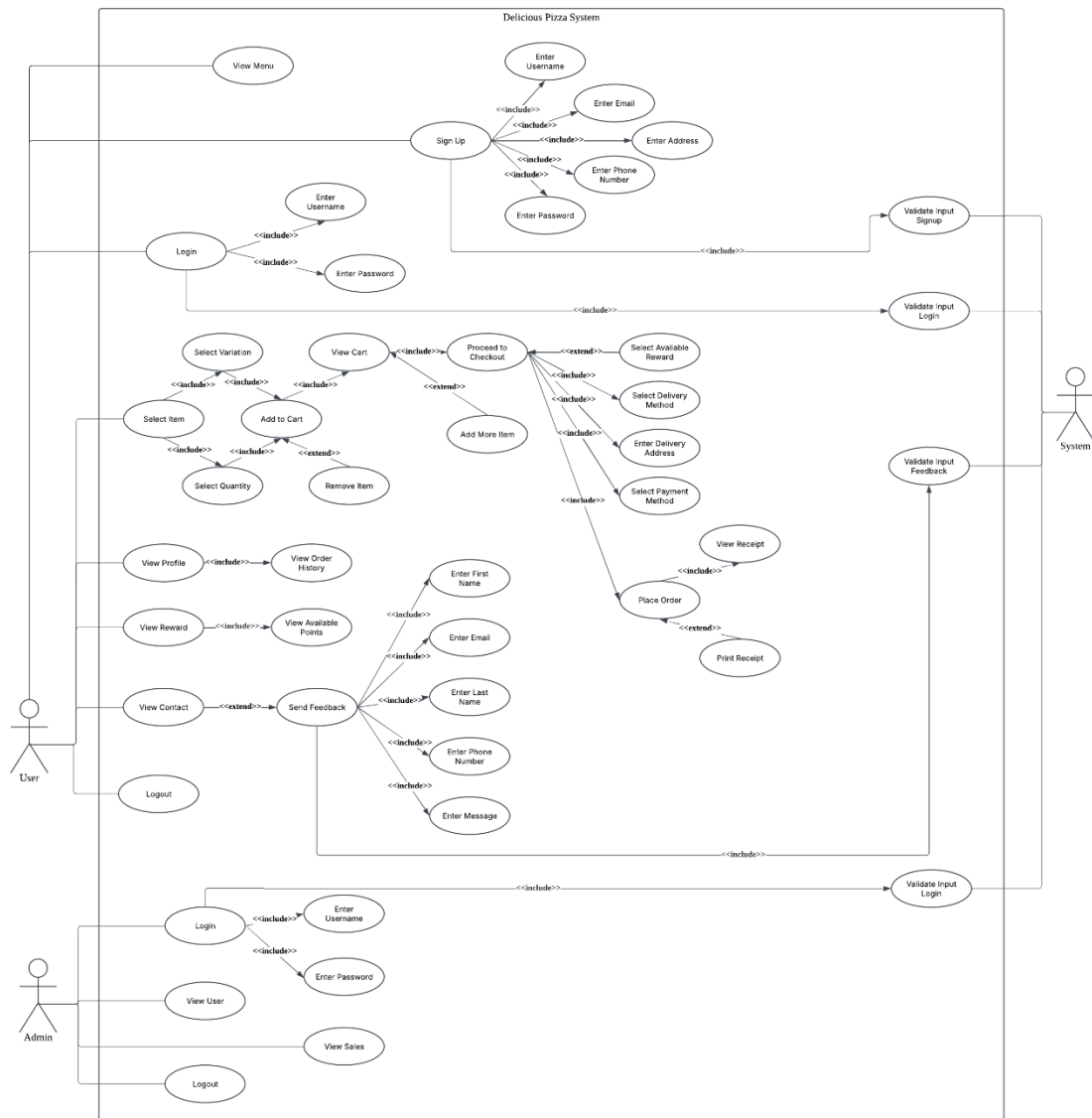


Figure 1. Use Case Diagram

https://lucid.app/lucidchart/a090440a-5c48-4c3c-992c-50e6dcb296f8/edit?viewport_loc=-469%2C-362%2C5097%2C2275%2C0_0&invitationId=inv_5e571af3-7d72-43ce-b275-803f7df301f7

3.2 Use Case Specification

Section	Content
Designation	UC-01-01
Name	View Menu
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for system usability.
Criticality	Medium - Important for accessing available food items.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view the restaurant's menu.
Trigger event	User clicks the "Menu" button on the homepage.
Actors	User
Pre-condition	<ul style="list-style-type: none">- The user is on the homepage.- The system is functional and able to retrieve menu data.
Post-condition	<ul style="list-style-type: none">- The system displays the menu page with the available food items.
Result	The user can see the list of food items along with their details (e.g., name, price, description, image).
Main scenario	<ol style="list-style-type: none">1. The user navigates to the homepage.2. The system displays the "View Menu" button.3. The user clicks the "View Menu" button.4. The system retrieves menu data and displays the menu page.
Alternative scenario	<ul style="list-style-type: none">- If the menu data fails to load, the system displays a message indicating that the menu is unavailable and prompts the user to retry.
Exception scenario	<ul style="list-style-type: none">- If the system encounters an error retrieving the menu, an error message is displayed.

Section	Content
Designation	UC-01-02
Name	Sign Up

Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for user account creation.
Criticality	Medium - Required for personalized user experience and order management.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows new users to create an account.
Trigger event	User clicks the "Sign Up" button on the homepage navigation bar.
Actors	User
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - A new user account is successfully created and stored in the system.
Result	A new user account is successfully created and stored in the system.
Main scenario	<ol style="list-style-type: none"> 1. The user clicks the "Sign Up" button. 2. The system displays the sign-up form. 3. The user enters the required details (username, email, phone number, address, password). 4. The user submits the form. 5. The system validates the input data. 6. If valid, the system creates a new user account and stores the information in the database. 7. The system displays a success message and redirects the user to the login page.
Alternative scenario	<ul style="list-style-type: none"> - If the user already has an account, they are redirected to the login page.
Exception scenario	<ul style="list-style-type: none"> - If any required field is empty, the system prompts the user to fill it. - If the email or phone number is already registered, an error message is displayed. - If the password is too short, the system requests a longer password, at least 8 characters.

Section	Content
Designation	UC-01-02a
Name	Sign Up (Username)
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for user identification.
Criticality	Medium - Required for account creation and login.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Defines the constraints for the username field during the sign-up process.
Trigger event	The user enters a username during sign-up.
Actors	<ul style="list-style-type: none"> - User (provides input for the username). - System (validates and checks if the username is available).
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the username meets all requirements. - If valid, the username is stored in the database for the new user account.
Result	The user successfully registers with a valid and unique username.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the sign-up page. 2. The user enters a username. 3. The system checks if the username is already taken. 4. If the username meets all requirements, the system accepts it. 5. The user proceeds to enter other sign-up details.
Alternative scenario	<ul style="list-style-type: none"> - If the username is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If the username is already registered, the system displays an error message: " Username is already in use. Please use a different username."

	<ul style="list-style-type: none"> - If there is a system error during validation, an error message is displayed, and the user is asked to try again later.
--	--

Section	Content
Designation	UC-01-02b
Name	Sign Up (Email)
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for user identification.
Criticality	Medium - Required for account creation and login.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Defines the constraints for the email field during the sign-up process.
Trigger event	The user enters an email during sign-up.
Actors	<ul style="list-style-type: none"> - User (provides input for the email). - System (validates and checks if the email is available).
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the email meets all requirements. - If valid, the email is stored in the database for the new user account.
Result	The user successfully registers with a valid and unique email.
Main scenario	<ol style="list-style-type: none"> 6. The user navigates to the sign-up page. 7. The user enters an email. 8. The system checks if the email is already taken. 9. If the email meets all requirements, the system accepts it. 10. The user proceeds to enter other sign-up details.
Alternative scenario	<ul style="list-style-type: none"> - If the email is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If the email is already registered, the system displays an error message: "Email is already in use. Please use a different email."

	<ul style="list-style-type: none"> - If there is a system error during validation, an error message is displayed, and the user is asked to try again later.
--	--

Section	Content
Designation	UC-01-02c
Name	Sign Up (Phone Number)
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for user identification.
Criticality	Medium - Required for account creation.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Defines the constraints for the phone number field during the sign-up process.
Trigger event	The user enters a phone number during sign-up.
Actors	<ul style="list-style-type: none"> - User (provides input for the phone number). - System (validates and checks if the phone number is available).
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the phone number meets all requirements. - If valid, the phone number is stored in the database for the new user account.
Result	The user successfully registers with a valid and unique phone number.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the sign-up page. 2. The user enters a phone number. 3. The system checks if the phone number is already in use. 4. If valid, the system accepts the phone number. 5. The user proceeds to enter other sign-up details.
Alternative scenario	<ul style="list-style-type: none"> - If the phone number is left blank, the system prompts the user to enter one.

Exception scenario	<ul style="list-style-type: none"> - If the phone number is already registered, the system displays an error message: "Phone number is already in use. Please use a different phone number." - If there is a system error during validation, an error message is displayed, and the user is asked to try again later.
--------------------	---

Section	Content
Designation	UC-01-02d
Name	Sign Up (Address)
Authors	Nor Mirza Afiah
Priority	Importance for system success: medium - Required for delivery
Criticality	Medium - Important for delivery accuracy and order processing.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Defines the constraints for the address field during the sign-up process.
Trigger event	The user enters an address during sign-up.
Actors	<ul style="list-style-type: none"> - User (provides input for the address). - System (validates the address).
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the address meets all requirements. - If valid, the address is stored in the database for the new user account.
Result	The user successfully registers with a valid address.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the sign-up page. 2. The user enters their address. 3. The user proceeds to enter other sign-up details.
Alternative scenario	<ul style="list-style-type: none"> - If the address is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If there is a system error during validation, an error message is displayed, and the user is asked to try again later.

Section	Content
Designation	UC-01-02e
Name	Sign Up (Password)
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for user account security
Criticality	High - Required to ensure account protection and creation.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Defines the constraints for the password field during the sign-up process.
Trigger event	The user enters a password during sign-up.
Actors	<ul style="list-style-type: none"> - User (provides input for the password). - System (validates the password).
Pre-condition	<ul style="list-style-type: none"> - The user is on the sign-up page. - The system is functional and ready to accept user details.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the password meets all security requirements. - If valid, the password is encrypted and stored in the database.
Result	The user successfully registers with a secure password.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the sign-up page. 2. The user enters a password. 3. The system validates the password length. 4. If the password meets all requirements, the system accepts it.
Alternative scenario	<ul style="list-style-type: none"> - If the password is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If the password is too short, the system displays an error message: "Password must be at least 8 characters long." - If there is a system error during validation, an error message is displayed, and the user is asked to try again later.

Section	Content
Designation	UC-01-03
Name	Log In
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for user authentication.
Criticality	High - Required for account access and personalized features.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows registered users to log in to their accounts using their username or email and password.
Trigger event	The user clicks the "Login" button on the homepage or after signing up.
Actors	<ul style="list-style-type: none"> - User (provides login credentials). - System (validates credentials and grants access).
Pre-condition	<ul style="list-style-type: none"> - The user has already registered an account. - The user is on the login page. - The system is functional and can validate login credentials.
Post-condition	<ul style="list-style-type: none"> - If login is successful, the user will be granted access to their account. - If login fails, the system prompts the user to re-enter valid credentials.
Result	The user successfully logs in and gains access to their profile and features.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the login page. 2. The system displays input fields for username/email and password. 3. The user enters their username or email and password. 4. The user clicks the "Login" button. 5. The system verifies the credentials. 6. If the credentials are valid, the user is granted access and redirected to the homepage or dashboard.
Alternative scenario	<ul style="list-style-type: none"> - If the user doesn't have an account, they will need to sign up first.

Exception scenario	<ul style="list-style-type: none"> - If the username/email does not exist, the system displays an error message: "Email/username not registered. Please sign up first." - If the password is incorrect, the system rejects it and displays an error message: "Incorrect password. Please try again."
--------------------	--

Section	Content
Designation	UC-01-03a
Name	Log In (Username)
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for identifying the user and granting access.
Criticality	High - Required for successful login.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows registered users to log in to their accounts using their password.
Trigger event	The user enters a username during the login process.
Actors	<ul style="list-style-type: none"> - User (provides username for login). - System (validates and checks the entered username).
Pre-condition	<ul style="list-style-type: none"> - The user is on the login page. - The system is functional and able to validate user credentials.
Post-condition	<ul style="list-style-type: none"> - If login is successful, the user will be granted access to their account. - If login fails, the system prompts the user to re-enter valid credentials.
Result	The user successfully logs in if the username exists and is linked to a valid password.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the login page. 2. The user enters their username.

	3. If the username is valid, the user proceeds to enter their password.
Alternative scenario	- If the username is left blank, the system prompts the user to enter one.
Exception scenario	- If the username, the system displays an error message: "Username not registered. Please sign up first."

Section	Content
Designation	UC-01-03b
Name	Log In (Password)
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for identifying the user and granting access.
Criticality	High - Required for successful login.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows registered users to log in to their accounts using their password.
Trigger event	The user enters a username during the login process.
Actors	<ul style="list-style-type: none"> - User (provides password for login). - System (validates and checks the entered password).
Pre-condition	<ul style="list-style-type: none"> - The user is on the login page. - The system is functional and able to validate user credentials.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the entered password matches the one stored in the database for the given username. - If the password is valid, the system grants the user access to their account.
Result	The user successfully logs in if the provided password matches the stored password.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the login page. 2. The user enters their password.

	<ol style="list-style-type: none"> 3. The system verifies that the entered password matches the one associated with the username. 4. If the password is correct, the system grants access and redirects the user to their account page.
Alternative scenario	<ul style="list-style-type: none"> - If the password is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If the password does not match the one stored in the system, the system displays an error message: "Incorrect password. Please try again."

Section	Content
Designation	UC-01-03b
Name	Log In (Password)
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for identifying the user and granting access.
Criticality	High - Required for successful login.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows registered users to log in to their accounts using their password.
Trigger event	The user enters a username during the login process.
Actors	<ul style="list-style-type: none"> - User (provides password for login). - System (validates and checks the entered password).
Pre-condition	<ul style="list-style-type: none"> - The user is on the login page. - The system is functional and able to validate user credentials.
Post-condition	<ul style="list-style-type: none"> - The system verifies that the entered password matches the one stored in the database for the given username. - If the password is valid, the system grants the user access to their account.
Result	The user successfully logs in if the provided password matches the stored password.

