APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) 631-648
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

CampusEats: The Student Food Hub

Muhammad Haziq Sumagi[1], Mohd Hamdi Irwan Hamzah[2]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, Malaysia

*Corresponding Author: hamdi@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.034

Article Info

Received: 15 June 2025
Accepted: 21 June 2026
Available online: 30 June 2026

Keywords

Food Ordering system, Object Oriented Approach, Web Based

Abstract

CampusEats is a web-based food ordering system developed to enhance the food purchasing experience within campus environments, particularly at KKTM Seri Gading. Traditionally, students and staff face long queues, limited accessibility to food owner information, and inefficient ordering processes. This project addresses these challenges by offering a centralized platform that enables users to browse menus, place orders, and provide feedback conveniently online. The main purpose of CampusEats is to simplify and streamline food ordering for students, food owner, and the campus community. The system was developed using the Software Development Life Cycle (SDLC) methodology and object-oriented programming principles. Laravel was used as the backend framework, incorporating features such as a payment gateway, customer feedback, and management. The result is a dynamic and user-friendly system that reduces waiting times, improves service efficiency, and supports digital transactions. CampusEats demonstrates its potential to significantly improve the overall food ordering experience on campus. In conclusion, this system serves as an effective digital solution to modernize food ordering and strengthen the connection between students and local food vendors.

1. Introduction

This CampusEats System is being built to replace the old and conventional ordering system that most food shops use with a new and more efficient ordering system that is more user-friendly. As a result of the high manual effort required by the old ordering method, both employees and consumers experience frustration. Manual taking orders performed by the employees will result in certain human mistakes, such as providing the wrong food to the customers, having ugly handwriting on the waiter's menu, and placing the wrong order in the proper sequence. These human faults will make the consumer dissatisfied with the food shop. As a result, CampusEats will be developed to assist the food shop in improving its administration and managing food stall operations.

The proposed CampusEats System seeks to eliminate these issues by introducing a user-friendly digital platform that streamlines the ordering process. This system will allow customers to place orders through the website, ensuring accuracy and reducing human error associated with manual entry. Besides, the demand for the online ordering system is increasing. By 2022, the food delivery market size is expected to grow to an annual revenue if USD$956M, one of the fastest-growing sectors in the market. [1]. It will feature real-time updates for meal preparation times, enabling customers to plan their schedules more effectively. Additionally, the system will facilitate easy modifications and cancellations of orders directly through the website, it will enhancing customer satisfaction and reducing food waste. By integrating a comprehensive database and management tools for vendors, CampusEats aims to create a more efficient and enjoyable dining experience for students and the campus community. Besides, this CampusEats also a system where users can shop for essential daily items. This platform allows vendors registers and showcase their business, whether it's a food stall or a

store selling daily necessities. Thus, it making it easier for students to access these services in a platform. CampusEats aims to make it easier for students and the campus community. For the project, there will be three objectives to accomplish. The following objectives for developing CampusEats are as follows:

i. To design a CampusEats system using object-oriented approach
ii. To develop CampusEats system using a web-based approach.
iii. To test the developed system using User Acceptance Testing.

The paper is structured as follows: Section 2 goes into a literature review of related work and existing applications. Following this, Section 3 outlines the methodology for development. Afterwards, Section 4 presents implementation and testing. and section 5 presents the conclusion. Finally, there are references and appendices for additional notes.

2. Related Work

This Section discusses on literature review that had been conducted for the system and the current system that is being used by CampusEats. Section 2.1 will cover about the implementation of using Laravel's framework. Section 2.2 will covers about e-