Main scenario	<ol style="list-style-type: none"> 5. The user navigates to the login page. 6. The user enters their password. 7. The system verifies that the entered password matches the one associated with the username. 8. If the password is correct, the system grants access and redirects the user to their account page.
Alternative scenario	<ul style="list-style-type: none"> - If the password is left blank, the system prompts the user to enter one.
Exception scenario	<ul style="list-style-type: none"> - If the password does not match the one stored in the system, the system displays an error message: "Incorrect password. Please try again."

Section	Content
Designation	UC-01-04
Name	Select Item
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for selecting and adding items to the cart.
Criticality	High - Required for users to view food descriptions and customize their order before adding to the cart.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows users to view the details of a selected food item, including its variations, and choose the desired quantity before adding the item to the cart.
Trigger event	The user clicks on the image of a food item after logging in.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user is logged in and, on the homepage, or food menu page. - The system has a list of food items with images and related details.
Post-condition	<ul style="list-style-type: none"> - The system displays the selected food item's description along with available variations.

	<ul style="list-style-type: none"> - The user selects the desired variation and enters the quantity. - The system updates the cart with the chosen food item and quantity.
Result	The user successfully selects a food item with the desired variations and quantity and adds it to the cart for checkout.
Main scenario	<ol style="list-style-type: none"> 1. The user logs in and views the available food items on the homepage or menu. 2. The user clicks on the “Select” of a food item to view its details. 3. The system displays the food description, available variations, and a quantity field. 4. The user selects the variation(s) and enters the quantity they wish to order. 5. The user clicks the "Add to Cart" button. 6. The system adds the item to the cart with the selected variations and quantity.
Alternative scenario	<ul style="list-style-type: none"> - The user can just add to cart without making any changes for the variations as the system is already set to basic variation.
Exception scenario	<ul style="list-style-type: none"> - If there is a system error while processing the selection or adding to the cart, an error message is displayed, and the user is asked to try again later.

Section	Content
Designation	UC-01-04a
Name	Select Variation
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for the user to customize their pizza order.
Criticality	High - Required for the user to choose the pizza size before adding to the cart.
Source	Nor Mirza Afiah

Responsible	Nor Mirza Afiah
Description	Allows users to select a pizza size and crust variation as part of the customization before adding the item to the cart.
Trigger event	The user clicks on the size options for the pizza item.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has selected the pizza item and is viewing its customization options. - The system is functional and ready to process the variation (size and crust) selection.
Post-condition	<ul style="list-style-type: none"> - The system registers the selected pizza size and crust for the item. - The selected variation is reflected in the cart along with the other customization options.
Result	The user successfully selects a pizza size and crust variation for their order.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the pizza item's details page. 2. The system displays the available size and crust options. 3. The user selects one of the size and crust options. 4. The user continues to customize or adds the item to the cart.
Alternative scenario	<ul style="list-style-type: none"> - If the user does not select a pizza size, the system may default to regular. - If the user does not select a quantity, the system defaults to 1.
Exception scenario	<ul style="list-style-type: none"> - If the system encounters an error while processing the quantity, the user is prompted to try again later.

Section	Content
Designation	UC-01-04b
Name	Select Quantity
Authors	Nor Mirza Afiah

Priority	Importance for system success: high - Essential for defining the number of items in an order
Criticality	High - Required to ensure the user selects the desired quantity before adding to the cart.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows users to specify the number of units they want to purchase for a selected food item before adding it to the cart.
Trigger event	The user interacts with the quantity selector (plus/minus buttons).
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has selected a food item and is on the item's details page. - The system is functional and able to process quantity selection.
Post-condition	<ul style="list-style-type: none"> - The system registers the selected quantity and updates the item details accordingly. - The total price is adjusted based on the selected quantity.
Result	The user successfully selects a quantity for the chosen food item, which is then added to the cart.
Main scenario	<ol style="list-style-type: none"> 5. The user navigates to the selected food item's description page. 6. The user selects the desired quantity. 7. The system updates the total price based on the chosen quantity. 8. The user proceeds to add the item to the cart.
Alternative scenario	<ul style="list-style-type: none"> - If the user does not select a quantity, the system defaults to 1.
Exception scenario	<ul style="list-style-type: none"> - If the system encounters an error while processing the quantity, the user is prompted to try again later.

Section	Content
Designation	UC-01-04c
Name	Add to Cart

Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for completing the ordering process.
Criticality	High - Required for users to store selected food items before proceeding to checkout.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to add a selected food item, along with its variations and quantity, to the shopping cart for future checkout.
Trigger event	The user clicks the "Add to Cart" button after selecting an item, variation, and quantity.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has selected an item, its variation (if applicable), and quantity. - The system is functional and able to process cart updates.
Post-condition	<ul style="list-style-type: none"> - The system adds the selected item to the cart with the specified variation and quantity. - The cart is updated to reflect the added item and the total price is adjusted.
Result	The user successfully adds an item to the cart, making it available for checkout.
Main scenario	<ol style="list-style-type: none"> 1. The user selects a food item. 2. The user selects the variation (if applicable) and quantity. 3. The user clicks the "Add to Cart" button. 4. The system updates the cart. 5. The user can view the cart and continue shopping or proceed to checkout.
Alternative scenario	<ul style="list-style-type: none"> - If the user does not select a quantity, the system defaults to 1.

	<ul style="list-style-type: none"> - If the user does not select variety of the sizes and crust, the system default to regular(size) and medium(crust).
Exception scenario	<ul style="list-style-type: none"> - If the system encounters an error while processing the quantity, the user is prompted to try again later.

Section	Content
Designation	UC-01-04d
Name	View Cart
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for users to review their selected items before checkout
Criticality	High - Required for users to verify order details and make modifications before proceeding.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view the list of food items they have added to the cart, along with their selected variations, quantity, and total price.
Trigger event	The user clicks the "Cart" icon.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has added at least one item to the cart. - The system is functional and can retrieve the cart details.
Post-condition	<ul style="list-style-type: none"> - The system displays the cart contents, including selected items, variations, quantity, and total price. - The user can review the cart and make modifications if needed.
Result	The user successfully views the cart and verifies order details.
Main scenario	<ol style="list-style-type: none"> 1. The user clicks the "Cart" logo button. 2. The system shows the food item name, selected variation, quantity, and total price.

	3. The user reviews the cart and ensures the order details are correct.
Alternative scenario	<ul style="list-style-type: none"> - If the user wants to modify an item, they can click on the item to adjust the quantity or variation. - If the user wants to remove an item, they can click a "Remove" button.
Exception scenario	<ul style="list-style-type: none"> - If the cart is empty, the system displays a message: "Your cart is empty."

Section	Content
Designation	UC-01-04e
Name	Remove Item
Authors	Nor Mirza Afiah
Priority	Importance for system success: high - Essential for modifying the cart before checkout.
Criticality	High - Ensures users can remove unwanted items before placing an order.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows users to remove an unwanted food item from the shopping cart before proceeding to checkout.
Trigger event	The user clicks the "Remove" or "Delete" button next to an item in the cart.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has at least one item in the cart. - The system is functional and can update the cart.
Post-condition	<ul style="list-style-type: none"> - The selected item is removed from the cart. - The cart total is updated accordingly
Result	The user successfully removes an item from the cart, and the updated cart is displayed.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the cart page. 2. The user clicks the "Remove" button for an item they no longer want.

Alternative scenario	<ul style="list-style-type: none"> - If the user accidentally removes an item, they can re-add it by returning to the menu. - If the cart becomes empty after removal, the system displays a message: "Your cart is empty."
Exception scenario	<ul style="list-style-type: none"> - If the system fails to remove the item, an error message is displayed.

Section	Content
Designation	UC-01-04f
Name	Add More Item
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for users to continue shopping and add more items to the cart.
Criticality	High - R Important for user experience but does not affect the checkout process directly.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to return to the menu or homepage to select and add more food items to their cart
Trigger event	The user clicks the "Add More Item" button from the cart page.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has at least one item in the cart. - The system is functional and can navigate back to the menu or homepage.
Post-condition	<ul style="list-style-type: none"> - The system redirects the user to the menu or homepage. - The user can select additional items.
Result	The user successfully returns to the menu and can add more items to the cart.
Main scenario	<ol style="list-style-type: none"> 1. The user clicks the "Add More Item" button on the cart page. 2. The system redirects the user to the menu. 3. The user browses the available food items.

	4. The user selects additional items and adds them to the cart.
Alternative scenario	- If the user decides not to add more items, they can return to the cart manually.
Exception scenario	- If the system fails to redirect the user to the menu or homepage, an error message is displayed: "Unable to load menu. Please try again later."

Section	Content
Designation	UC-01-04g
Name	Proceed to Checkout
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for completing the order process.
Criticality	High - Required for finalizing the order and proceeding with payment.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to proceed from the cart to the checkout page, where they can review order details, enter payment and delivery information, and confirm their purchase.
Trigger event	The user clicks the "Proceed to Checkout" button on the cart page.
Actors	- User
Pre-condition	- The user has at least one item in the cart. - The system is functional and able to retrieve cart details.
Post-condition	- The user is redirected to the checkout page. - The cart details are displayed for final review.
Result	The user successfully moves to the checkout page to complete the purchase.
Main scenario	3. The user clicks the "Proceed to Checkout" button.

	<ol style="list-style-type: none"> 4. The user reviews the order summary, including item details, total cost, and delivery information. 5. The user proceeds to enter payment details and confirm the order.
Alternative scenario	<ul style="list-style-type: none"> - If the user wants to modify the cart, they can just adjust on the cart page before completing checkout.
Exception scenario	<ul style="list-style-type: none"> - If the cart is empty, the system displays an error message: "Your cart is empty."

Section	Content
Designation	UC-01-04h
Name	Proceed to Checkout
Authors	Nor Mirza Afiqah
Priority	Importance for system success: medium - Enhances user experience by allowing redemption of rewards
Criticality	Medium - Affects discount application but does not block checkout if unavailable
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to redeem an available reward based on their accumulated points to get a discount on their total order price.
Trigger event	The user proceeds to checkout and chooses to redeem a reward before finalizing payment.
Actors	<ul style="list-style-type: none"> - User - System
Pre-condition	<ul style="list-style-type: none"> - The user has accumulated enough reward points. - The system has a list of available rewards and their respective point requirements.
Post-condition	<ul style="list-style-type: none"> - The selected reward is applied to the order total. - The user's reward points are deducted accordingly.
Result	The user successfully redeems a reward, and the order total is updated with the discount.

Main scenario	<ol style="list-style-type: none"> 1. The user reaches the checkout page. 2. The user selects a reward: <ul style="list-style-type: none"> - RM5 Off (500 points) - RM10 Off (1000 points) - RM20 Off (2000 points) 3. The system verifies if the user has enough points for the selected reward. 4. The system applies the discount and updates the total price. 5. The system deducts the corresponding reward points from the user's balance. 6. The updated total price is displayed, and the user proceeds with payment.
Alternative scenario	<ul style="list-style-type: none"> - If the user decides not to use a reward, they can proceed without applying any discount.
Exception scenario	<ul style="list-style-type: none"> - If the user does not have enough points for a reward, the button for reward remains disabled.

Section	Content
Designation	UC-01-04i
Name	Select Delivery Method
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Essential for order fulfilment
Criticality	High - Determines how the user will receive their order.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to choose between Delivery and Pickup as their preferred method of receiving their order.
Trigger event	The user proceeds to checkout and reaches the delivery method selection section.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has at least one item in the cart.

	<ul style="list-style-type: none"> - The system is functional and can process delivery and pickup options.
Post-condition	<ul style="list-style-type: none"> - The selected delivery method is saved and applied to the order.
Result	The user successfully selects a delivery method, and the system updates the order details accordingly.
Main scenario	<ol style="list-style-type: none"> 1. The user reaches the checkout page. 2. The system displays the "Delivery Method" section with two options: <ul style="list-style-type: none"> - Delivery - Pickup 3. The user selects either Delivery or Pickup. 4. The user proceeds to the next step of checkout
Alternative scenario	<ul style="list-style-type: none"> - If the user changes their selection, the system updates the order details accordingly.
Exception scenario	<ul style="list-style-type: none"> - If the system fails to save the selection, error message will appear.

Section	Content
Designation	UC-01-04j
Name	Enter Delivery Address
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high - Required for successful order delivery.
Criticality	High - Ensures the order is sent to the correct location.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to enter their delivery address when selecting the Delivery option at checkout.
Trigger event	The user selects Delivery as the delivery method.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has selected Delivery as the delivery method.

	<ul style="list-style-type: none"> - The system is functional and capable of processing address details.
Post-condition	<ul style="list-style-type: none"> - The delivery address is saved and linked to the order.
Result	The user successfully selects a delivery method, and the system updates the order details accordingly.
Main scenario	<ol style="list-style-type: none"> 1. The user selects Delivery as the delivery method. 2. The system displays an address input form with field. 3. The user fills in the required field. 4. If valid, the system saves the address and updates the order summary. 5. The user proceeds to the next step of checkout.
Alternative scenario	<ul style="list-style-type: none"> - If the user choose pickup for delivery method, the system will not display requirement field.
Exception scenario	<ul style="list-style-type: none"> - If required field are missing, the system prevents checkout and displays error message.

Section	Content
Designation	UC-01-04k
Name	Select Payment Method
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high – Required for order completion.
Criticality	High - Ensures the user can proceed with payment successfully.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to select their preferred payment method before completing the order.
Trigger event	The user reaches the payment section after entering the delivery details.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user has at least one item in the cart.

	<ul style="list-style-type: none"> - The system is functional and capable of processing payments.
Post-condition	<ul style="list-style-type: none"> - The selected payment method is saved and linked to the order.
Result	The user successfully selects a payment method, and the system updates the order details accordingly.
Main scenario	<ol style="list-style-type: none"> 1. The user proceeds to the payment section on the checkout page. 2. The system displays available payment methods: <ul style="list-style-type: none"> - Credit Card - Online Banking - Cash on Delivery 3. The user selects their preferred payment method. 4. The user proceeds to the final order confirmation step.
Alternative scenario	<ul style="list-style-type: none"> - If the user changes their selection before confirming the order, the system updates the order details accordingly.
Exception scenario	<ul style="list-style-type: none"> - If the system fails to process the payment, an error message will appear.

Section	Content
Designation	UC-01-04I
Name	Place Order
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high – Required to complete the order process.
Criticality	High - Ensures the user successfully submits their order.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to finalize their order by submitting it for processing after selecting the payment method.
Trigger event	The user has selected a payment method and clicks the "Place Order" button.

Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has completed all required checkout steps. - The system is functional and able to process orders.
Post-condition	<ul style="list-style-type: none"> - The order is successfully submitted and recorded in the system. - The user is redirected to the order confirmation page.
Result	The system successfully processes the order, and the user receives a confirmation message.
Main scenario	<ol style="list-style-type: none"> 1. The user reviews the order summary and confirms all details are correct. 2. The user clicks the "Place Order" button. 3. The system validates the order details. 4. The system displays an order confirmation message with order details. 5. The user is redirected to the Order Confirmation page, where they can view or print receipt.
Alternative scenario	<ul style="list-style-type: none"> - If the user wants to modify their order before placing it, they can go back and make changes.
Exception scenario	<ul style="list-style-type: none"> - If the system encounters an issue saving the order, error message will appear.

Section	Content
Designation	UC-01-04m
Name	View Receipt
Authors	Nor Mirza Afiqah
Priority	Importance for system success: high – Required for order confirmation and reference.
Criticality	Medium - Allows users to review their order details after purchase.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view the receipt of their placed order, including order details, total amount, and payment status.

Trigger event	The user successfully places an order and is redirected to the order confirmation page.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has successfully placed an order. - The system has recorded the order details in the database.
Post-condition	<ul style="list-style-type: none"> - The receipt is displayed on the screen with complete order details.
Result	The user can view all order details, including itemized pricing, payment method, and delivery information.
Main scenario	<ol style="list-style-type: none"> 1. The user successfully places an order. 2. The system generates an electronic receipt containing: <ol style="list-style-type: none"> a. Order ID b. Order date c. Username d. Delivery Method e. Delivery Address f. Payment Method g. List of ordered items h. Subtotal, discounts (if any), and total amount 3. The user can review the order details. 4. The user has options to: <ol style="list-style-type: none"> a. Print the receipt b. Go back to the homepage
Alternative scenario	<ul style="list-style-type: none"> - If the user needs to access the receipt later, they can find it under Order History in their profile page.
Exception scenario	<ul style="list-style-type: none"> - If the receipt fails to load, error message will appear.

Section	Content
Designation	UC-01-04m
Name	Print Receipt
Authors	Nor Mirza Afiqah

Priority	Importance for system success: medium - Optional but useful for record-keeping.
Criticality	Low - Does not affect order processing but provides convenience.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to print a copy of their order receipt for reference.
Trigger event	The user selects the "Print Receipt" option after placing an order.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has successfully placed an order. - The system has generated and displayed the receipt.
Post-condition	<ul style="list-style-type: none"> - The receipt is printed successfully, or the print dialog is displayed.
Result	The user obtains a printed copy of the receipt.
Main scenario	<ol style="list-style-type: none"> 1. The user successfully places an order and views the receipt. 2. The user clicks "Print Receipt". 3. The browser or device opens the print dialog. 4. The user selects a printer and confirms the print action. 5. The receipt is printed successfully.
Alternative scenario	<ul style="list-style-type: none"> - The user saves the receipt as a file instead of printing.
Exception scenario	<ul style="list-style-type: none"> - If the printer is unavailable or there is a printing issue, error message will appear.

Section	Content
Designation	UC-01-05
Name	View Profile
Authors	Nor Mirza Afiqah
Priority	Importance for system success: medium - Allows users to manage their personal details

Criticality	Low - Does not impact order processing but enhances user experience.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view their personal information, including profile picture, username, email, phone number, and address.
Trigger event	The user clicks the profile logo button from the navigation menu.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user is logged in. - The system has stored the user's profile details in the database.
Post-condition	<ul style="list-style-type: none"> - The user's profile information is displayed on the screen.
Result	The user can see their personal details, including profile image, name, email, phone number, and address.
Main scenario	<ol style="list-style-type: none"> 1. The user clicks the profile logo button. 2. The system displays the user's profile, including: <ol style="list-style-type: none"> a. Profile Picture b. Username c. Email d. Phone Number e. Address 3. The user has options to view their Order History
Alternative scenario	<ul style="list-style-type: none"> - If the user has not uploaded a profile picture, a default image is displayed.
Exception scenario	<ul style="list-style-type: none"> - If the system fails to retrieve user details, error message will appear.

Section	Content
Designation	UC-01-05a
Name	View Order History
Authors	Nor Mirza Afiqah

Priority	Importance for system success: High - Allows users to track their past orders
Criticality	Medium - Important for customer order tracking
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view a list of their past orders, including order details, total amount, earned reward points, and delivery method.
Trigger event	The user scrolls down Profile Page.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user is logged into their account. - The system has stored order history details in the database.
Post-condition	- The user's past orders are displayed in a list format.
Result	The user can review their order history, including order ID, date, items, amount, points earned, and delivery method.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the Profile Page. 2. The user view "Order History". 3. The system displays the order history in a table format with: <ul style="list-style-type: none"> - Order ID - Date - Items Ordered - Total Amount Paid - Points Earned - Delivery Method 4. The user reviews their past orders.
Alternative scenario	- If the user has no past orders, the system displays a message: "No orders found."
Exception scenario	- If the order history fails to load, error message will appear.

Section	Content
---------	---------

Designation	UC-01-06
Name	View Reward
Authors	Nor Mirza Afiqah
Priority	Importance for system success: Medium - Enhances customer loyalty and engagement.
Criticality	Low - Does not impact order processing but improves user experience.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to view their total reward points and their purchase history, including how many points they have earned from previous orders.
Trigger event	The user navigates to the "Rewards" section from main menu.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user is logged in. - The system has stored reward points and purchase history in the database.
Post-condition	<ul style="list-style-type: none"> - The user sees their available points and purchase history.
Result	The user is informed about their reward points and how they were earned.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to the "Rewards" section. 2. The system displays the points in the interface. 3. The system retrieves the purchase history, showing: <ul style="list-style-type: none"> - Order ID - Date - Items Purchased - Total Amount Spent - Points Earned 4. The user reviews their past orders and how many points they earned from each.
Alternative scenario	<ul style="list-style-type: none"> - If the user has no reward points, the system displays: "You have 0 points. Start earning by making purchases!"

Exception scenario	<ul style="list-style-type: none"> - If the system cannot retrieve reward data, error message will appear.
--------------------	---

Section	Content
Designation	UC-01-06a
Name	View Available Reward
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Encourages users to redeem rewards.
Criticality	Medium - Important for customer engagement.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to see which rewards they can redeem based on their available points.
Trigger event	The user scrolls down "Rewards" page.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user is logged in. - The system has stored reward data.
Post-condition	<ul style="list-style-type: none"> - The user sees a list of rewards along with the required points for each.
Result	The user knows which rewards they can redeem.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to "Available Rewards". 2. The system displays rewards, including: <ul style="list-style-type: none"> - Reward Name - Points Required - Reward Description 3. The user reviews the list to see which rewards they can claim.
Alternative scenario	<ul style="list-style-type: none"> - If the user has insufficient points, the system disables the button for the reward.
Exception scenario	<ul style="list-style-type: none"> - If reward data cannot be loaded, error message will appear.

Section	Content
Designation	UC-01-07
Name	View Contact
Authors	Nor Mirza Afiah
Priority	Importance for system success: Medium - Important for customer support and inquiries.
Criticality	Low - Does not impact ordering process but improves customer experience.
Source	Nor Mirza Afiah
Responsible	Nor Mirza Afiah
Description	Allows users to view the company's contact information, including phone number, email, website, and social media links.
Trigger event	The user navigates to the "Contact" section from the navigation bar.
Actors	- User
Pre-condition	- The system has stored company contact details.
Post-condition	- The user can see contact details and options to reach out to customer support.
Result	The user finds ways to communicate with the company.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to "Contact". 2. The system displays the contact details, including: <ul style="list-style-type: none"> - Phone Number - Email Address - Website Link - Social Media Links 3. The user can click on the phone number to call, the email to send an email, or visit the website.
Alternative scenario	- If the user wants to send feedback, they can fill the form right next to contact information.
Exception scenario	- If contact details cannot be loaded, error message will appear.

Section	Content
Designation	UC-01-07a
Name	Send Feedback
Authors	Nor Mirza Afiqah
Priority	Importance for system success: Medium - Helps improve customer service and gather user feedback.
Criticality	Low - Does not affect ordering but enhances customer engagement.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows users to send feedback or inquiries by filling out a form with their first name, last name, email, phone number, and message.
Trigger event	The user clicks the "Send Feedback" button after filling form next to the contact information.
Actors	- User
Pre-condition	<ul style="list-style-type: none"> - The user has access to the feedback form. - The system is ready to receive and process feedback submissions.
Post-condition	<ul style="list-style-type: none"> - The feedback message is stored in the system for review by customer support.
Result	The user's feedback is successfully submitted, and they receive a confirmation message.
Main scenario	<ol style="list-style-type: none"> 1. The user navigates to "Send Feedback". 2. The system displays a feedback form with fields for: <ul style="list-style-type: none"> - First Name - Last Name - Email - Phone Number - Message 3. The user fills in the form with their details. 4. The user clicks the "Submit" button.

Alternative scenario	<ul style="list-style-type: none"> - If the user enters incomplete details, the system displays an error: "All fields are required. Please fill out the form completely."
Exception scenario	<ul style="list-style-type: none"> - If the system fails to store the feedback, error message will appear.

Section	Content
Designation	UC-01-08
Name	User Logout
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Essential for session management and security.
Criticality	Medium - Ensures users can safely exit the system.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows user to log out of their accounts, clearing their session and returning them to the homepage or login page.
Trigger event	The users click the "Logout" button.
Actors	<ul style="list-style-type: none"> - User
Pre-condition	<ul style="list-style-type: none"> - The user is already logged into the system.
Post-condition	<ul style="list-style-type: none"> - The system clears the admin session and redirects them to the homepage or login page.
Result	The user is successfully logged out and cannot access restricted pages without logging in again.
Main scenario	<ol style="list-style-type: none"> 1. The user clicks the "Logout" button in the navigation menu. 2. The system processes the session termination. 3. The system redirects the admin to: <ul style="list-style-type: none"> - Homepage
Alternative scenario	<ul style="list-style-type: none"> - If the user tries to access a restricted page after logging out, they are redirected to the login page.
Exception scenario	<ul style="list-style-type: none"> - If session termination fails due to a server issue, the system will display error message.

Section	Content
Designation	UC-02-01
Name	Admin Login
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Required for admin access to manage the system.
Criticality	High - Ensures security and access control
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows the admin to log into the system by entering a username and password that are already stored in the database.
Trigger event	The admin navigates to the Admin Login page.
Actors	<ul style="list-style-type: none"> - Admin
Pre-condition	<ul style="list-style-type: none"> - The admin account is already registered in the database. - The admin has access to the login page.
Post-condition	<ul style="list-style-type: none"> - The admin is successfully logged in and redirected to the admin dashboard.
Result	The admin gains access to system management functionalities.
Main scenario	<ol style="list-style-type: none"> 1. The admin opens the Admin Login page. 2. The system displays input fields for: <ul style="list-style-type: none"> - Username - Password 3. The admin enters their username and password. 4. The admin clicks the "Login" button. 5. The system verifies the credentials against the database. 6. If valid, the system redirects the admin to the admin dashboard.
Alternative scenario	<ul style="list-style-type: none"> - Admin will be logging on the same page as users.

Exception scenario	<ul style="list-style-type: none"> - If the username or password is incorrect, the system displays an error: "Invalid credentials. Please try again."
--------------------	--

Section	Content
Designation	UC-02-02
Name	View User
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Essential for user management in the admin panel.
Criticality	Medium - Allows the admin to monitor and manage users.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows the admin to view the list of registered users, including their usernames, email, phone numbers, and addresses.
Trigger event	The admin clicks on the "View Users" option in the admin dashboard.
Actors	<ul style="list-style-type: none"> - Admin
Pre-condition	<ul style="list-style-type: none"> - The admin is logged into the admin panel.
Post-condition	<ul style="list-style-type: none"> - The system displays a list of registered users with relevant details.
Result	The admin successfully views the user list for monitoring or management.
Main scenario	<ol style="list-style-type: none"> 1. The admin logs into the admin panel. 2. The admin clicks on "View Users". 3. The system retrieves the list of registered users from the database. 4. The system displays user information such as: <ol style="list-style-type: none"> a. Id b. Username c. Email d. Phone number e. Address

	f. Points g. Role
Alternative scenario	- If no users are found, the system displays: "No registered users found."
Exception scenario	- If the database query fails, the system will display error message.

Section	Content
Designation	UC-02-03
Name	View Sales
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Essential for tracking business performance.
Criticality	High - Provides insights into sales trends and revenue.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	Allows the admin to view sales data, including total revenue, number of orders, and sales trends over time.
Trigger event	The admin clicks on the "View Sales" option in the admin dashboard.
Actors	- Admin
Pre-condition	- The admin is already logged into the system.
Post-condition	- The system displays sales records and key performance indicators.
Result	The admin successfully reviews sales data to assess performance and trends.
Main scenario	1. The admin logs into the admin panel. 2. The admin clicks on "View Sales". 3. The system retrieves sales data from the database, including: <ul style="list-style-type: none"> - Total revenue - Number of orders - Sales by date

	<ul style="list-style-type: none"> - Total Quantity Sold <p>4. The system displays the sales report in a structured format with charts or tables.</p>
Alternative scenario	<ul style="list-style-type: none"> - If no sales data is available, the system displays: "No sales records found."
Exception scenario	<ul style="list-style-type: none"> - If If the database query fails, the system will display error message.

Section	Content
Designation	UC-02-04
Name	Admin Logout
Authors	Nor Mirza Afqah
Priority	Importance for system success: High - Essential for session management and security.
Criticality	Medium - Ensures admins can safely exit the system.
Source	Nor Mirza Afqah
Responsible	Nor Mirza Afqah
Description	Allows admin to log out of their accounts, clearing their session and returning them to the homepage or login page.
Trigger event	The admin clicks the "Logout" button.
Actors	<ul style="list-style-type: none"> - Admin
Pre-condition	<ul style="list-style-type: none"> - The admin is already logged into the system.
Post-condition	<ul style="list-style-type: none"> - The system clears the admin session and redirects them to the homepage or login page.
Result	The admin is successfully logged out and cannot access restricted pages without logging in again.
Main scenario	<p>4. The admin clicks the "Logout" button in the navigation menu.</p> <p>5. The system processes the session termination.</p> <p>6. The system redirects the admin to:</p> <ul style="list-style-type: none"> - Homepage
Alternative scenario	<ul style="list-style-type: none"> - If the admin tries to access a restricted page after logging out, they are redirected to the login page.

Exception scenario	<ul style="list-style-type: none"> - If session termination fails due to a server issue, the system will display error message.
--------------------	--

Section	Content
Designation	UC-03-01
Name	Validate Input Signup
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Prevents invalid user registrations.
Criticality	High - Ensures valid user data and avoids duplicate accounts.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	<p>The system validates user input when signing up by ensuring:</p> <ul style="list-style-type: none"> - Username is alphanumeric and not taken. - Email is in correct format and unique. - Phone number is numeric and unique. - Address is provided. - Password meets security criteria.
Trigger event	The user submits the sign-up form.
Actors	<ul style="list-style-type: none"> - System
Pre-condition	<ul style="list-style-type: none"> - The sign-up form is accessible.
Post-condition	<ul style="list-style-type: none"> - A valid user account is created or an error is displayed.
Result	Ensures only valid user accounts are registered.
Main scenario	<ol style="list-style-type: none"> 1. The user enters username, email, phone number, address, and password. 2. The system validates: <ul style="list-style-type: none"> - Username (Unique, no special characters) - Email (Valid format, not registered) - Phone number (Numeric) - Address (Not empty) - Password (Minimum 8 characters) 3. If all fields are valid, the system creates an account.
Alternative scenario	<ul style="list-style-type: none"> - If username exists, display: "Username already taken."

	<ul style="list-style-type: none"> - If email exists, display: "Email already registered." - If password is weak, display: "Password must be at least 8 characters."
Exception scenario	<ul style="list-style-type: none"> - If database validation fails, display: "Sign-up service unavailable."

Section	Content
Designation	UC-03-02
Name	Validate Input Login
Authors	Nor Mirza Afiqah
Priority	Importance for system success: High - Ensures only valid credentials are processed.
Criticality	High - Prevents login errors and enhances security.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	<p>The system validates user inputs when logging in by checking:</p> <ul style="list-style-type: none"> - Username or email is not empty. - Password is not empty. - The credentials match an existing user in the database.
Trigger event	The user submits the login form.
Actors	<ul style="list-style-type: none"> - System
Pre-condition	<ul style="list-style-type: none"> - The login form is accessible.
Post-condition	<ul style="list-style-type: none"> - The system either logs in the user or shows an error message.
Result	The system ensures only valid credentials proceed to authentication.
Main scenario	<ol style="list-style-type: none"> 1. The user enters username/email and password. 2. The system checks if both fields are filled. 3. The system verifies the credentials in the database. 4. If valid, the user is redirected to the homepage.
Alternative scenario	<ul style="list-style-type: none"> - If fields are empty, display: "All fields are required."

	<ul style="list-style-type: none"> - If credentials do not match, display: "Invalid username/email or password."
Exception scenario	<ul style="list-style-type: none"> - If the database is unavailable, display: "Login service temporarily unavailable."

Section	Content
Designation	UC-03-03
Name	Validate Input Feedback
Authors	Nor Mirza Afiqah
Priority	Importance for system success: Medium - Ensures valid feedback submissions.
Criticality	Medium - Prevents spam and incomplete submissions.
Source	Nor Mirza Afiqah
Responsible	Nor Mirza Afiqah
Description	<p>The system validates the feedback form input to ensure:</p> <ul style="list-style-type: none"> - First Name & Last Name are not empty. - Email follows a valid format. - Phone Number contains only numbers. - Message
Trigger event	The user submits feedback.
Actors	<ul style="list-style-type: none"> - System
Pre-condition	<ul style="list-style-type: none"> - The feedback form is accessible.
Post-condition	<ul style="list-style-type: none"> - A valid feedback entry is stored, or an error is displayed.
Result	Ensures only valid feedback is submitted.
Main scenario	<ol style="list-style-type: none"> 1. The user enters first name, last name, email, phone number, and message. 2. The system checks: <ul style="list-style-type: none"> - Names are not empty. - Email is valid. - Phone number is numeric. - Message 3. If valid, feedback is stored.

Alternative scenario	<ul style="list-style-type: none"> - If fields are empty, display: "All fields are required." - If message is too short, display: "Message must be at least 10 characters."
Exception scenario	<ul style="list-style-type: none"> - If the feedback system fails, display: "Feedback submission failed. Try again later."

3.3 System Architecture

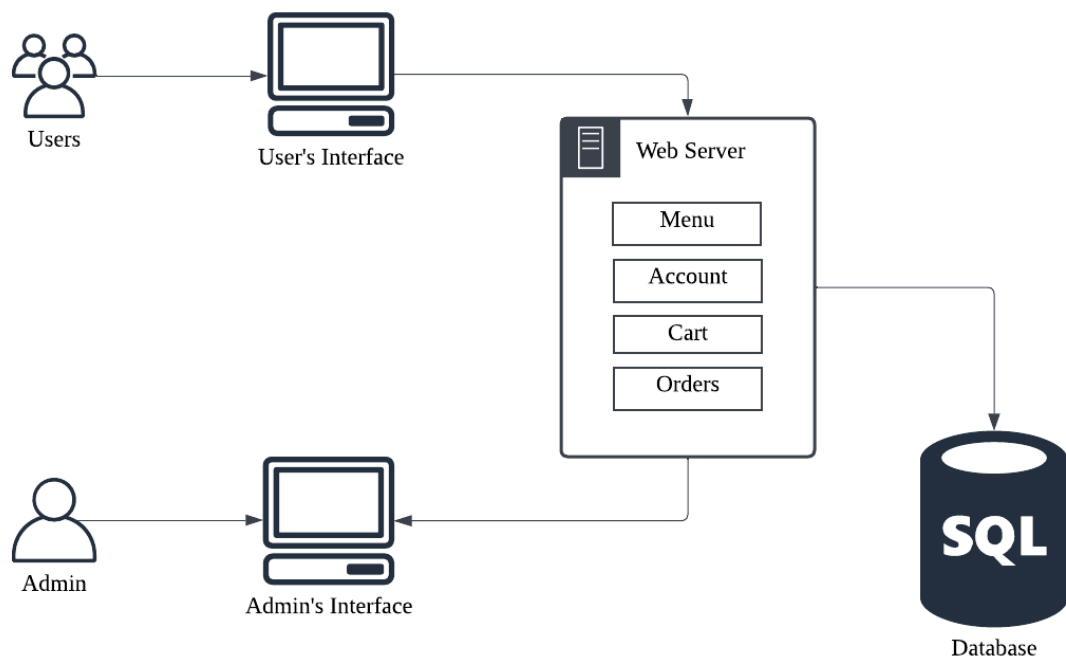


Figure 2. Deli Pizza System Architecture

- **Users** – General customers can access the system through their user interface where users can view the menu, place orders and view their account.
- **Admin** – The administrator access admin interface where they can manage users details and view the purchases including report summary.
- **Web Server** – Handle all users' request where it connect all the interface to the database and manage features like:
 - **Menu**: Displays pizza and other menu options.
 - **Account**: Handles user registration, login, and user profile.
 - **Cart**: Manages items that users add to cart before processing order.
 - **Orders**: Processes and tracks customer orders.

- Database – Store all system’s data including user details, orders and feedback received. The web server will retrieve and updates data in the database as users and admins interact with the system.

4. IMPLEMENTATION

4.1 User Manual of Delicious Pizza System

Getting Started

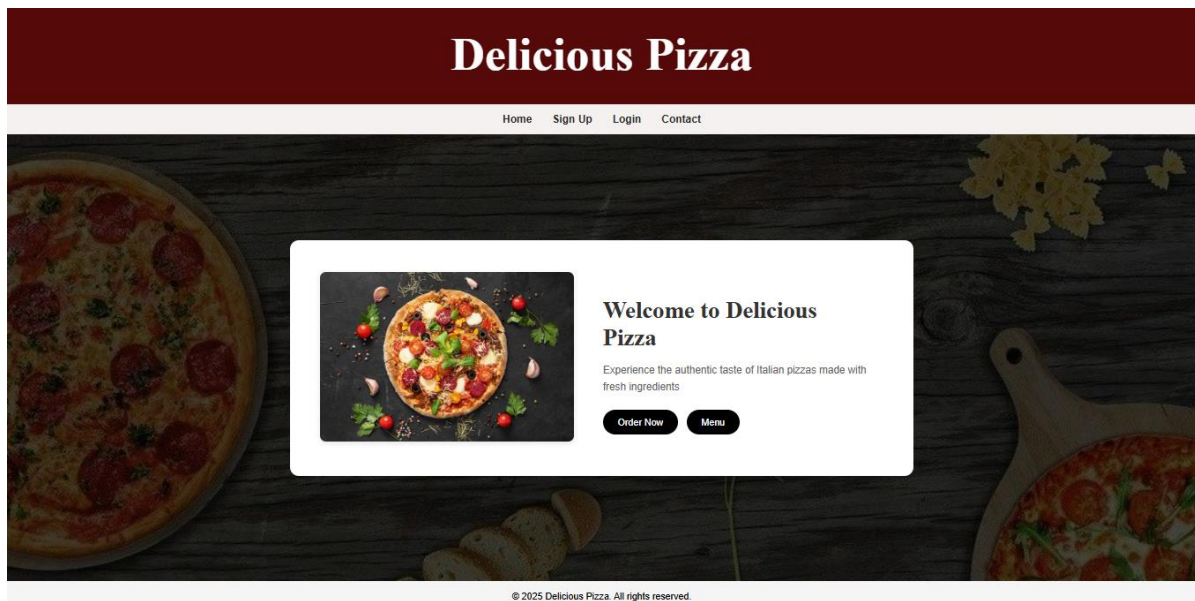


Figure 3. Delicious Pizza Homepage

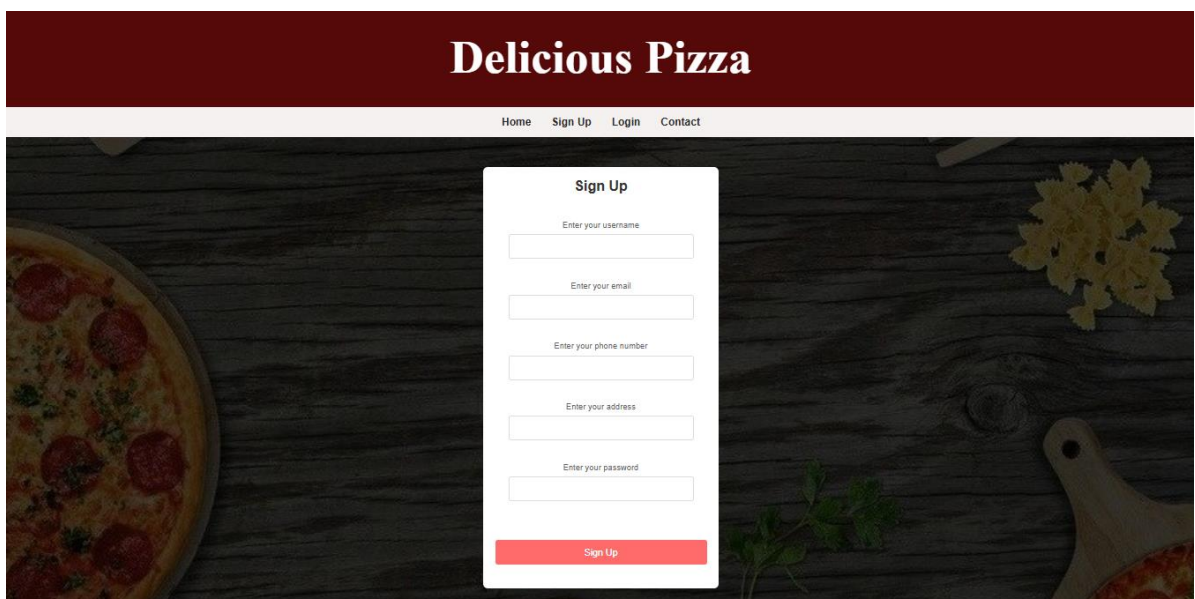


Figure 4. Sign up page

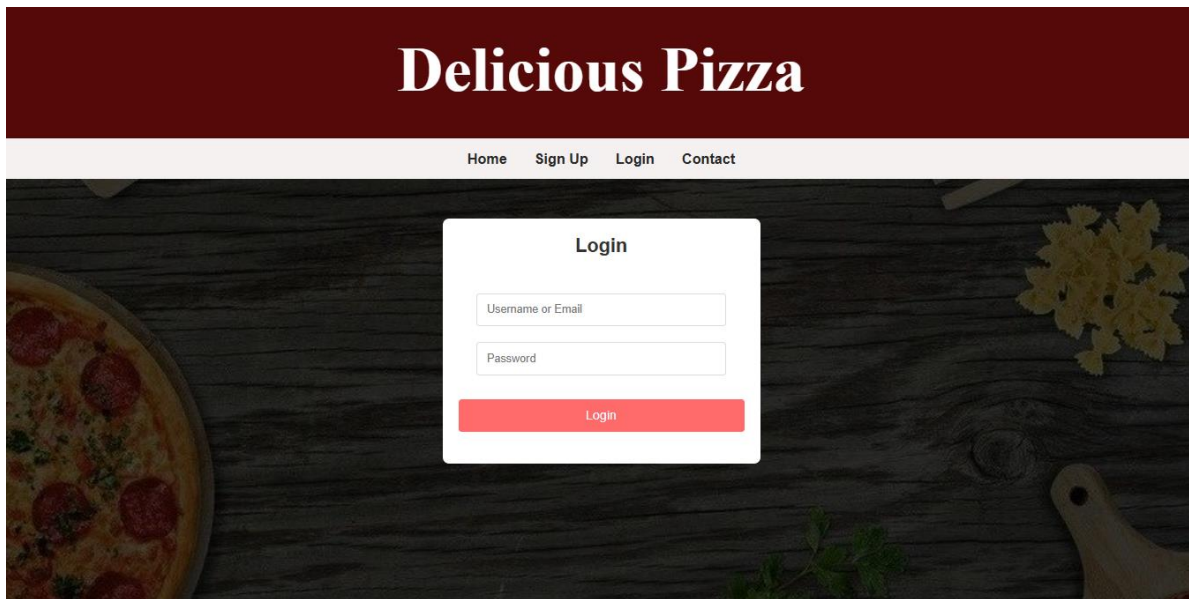


Figure 5. Login page

1. Visit the Delicious Pizza website and navigate to the login page.
2. If you are a new user, click on the "Sign Up" button to create a new account. Fill in the required information, including your name, email, and a secure password.
3. If you already have an account, enter your email/username and password in the login form and click "Login".

Browse Menu

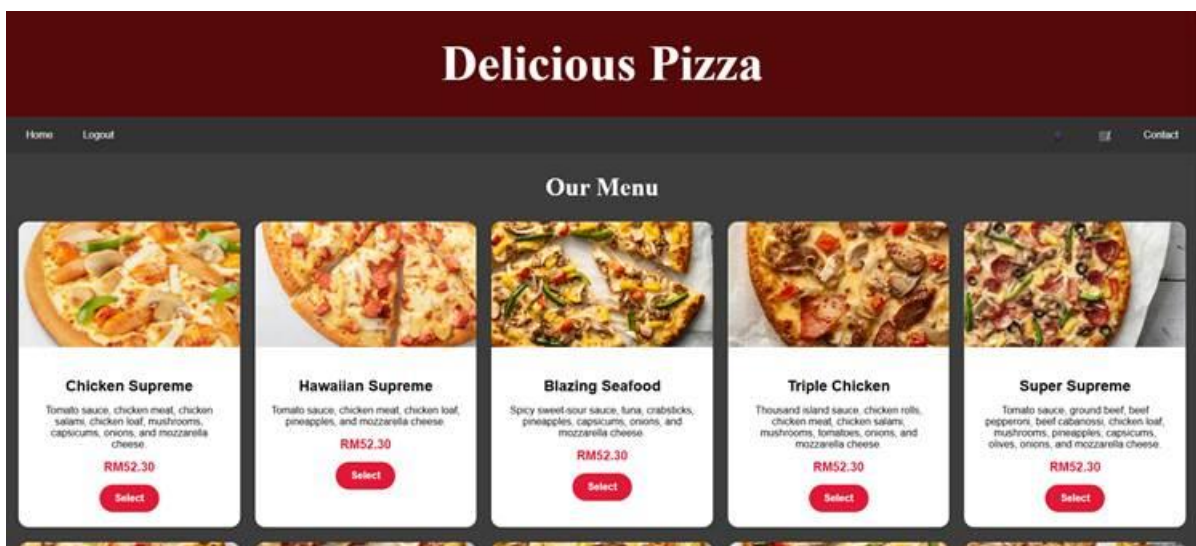


Figure 6. Menu Selection page

1. After logging in, you will be taken to the main menu page, where you can view the available pizza options.

2. Each pizza item is displayed with an image, name, description, and price.
3. To view more details about a specific pizza, click on the item to be taken to the menu details page.

Customize Order

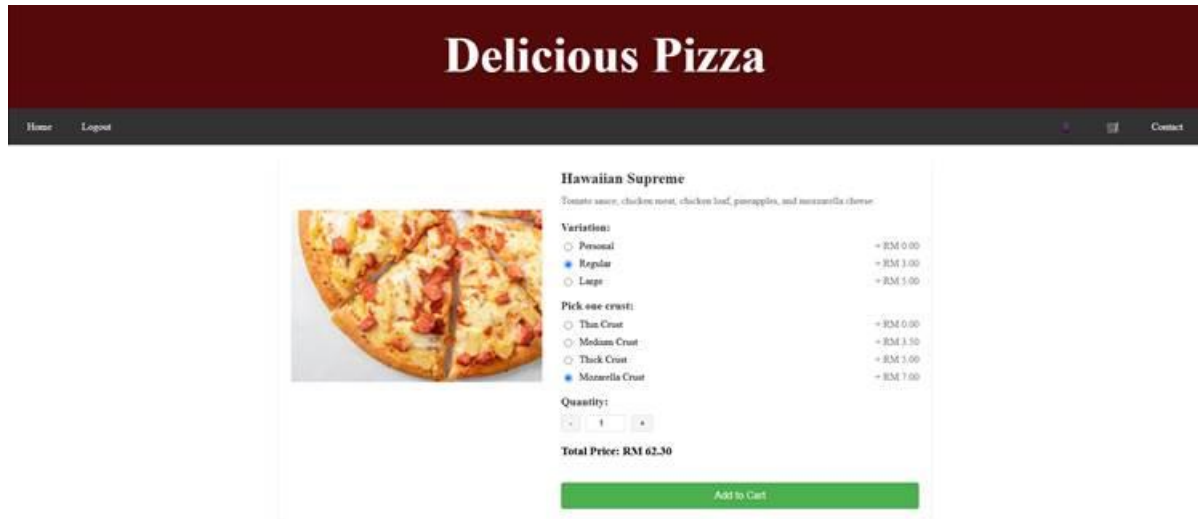


Figure 7. Customize Selection page

1. On the menu details page, you can select the size and crust type for your pizza.
2. The system will display the base price and any additional charges for the size and crust selections.
3. Use the quantity buttons to adjust the number of pizzas you would like to order.
4. Once you are satisfied with your order, click the "Add to Cart" button to add the item to your shopping cart.

View Cart

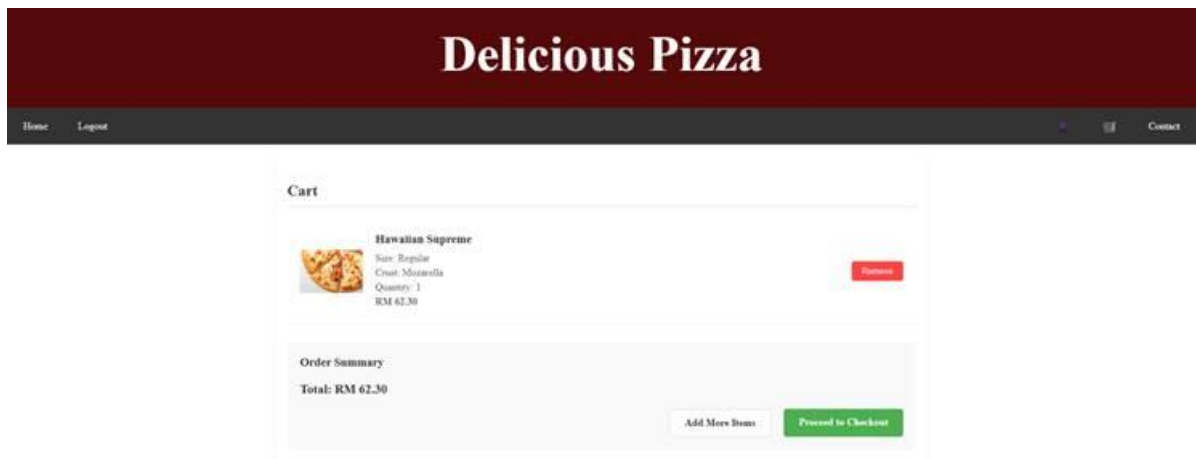


Figure 8. User Cart page

1. To view the items in your cart, click on the "Cart" link in the header.
2. The cart page will display a summary of your order, including the items, quantities, and total cost.
3. If you need to make any changes to your order, you can remove any items from the cart.
4. You also can add more items you wanted.
5. When you are ready to proceed, click the "Checkout" button.

Checkout and Payment

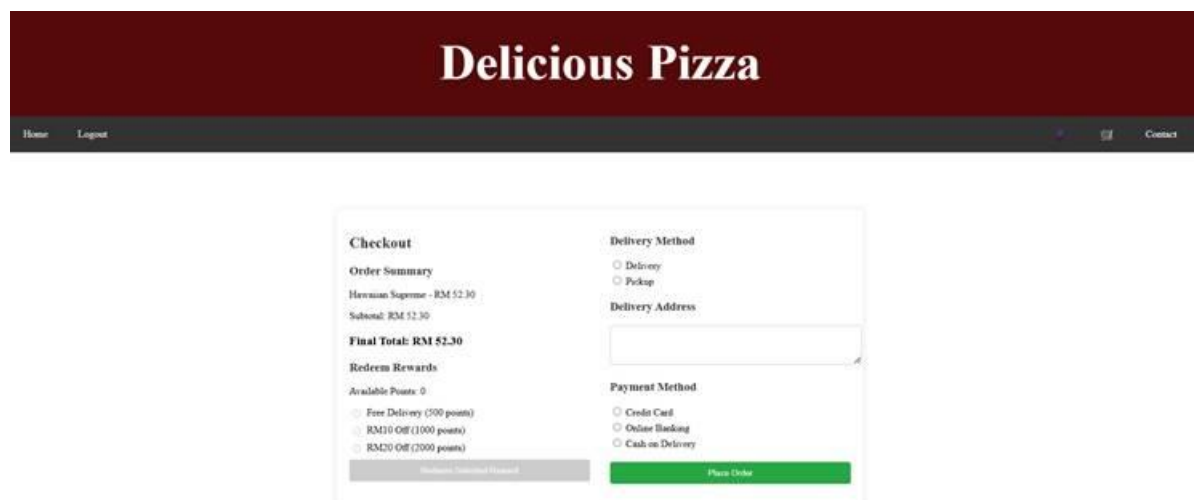


Figure 9. Checkout page

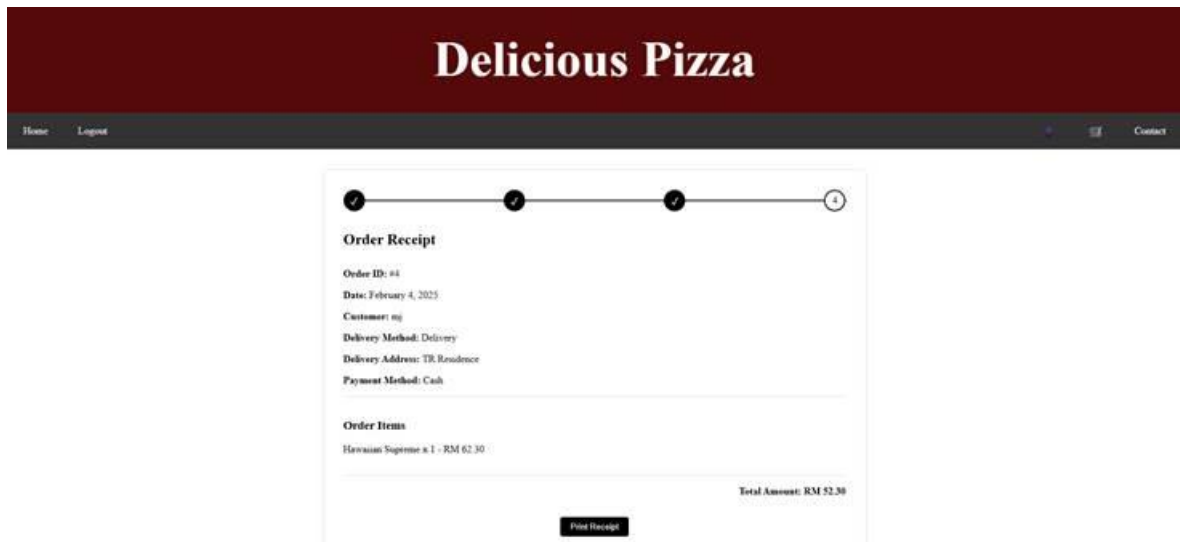


Figure 10. Receipt page

1. On the checkout page, you will be asked to select your delivery method (delivery or pickup) and your payment method (credit card, online banking, or cash on delivery).
2. If you have any available rewards points, you can redeem them on this page to receive a discount on your order.
3. Review the order summary and total cost, then click the "Place Order" button to complete your purchase.
4. The receipt will be displayed with the items' details and total amount.

Personal Information

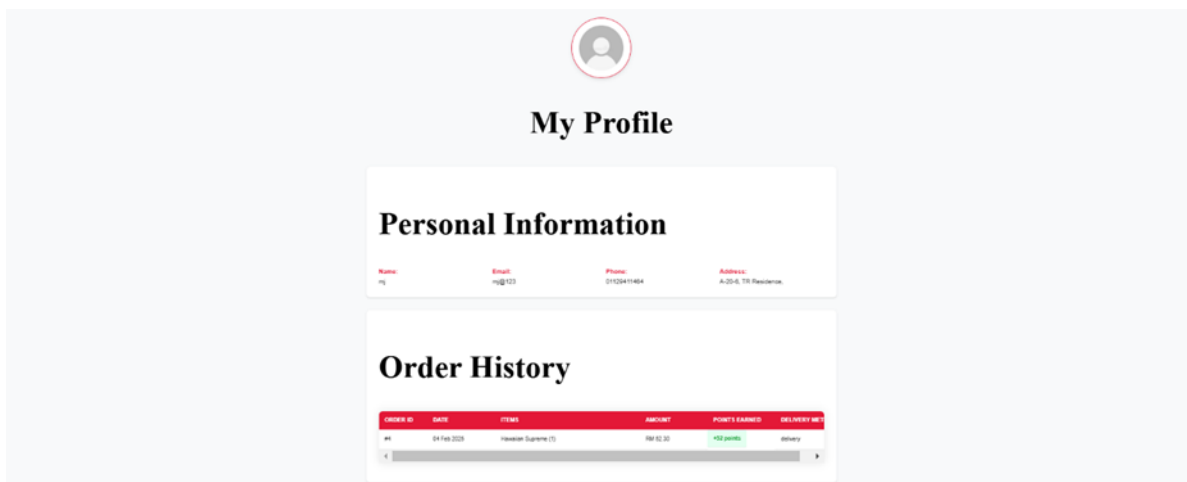


Figure 11. User Profile page

1. On the My Profile page, you can overview your personal information and order history.
2. The page will display your name, email, phone number and address.

3. You also can see your order history by the order ID, date, items, amount, points you earned and delivery method.

Rewards Page

My Rewards

Available Points

52

Available Rewards

- Free Delivery:**
500 points
Save RM3 on delivery fees
- RM10 Off:**
1000 points
Get RM10 off your order
- RM20 Off:**
2000 points
Get RM20 off your order

Purchase History

Order ID	Date	Items	Amount	Points Earned
#4	04 Feb 2025	Flavours Supreme (1)	RM 52.30	+52 points

How to Earn Points

- Earn 1 point for every RM1 spent
- Points are automatically added to your account after each purchase
- Use your points to redeem rewards during checkout

Figure 12. My Rewards page

1. The Delicious Pizza rewards program allows you to earn points for every purchase you make.
2. You can view your available points, as well as the current reward options, on the "My Rewards" page.
3. To redeem your points, simply select the desired reward during the checkout process.

Send Feedback

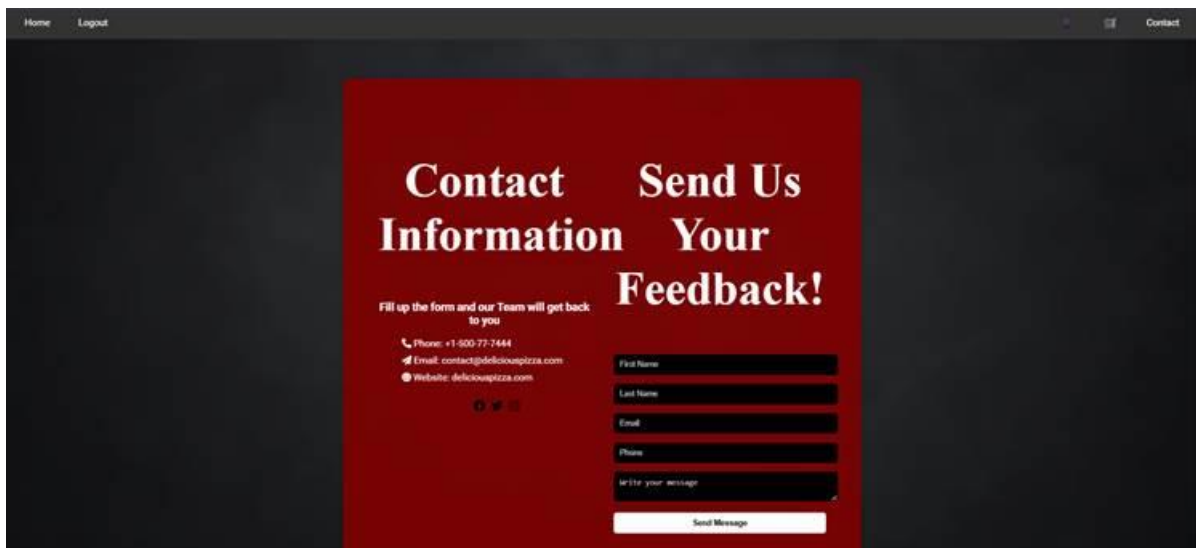


Figure 13. Feedback and Contact page

1. The Feedback page serves as your direct line of communication with our team with our phone number, email address, and website. We've also included links to our social media profiles so you can connect with us on various platforms.
2. The feedback form where you can send us your message, questions or concerns.
3. The form includes fields for your first name, last name, email address, phone number, and a space for your message.

4.2 Technical Specifications

4.2.1 Exception Handling – Sign up, Login

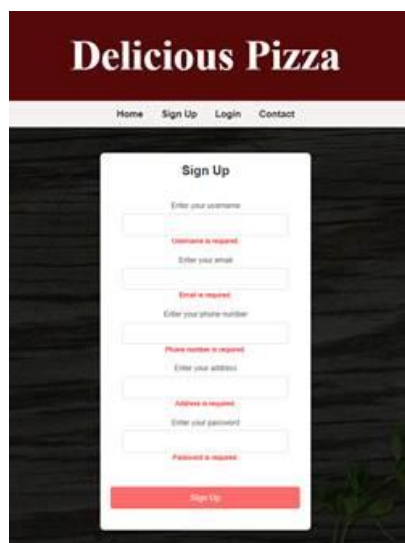


Figure 14. Sign up Input Error Validation

If the user enters invalid input (e.g., missing required fields) during the signup, the system will display an error message to enter their details.

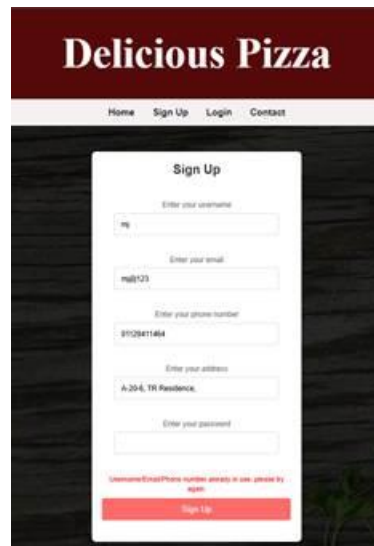


Figure 15. Exist User Account Validation

If the user tries to sign up with an already exist username or email, the system will display an error message prompting the user to login.

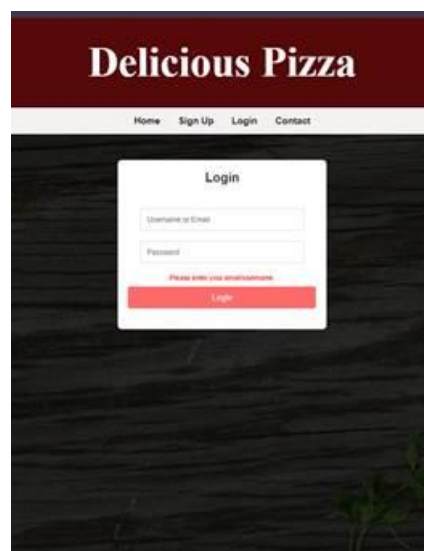


Figure 16. Login Input Error Validation

If the user tries to login without entering any input, the system will display an error message prompting the user to enter their username or email.

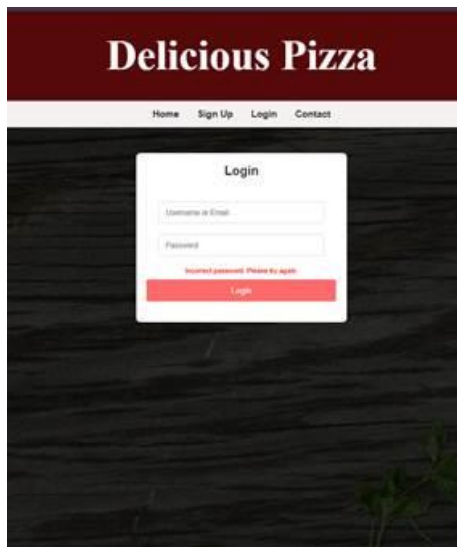


Figure 17. Incorrect Password Validation

If the user tries to login with incorrect password, the system will display an error message prompting the user to try again.

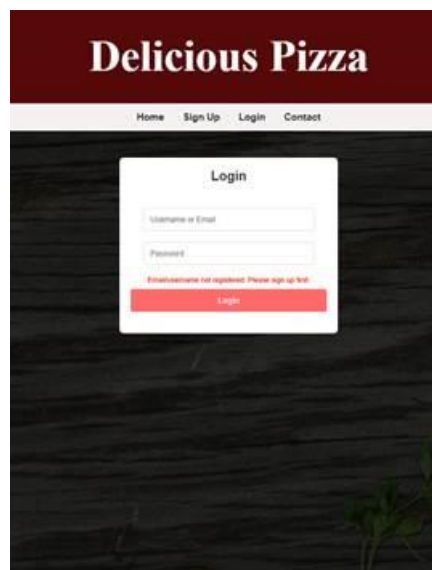


Figure 18. Invalid User Account Validation

If the user tries to login without a valid username or email, the system will display an error message prompting the user to sign up first.

4.2.2 Design Patterns Applied

- **Singleton Pattern** – Database Connection
Ensures that there is only one instance of the database connection throughout the application to avoid multiple connections and ensure resource efficiency.
- **Factory Pattern** – Pizza Menu
The Factory Pattern creates different items of pizza menu.
- **Template Pattern** – Login and Signup
Defines a standard structure for login and signup processes while allowing specific steps (like validation) to be customized.
- **Observer Pattern** – Price Changes
Implemented to notify users about price changes for pizzas. This enhances the system by providing real-time updates to observers without tightly coupling them to the pizza object.

4.3 Explanations For Every Function and Module

➤ User Authentication (Login and Signup)

User authentication is the gateway to the system. Without it, users cannot place orders, earn rewards, or manage their profiles. The authentication module consists of **signup** and **login** functions, each ensuring data integrity and security.

- **Signup Process:** When a user registers, the system validates the input to ensure all fields—**username, email, phone number, password, and address**—are provided. The password is hashed using `password hash()` before being stored in the database. If the email or username already exists, the system rejects the registration.
- **Login Process:** Upon entering credentials, the system checks for a matching record. `password_verify()` ensures the password matches the hashed version stored in the database. If successful, session variables (`$_SESSION['user_id']`, `$_SESSION['role']`) are set, directing the user to the appropriate dashboard.

This module follows the **Singleton Pattern** for database connection, ensuring a single instance is reused across multiple authentication requests. It prevents unnecessary connections and optimizes performance.

➤ **Menu and Pizza Selection**

The menu module dynamically displays available pizzas with details such as **name, image, description, price, and customization options**. Users can:

- Select **size and crust type**, which updates the price accordingly.
- Adjust **quantity** before adding to the cart.

This module employs the **Factory Pattern**, where a `PizzaFactory` class generates different pizza objects based on user selection. It abstracts the creation logic, making it easier to add new pizza varieties without modifying core logic.

➤ **Shopping Cart Management**

A well-structured shopping cart enhances the user experience. This module allows users to:

- **View items in their cart**, along with real-time price calculations.
- **Remove items** if they change their mind.
- **Increase or decrease quantities**, triggering automatic price updates.

Every time an action is performed, the **cart updates asynchronously** without requiring a full page reload. This enhances speed and usability.

➤ **Order Processing and Checkout**

Placing an order should be seamless. This module handles:

1. **Delivery or Pickup Selection:** Users choose between home delivery and self-pickup.
2. **Payment Options:** The system supports multiple payment methods: **credit card, online banking, and cash on delivery**.
3. **Rewards System:** If users have loyalty points, they can redeem them for discounts at checkout.
4. **Final Order Review:** Before confirming, users see a summary of their order.

Once an order is placed, a unique **order ID** is generated, and the order is stored in the database. The system then redirects users to a confirmation page displaying their **receipt**.

➤ **User Profile Management**

The profile module gives users full control over their information. The **My Profile** page includes:

- **Personal Details:** Name, email, phone number, and address.
- **Order History:** A list of past orders with **order ID, date, items purchased, total amount, and points earned**.

Users can edit their personal details, ensuring their data remains up-to-date.

➤ **Reward System**

A great incentive drives engagement. The **Rewards System** module tracks points accumulated from previous purchases. Every RM1 spent earns a point, which can later be redeemed for discounts.

On the **My Rewards** page:

- Users see their **current points balance**.
- Available rewards are displayed, allowing them to **redeem points at checkout**.

This module follows the **Observer Pattern**, ensuring real-time updates. When a user earns or redeems points, the system instantly reflects the change in their rewards balance.

➤ **Feedback System**

Customer feedback is crucial for improvement. The feedback module provides:

- A **contact section** displaying customer support details.
- A **feedback form** where users can submit their thoughts.

Feedback is stored in the database and can be viewed by administrators for quality improvements.

➤ Error Handling and Exception Management

The system includes **robust validation and error handling**:

- Missing fields trigger **error messages**.
- Incorrect credentials display **login errors**.
- Payment failures return **specific error notifications**.

This ensures users receive immediate feedback, improving usability.

Every function and module in this pizza ordering system is designed with usability, security, and efficiency in mind. From the **Factory Pattern for pizza creation** to the **Observer Pattern for real-time price updates**, the system ensures seamless user interaction while maintaining structured and scalable code. Each component plays a crucial role in delivering a smooth and enjoyable ordering experience.

4.4 Discussions

4.4.1 Singleton Pattern

The Singleton Pattern is a design pattern used to ensure that a class has only one instance and provides a global point of access to that instance. In the PHP, the Singleton pattern is applied in the **DatabaseConnection** class as shown below:

```
class DatabaseConnection {  
    private static $instance = null;
```

This variable holds the single instance of the class. It is set to null initially.

```
private $connection;  
  
private function __construct() {  
    $this->connection = mysqli_connect('localhost', 'root', '', 'delipizza');  
  
    if (mysqli_connect_errno()) {  
        throw new Exception("Failed to connect to MySQL: " . mysqli_connect_error());  
    }  
}
```

The constructor is private, ensuring that it cannot be instantiated directly from outside the class. It creates a database connection using MySQL. If the connection fails, an exception is thrown.

```
public static function getInstance() {  
    if (self::$instance == null) {  
        self::$instance = new DatabaseConnection();  
    }  
    return self::$instance;  
}  
  
public function getConnection() {  
    return $this->connection;  
}  
  
// Prevent cloning of the instance  
private function __clone() {}  
  
// Prevent unserializing of the instance  
public function __wakeup() {}  
}
```

The getInstance method checks if the \$instance is null. If it is, a new DatabaseConnection object is created. This method ensures that only one instance of the DatabaseConnection class exists throughout the application's lifecycle. The __clone method is private to prevent cloning of the instance. The __wakeup method is defined but is empty. This method is called when an object is unserialized. Its purpose is to prevent re-creating the instance after serialization.

- **How the User Interface works**

Sign Up Section:

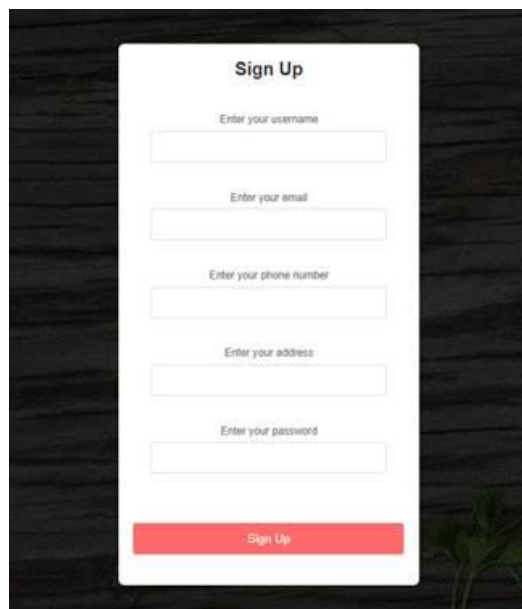


Figure 19. Sign up section

User submits the registration form with their username, email, password, phone, and address.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    $username = trim($_POST['username']);  
    $email = trim($_POST['email']);  
    $password = trim($_POST['password']);  
    $phone = trim($_POST['phone']);  
    $address = trim($_POST['address']);  
}
```

The backend script processes the POST request and retrieves the form data.

Sign Up

Enter your username

 Username is required.

Enter your email

 Email is required.

Enter your phone number

 Phone number is required.

Enter your address

 Address is required.

Enter your password

 Password is required.

Sign Up

Figure 20. Sign up Validation Check

Validation checks are performed:

- Ensures all fields (username, email, phone, password, and address) are provided.
- Checks that the password meets the minimum length requirement (8 characters).
- Verifies that the username, email, or phone number does not already exist in the database.

```
// If there are no errors, proceed with registration
if (empty($errors)) {
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
    $insert_query = "INSERT INTO users (username, email, password, phone, address, role)
    VALUES (?, ?, ?, ?, ?, 'user')";
    $insert_stmt = mysqli_prepare($dbc, $insert_query);
    mysqli_stmt_bind_param($insert_stmt, 'sssss', $username, $email, $hashed_password, $phone, $address);

    if (mysqli_stmt_execute($insert_stmt)) {
        $_SESSION['success'] = 'Registration successful! You can now log in to start using the system.';
        exit();
    } else {
        $errors[] = 'Something went wrong. Please try again later.';
    }
}
```

User Registration:

- If validation is successful, the password is hashed using `password_hash()` and a new record is inserted into the users table.

- If registration is successful, the user is shown a success message. The user is expected to log in afterward.

Login Section:

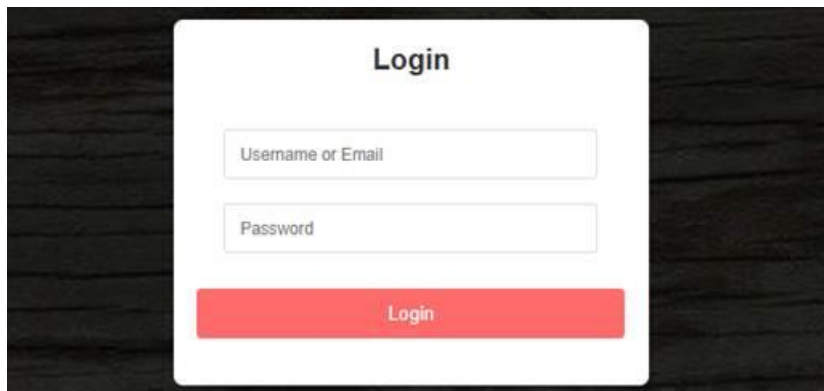
A screenshot of a login form titled "Login" centered on a dark background. The form is a white rounded rectangle containing two input fields: "Username or Email" and "Password". Below these fields is a red button labeled "Login".

Figure 21. Login section

Form Processing:

- Upon form submission (via POST request), the backend script fetches the user input (`$_POST['username']`, `$_POST['password']`).
- It checks for missing inputs, such as username or password. If any of these are missing, an exception is thrown.
- The DatabaseConnection class is used to connect to the database, and a SQL query is prepared to fetch the user credentials from the users table based on the username/email.

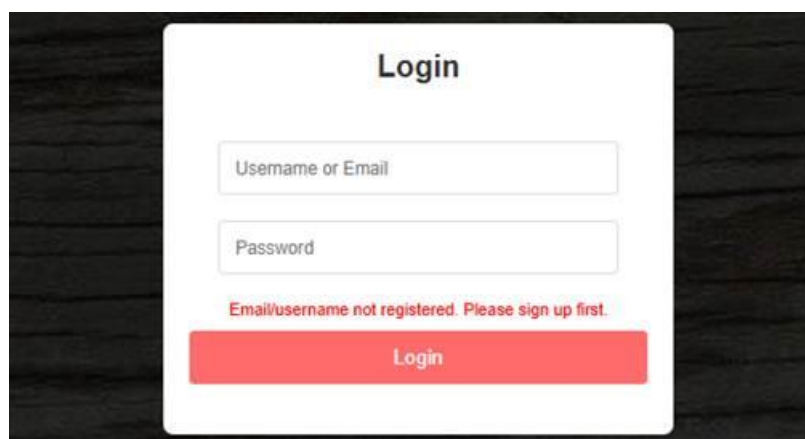
A screenshot of the login form showing an error state. The "Username or Email" and "Password" input fields are present. Below them, a red error message reads: "Email/username not registered. Please sign up first." The red "Login" button is still at the bottom.

Figure 22. Login Validation

Login Validation:

- If a matching user is found, the password is verified using `password_verify()`. If the password is incorrect, an error message is shown.
- If login is successful, session variables are set (`$_SESSION['user_id']`, `$_SESSION['username']`, `$_SESSION['role']`), and the user is redirected to either the `Admin.php` or `UserMenu.php` based on their role.

The Singleton pattern ensures that the database connection (`$dbc`) is created only once throughout the lifecycle of the application. When the application requests a database connection, it uses the `DatabaseConnection::getInstance()` method to retrieve the same instance every time. This minimizes overhead by preventing the creation of multiple database connections.

4.4.2 Abstract Factory Design Pattern

This factory system used to display different pizza types in home menu that user can view all the menus as it shown below:

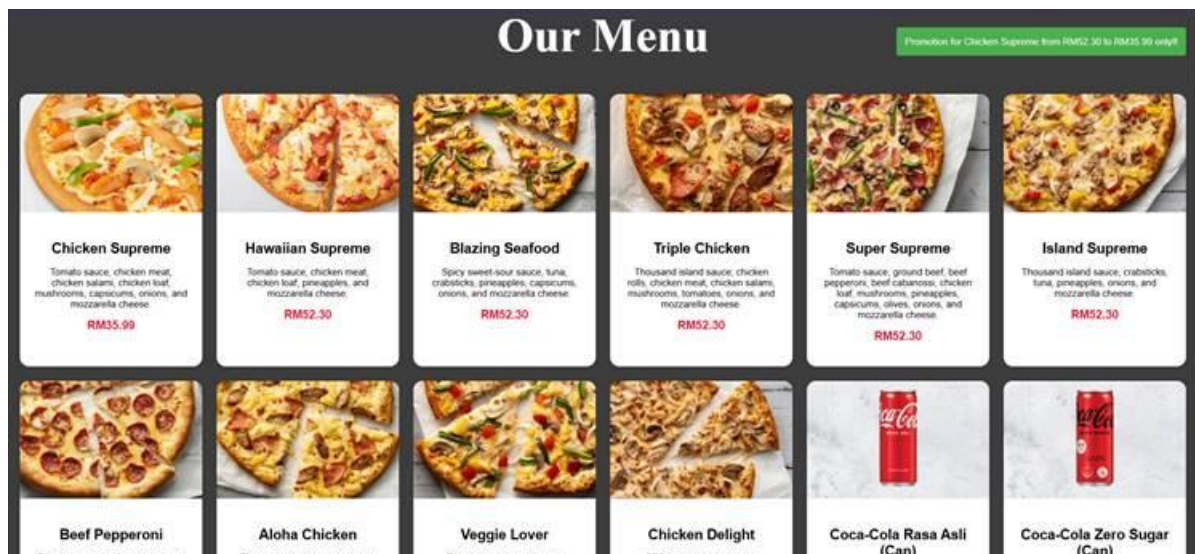


Figure 23. Menu Selection

How the User Interface works

```

class Pizza {
    private $name;
    private $description;
    private $price;
    private $image;
    private $subject;

    public function __construct($name, $description, $price, $image) {
        $this->name = $name;
        $this->description = $description;
        $this->price = $price;
        $this->image = $image;
        $this->subject = new PizzaSubject();
    }

    public function getName() {
        return $this->name;
    }

    public function getDescription() {
        return $this->description;
    }

    public function getPrice() {
        return $this->price;
    }

    public function getImage() {
        return $this->image;
    }

    public function addObserver(Observer $observer) {
        $this->subject->attach($observer);
    }

    public function notifyPriceChange($newPrice) {
        $oldPrice = $this->price;

```

The Pizza class represents the product that the factory creates. It has properties like name, description, price, and image. It also integrates with the observer pattern to notify users of price changes.

```

class PizzaFactory {
    public static function createPizza($row) {
        return new Pizza(
            $row['FoodName'],
            $row['About'],
            $row['FoodPrice'],
            $row['ItemImage']
        );
    }
}

```

The PizzaFactory class is responsible for creating pizzas details based on the input provided. This class abstracts the creation logic, making it easy to add new pizza details.

```

$pizzas = [];
if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        $pizza = PizzaFactory::createPizza($row);
        $pizza->addObserver($notificationManager);
        $pizzas[] = $pizza;
    }
} else {
    echo "0 results";
}

```

```

<div class="menu-grid">
    <?php foreach($pizzas as $pizza): ?>
    <div class="menu-item">
        <div class="menu-image">
            getName()); ?>" />
        </div>
        <div class="menu-details">
            <h3><?php echo htmlspecialchars($pizza->getName()); ?></h3>
            <p><?php echo htmlspecialchars($pizza->getDescription()); ?></p>
            <p class="price">RM<?php echo number_format($pizza->getPrice(), 2); ?></p>
        </div>
    </div>
    <?php endforeach; ?>
</div>

```

In the HomeMenu, the user can view all the menu with the description. The created Pizza object is then displayed with its details.

4.4.3 Observer Pattern

The Observer and PizzaSubject classes implement the Observer pattern, allowing the Pizza class to notify observers (like the NotificationManager) about changes, such as price updates. This promotes loose coupling and the ability to add new observers without modifying the Pizza class.

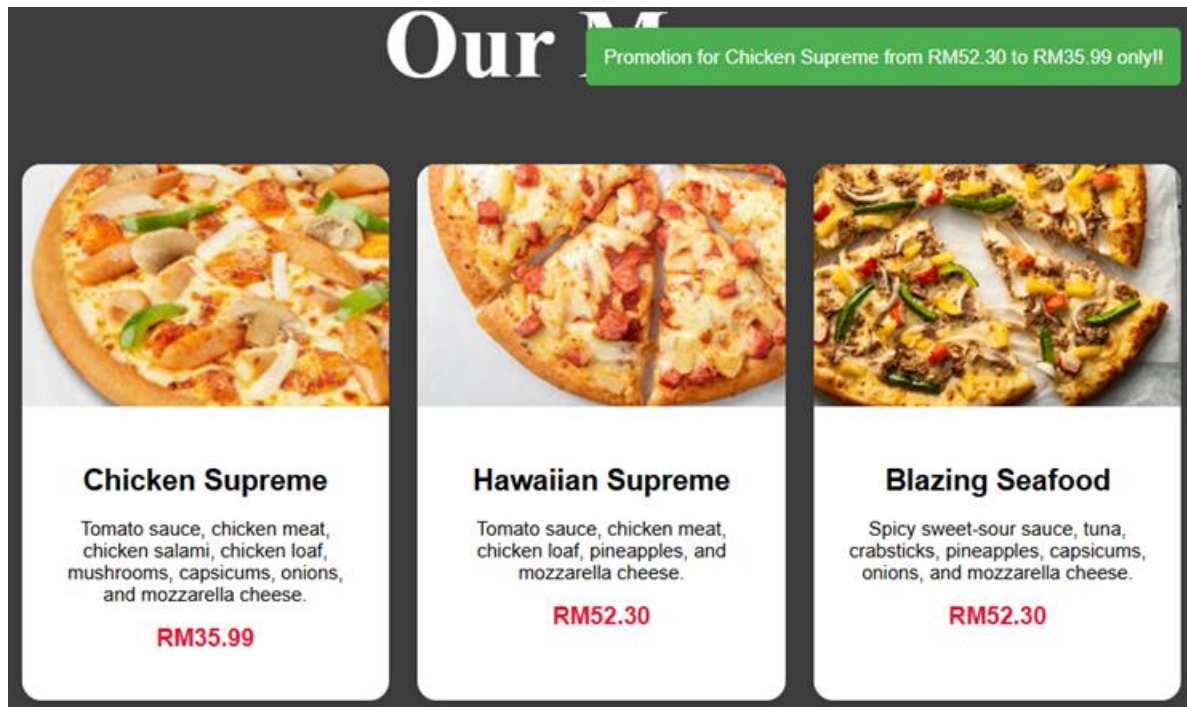


Figure 24. Promotion Notification

How the User Interface works

```
interface Observer {
    public function update($message);
}

class NotificationManager implements Observer {
    public function update($message) {
        // In a real application, this could send emails, push notifications, etc.
        echo "<div class='notification'>$message</div>";
    }
}
```

The PizzaSubject class acts as the subject that maintains a list of observers and notifies them whenever there is a change in the pizza price. The NotificationManager class implements the Observer interface and is responsible for displaying notifications. The Observer interface defines the update method that all observers must implement. The NotificationManager class implements this interface to receive updates.

```

class PizzaSubject {
    private $observers = array();

    public function attach(Observer $observer) {
        $this->observers[] = $observer;
    }

    public function detach(Observer $observer) {
        $key = array_search($observer, $this->observers, true);
        if ($key !== false) {
            unset($this->observers[$key]);
        }
    }

    public function notify($message) {
        foreach ($this->observers as $observer) {
            $observer->update($message);
        }
    }
}

```

The PizzaSubject class maintains a list of observers and provides methods to attach, detach, and notify them.

```

class Pizza {

    public function notifyPriceChange($newPrice) {
        $oldPrice = $this->price;
        $this->price = $newPrice;
        $this->subject->notify("Promotion for {$this->name} from RM{$oldPrice} to RM{$newPrice} only!!");
    }
}

```

The Pizza class integrates with the PizzaSubject to notify observers of price changes.

The UI shows notifications when a pizza's price changes. When a price update occurs, the notification appears in a designated section of the page, informing the user about the promotion. This is achieved by calling the update method in the NotificationManager class.

4.4.4 Template Pattern

The Template Pattern is a design pattern that defines the **skeleton of an algorithm** in a base class, allowing specific steps of the algorithm to be implemented by subclasses or through reusable components.

1. Reusable Structure with Includes

- **header.html** and **footer.html** provide a consistent structure and design across all pages.

- The **business logic** (PHP scripts) is kept separate from the presentation layer (HTML), improving code organization and reusability.
- **mysqli_connect.php** serves as a central file to manage the database connection, avoiding redundant connection code across multiple scripts.
- Example of Reusable Structure:

include 'includes/header.html'; // Common header for all pages

include 'includes/mysqli_connect.php'; // Database connection

2. Business Logic as the Template Core

The **business logic** acts as the "template method" in the Template Pattern. It defines the steps for handling user input, database interaction, and error handling. For instance, in the login process:

1. Validate the user input.
2. Check the database for a matching user.
3. Verify the password.
4. Redirect the user based on their role (admin or regular user).

Example (Login Process):

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    try {
        $dbc = DatabaseConnection::getInstance()->getConnection();
        $username_or_email = $_POST['username'];
        $password = $_POST['password'];

        if (empty($username_or_email)) {
            throw new InvalidArgumentException('Please enter your email/username.');
```

The process always follows the same structure:

Input Validation → Database Query → Error Handling → Success/Failure Response.

3. Template-Like UI Design

In the presentation layer, your **HTML and CSS** follow a structured and reusable template for the user interface. The **signup and login forms** share similar designs, using consistent form elements and validation error displays.

signup.php:

```
<div class="input-field">
  <label for="username">Enter your username</label>
  <input type="text" id="username" name="username" value="<?php echo htmlspecialchars($username ?? ''); ?>"
  <br><span style="color:red; font-size: 13px;"> <?php echo $errors['username'] ?? ''; ?></span><br>
</div>

<div class="input-field">
  <label for="email">Enter your email</label>
  <input type="text" id="email" name="email" value="<?php echo htmlspecialchars($email ?? ''); ?>"
  <br><span style="color:red; font-size: 13px;"> <?php echo $errors['email'] ?? ''; ?></span><br>
</div>

<div class="input-field">
  <label for="phone">Enter your phone number<br></label>
  <input type="text" id="phone" name="phone" value="<?php echo htmlspecialchars($phone ?? ''); ?>"
  <br><span style="color:red; font-size: 13px;"> <?php echo $errors['phone'] ?? ''; ?></span><br>
</div>

<div class="input-field">
  <label for="address">Enter your address</label>
  <input type="text" id="address" name="address" value="<?php echo htmlspecialchars($address ?? ''); ?>"
  <br><span style="color:red; font-size: 13px;"> <?php echo $errors['address'] ?? ''; ?></span><br>
</div>

<div class="input-field group">
  <label for="password">Enter your password</label>
  <input type="password" id="password" name="password">
  <br><span style="color:red; font-size: 13px;"> <?php echo $errors['password'] ?? ''; ?></span><br>
</div>
<div><span style="color:red; font-size: 13px;"> <?php echo $errors['existinguser'] ?? ''; ?></span><br></div>
<button type="submit" class="btn btn-submit">Sign Up</button>
```

This structure:

- **Provides user feedback** (error messages in red for missing or incorrect fields).
- **Ensures consistency** across the site (all forms use the same styling).

How the User Interface works

1. Login Page Workflow

The image shows three wireframes of a Login page. Each wireframe has a title 'Login' at the top. Below the title are two input fields: 'Username or Email' and 'Password'. At the bottom is a red 'Login' button. The second wireframe shows an error message 'Email/username not registered. Please sign up first.' below the 'Username or Email' field. The third wireframe shows an error message 'Incorrect password. Please try again.' below the 'Password' field.

- The user enters their **username/email** and **password**.
- The PHP script validates the input and checks the database for the credentials.
- **If successful**, the user is redirected to:
 - **Admin.php** (if they are an admin).
 - **UserMenu.php** (if they are a regular user).
- **If an error occurs**, an error message is displayed below the corresponding input field.

2. Registration Page Workflow

The image shows three wireframes of a Sign Up page. Each wireframe has a title 'Sign Up' at the top. Below the title are five input fields: 'Enter your username', 'Enter your email', 'Enter your phone number', 'Enter your address', and 'Enter your password'. At the bottom is a red 'Sign Up' button. The second wireframe shows validation errors for each field: 'Username is required.', 'Email is required.', 'Phone number is required.', 'Address is required.', and 'Password is required.'. The third wireframe shows a final error message 'Username/Email/Phone number already in use, please try again.' at the bottom.

- The user fills out the registration form with their **username, email, phone, address, and password**.
- The script performs the following checks:
 - Ensures all fields are filled.
 - Verifies that the password is at least 8 characters long.

- Checks for an existing username, email, or phone number in the database.
- **If all checks pass**, the user is registered with a success message and ask user to login with username or email.

4. Reusable Header and Footer

header.html:



headerlogin.html:



Figure 25. Header page

- The header.html file contains the navigation bar and site logo, ensuring a consistent look.

Footer.html:



Figure 26. Footer of the page

- The footer.html includes site-wide footer content (like contact information or links).

5. TESTING

This testing phase involved executing the application using various test inputs to validate its functionality. Below is a table of test cases and results:

5.1 Table of Input and Output Process

Input	Process	Output	Test Data	Status
Sign Up: Enter username, email, phone number, address, password	System validates inputs	Success message: 'Registration successful! You can now log in to start using the system.'	Username: norlela Email: lela@gmail.com Phone: 0123446787 Address: Forest Green Condominium Password: abc@@123	Pass
Sign Up: Missing field	System validates form inputs	Error message: 'Email is required!'	Username: norlela Email: (empty) Phone: 0123446787 Address: Forest Green Condominium Password: abc@@123	Pass
Sign Up: Existing user	System checks for username/email/ phone duplication	Error message: 'Username/Email/ Phone number already in use, please try again.'	Username: norlela Email: lela@gmail.com Phone: 0123446787 Address: Forest Green Condominium Password: abc@@123	Pass
Sign Up: Password less than 8 characters	System validates password requirements	Error message: 'Password must be at least 8 characters.'	Username: naim Email: naim@gmail.com Phone: 0123442345	Pass

			Address: Forest Green Condominium Password: abc	
Login: Valid user (customer)	System validates username, password and role	If successful, proceed to Menu page	Username: norlela Password: abc@@123	Pass
Login: Missing field	System validates form inputs	Error message: 'Please enter your email/username.'	Username: norlela Password: (empty)	Pass
Login: Invalid Login (Incorrect Password)	System validates credentials	Error message: 'Incorrect password. Please try again.'	Username: norlela Password: abc	Pass
Login: New user	System validates username and password	Error message: 'Email/username not registered. Please sign up first.'	Username: user Password: abc	Pass
Login: Valid user (admin)	System validates username, password and role	If successful, proceed to Admin Dashboard	Username: najmi Password: abc@@123	Pass
Select a menu from the menu page	System redirects to the variation selection page	Menu customization page appears	Click "Chicken Supreme"	Pass
Select pizza size	System registers selection	The selected size is highlighted, and	Click "Large"	Pass

		the price is updated		
Select crust type	System registers selection	The selected crust is highlighted, and the price is updated	Click "Thin Crust"	Pass
Choose quantity: Select "2" as the quantity	System registers the selected quantity	The correct quantity is displayed, and the price is updated	Quantity: 2	Pass
Successful addition to cart	System adds pizza to cart with selected variations	Cart displays correct menu details	Click "Add to Cart"	Pass
Add item to cart	System adds selected item to cart	Cart displays the correct menu, quantity, and price	Click "Add More Items"	Pass
Remove single item from cart	System deletes the selected item from the cart	Item disappears from the cart, and total price updates	Remove "Chicken Supreme"	Pass
Proceed to checkout	System validates cart and order details	Order summary page loads successfully	Click "Proceed to Checkout" button Cart: (Not empty)	Pass
Select delivery method: Delivery or Pickup	System registers selection	The chosen method is highlighted	Select "Delivery"	Pass
Enter delivery address	System saves and displays address	Address appears in checkout details	Delivery Address: Forest Green Condominium	Pass
Select payment method	System registers selection	Payment method is displayed correctly	Select "Credit Card"	Pass

Apply redeemed reward and proceed to checkout	System deducts points and updates total	Discount applied, points reduced, order placed	Available Points: 500 Select “Free Delivery (500 points)” Click “Redeem Selected Reward”	Pass
Print receipt	System sends receipt to printer	Printer prints order receipt successfully	Click "Print Receipt" button	Pass
Send feedback	System validates form inputs	Success message: ‘Feedback submitted successfully.’	First Name: Norlela Last Name: Salleh Email: lela@gmail.com Phone: 0123446787 Write your message: okay	Pass
Send feedback: Missing field	System validates form inputs	Error message: ‘Please fill out this field.’	First Name: Norlela Last Name: Salleh Email: lela@gmail.com Phone: 0123446787 Write your message: (empty)	Pass
Admin Dashboard: Sorting sales by product reports by total revenue	System sorts sales reports by the total revenue	Reports are displayed from highest revenue	Select "Total Revenue"	Pass
Admin Dashboard: Sorting sales by product reports by product name (A-Z)	System sorts sales reports by product name in alphabetical order	Reports are displayed in alphabetical order	Select "Product Name (A-Z)"	Pass
Admin Dashboard: Sorting all sales reports by date	System sorts sales reports by date	Reports are displayed in chronological order	Select "Latest to Oldest (Sales Date)"	Pass

Admin Dashboard: Sorting all sales reports by sales ID	System sorts sales reports by sales ID	Reports are displayed increasing order	Select " Sales ID"	Pass
--	--	--	-----------------------	------

6. MAINTENANCE

6.1 Discussion

The maintenance phase focused on repairing existing issues and extending the system with additional functionalities. Several improvements were made to the current modules, including enhancements in login and sign-up to let users navigate easily once signed up. Users can sign in using email or username, providing more flexibility. The cart functionality was also enhanced by making it more visible and allowing users to modify their selections easily. Users can now change their cart by removing items before proceeding to checkout. Initially, the system lacked a streamlined payment process. The new update ensures a smooth checkout flow, allowing users to choose different payment methods and delivery or pickup options.

To further enhance the system, new modules, like a reward program, will award points to customers for their purchases, and a feedback system will be created where users can submit reviews. Users can now earn points for every purchase and redeem them for discounts. The system tracks loyalty points, displaying the total in the user profile. These enhancements are expected to improve customer engagement and satisfaction.

6.2 Benefits of the Project

The benefits of this project include:

- Improved Efficiency

The system ensures a streamlined ordering process and faster checkout. Users can browse the menu, customise their pizza, and add items to the cart in just a few clicks. The system automatically calculates the total price, including applicable taxes, discounts, and delivery charges, ensuring transparency and accuracy for faster checkout.

- **Enhances User Experience**

Enhanced navigation and user-friendly design make it easier for customers to place orders. The menu is visually appealing and easy to browse, with high-quality images, detailed descriptions, and pricing for each item. The checkout process is straightforward, guiding users step-by-step to select their delivery method, enter payment details, and confirm their order.

- **Higher Customer Retention**

The system includes a reward program, where customers earn points for every order placed. Points can be redeemed for discounts, free items, or special promotions, giving customers an incentive to return. This approach increases customer engagement and encourages habitual ordering through the platform.

- **Better Sales and Business Insights**

Detailed sales reports and insights into users are provided to the administrator for better business insights. Sales reports can be filtered by date range or specific menu items, allowing managers to identify peak ordering times and high-demand products.

6.3 Limitations of the Project

Despite its advantages, the system has some limitations that may require future improvements:

1. **Internet Dependency**

Since the system is web-based, it depends on a stable internet connection, making it inaccessible in areas with poor connectivity.

2. **Regular Maintenance Requirements:**

To ensure the system remains secure and functional, regular updates and maintenance are necessary.

3. **Scalability Issues**

As the number of users increases, the system may require database optimization and server upgrades to handle high traffic.

6.4 Recommendations

To further improve the system, the following recommendations should be considered:

1. Integration of Additional Payment Methods

Adding options like PayPal, Google Pay and Apple Pay will make the system more flexible, convenient and accessible to a wider audience.

2. Development of a Mobile Application

A dedicated mobile app for iOS and Android will allow customers to place orders conveniently from their smartphones.

3. AI-Powered Personalized Recommendations

Implementing machine learning algorithms to suggest menu items based on customer preferences can increase sales and improve user experience.

6.5 Conclusion

The Deli Pizza System successfully enhances the user experience by introducing valuable new features. Improvements in login and user authentication, cart functionality, and checkout ensure a smooth ordering process. The addition of a reward program and a feedback system increases customer engagement. Although the system has some limitations, it remains a highly efficient and user-friendly platform for both customers and restaurant managers. Ongoing updates and enhancements will ensure its continued success in meeting user needs efficiently.

7. REFERENCE

- i. Visual Paradigm. (n.d.). *What is use case specification?* Visual Paradigm. Retrieved from <https://www.visual-paradigm.com/guide/use-case/what-is-use-case-specification/>
- ii. PHP Manual. (n.d.). *Exceptions*. PHP.net. Retrieved from <https://www.php.net/manual/en/language.exceptions.php>
- iii. Visual Paradigm. (n.d.). *What is use case diagram?* Visual Paradigm. Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
- iv. System Architecture - Detailed explanation. (2023, June 22). InterviewBit. <https://www.interviewbit.com/blog/system-architecture/>
- v. How to draw 5 types of architectural diagrams. (2021, April 5). Lucidchart. <https://www.lucidchart.com/blog/how-to-draw-architectural-diagrams>

8. APPENDIX

8.1 Minutes of Meeting 1

Date: 20/12/2024

Time: 10.30 a.m.

Place: Microsoft Teams

Attendees:

- Nur Afikah Binti Noor Azman
- Nor Mirza Afiqah Binti Abdul Rahman
- Nur Nisa Rafiah Binti Ramlee
- Nur Hamizah Zahirah Binti Nor'azman
- Nur Sazahah Binti Salauddin

Meeting Points:

- Explanation on the current system version by Nisa Rafiah.
- Discussion on enhancement and new modules for new system.
 - Mirza suggested for a cart icon which is not yet developed in the current system.
 - Some members suggested for a validation error message for all input fields and adding more details on the sign-up process.
 - Members also suggested for making an option feature for pickup or delivery and time for a delivery.
 - Members also pointed out that our current system lack of feedback page and smooth flow operation from one page to another.
 - Sazahah then suggested if possible to include order tracking, language switching and notification for an order placed.
 - Afikah take notes of all suggestion including suggesting for a loyalty points system and asked for a final decision on the matter.
 - Members unanimously agreed for validation messages, cart icon, loyalty points, details sign up and order process.
- Discussion on mini project proposal tasks arrangement.
- Hamizah notified all members that she had sent in the tasks division in the WhatsApp group chat and asked all team members to put in their names for their desired tasks.
- Meeting adjourned at 12 p.m.

8.2 Minutes of Meeting 2

Dates: 19/1/2025

Time: 9.00 p.m.

Place: WhatsApp Group Chat

Attendees:

- Nur Afikah Binti Noor Azman
- Nor Mirza Afiqah Binti Abdul Rahman
- Nur Nisa Rafiah Binti Ramlee
- Nur Hamizah Zahirah Binti Nor'azman
- Nur Sazahah Binti Salauddin

Meeting Points:

- Nisa presented on the storyboard that is submitted on 18/1/2025.
- Afikah started the discussion of tasks division for system coding.
- List of website pages needed to be improved and implemented is showed to the members.
- Nisa points out that loyalty points page is missing and also suggested for a profile icon menu to navigate to profile page and rewards page that displays user's accumulated points.
- Members suggested to update the storyboard to include new profile page and rewards page.
- Nisa takes on the role to re-draw the storyboard.
- A new list of the system pages is presented, and members chose their desired tasks.
- Mirza also suggested for a change in programming language from C# to php as the implementation is easier for our current knowledge.
- Members unanimously agreed on that matter.
- Discussion on the presentation parts' division started.
- A list of tasks for presentation is sent to the members and members were asked to choose their desired ones.
- Members unanimously agreed that Mirza will take on the role of sharing the screen during the presentation.
- Sazahah takes on the role to make the Canva slides for the presentation and the links is sent to the group chat.
- Meeting adjourned at 1.00 a.m.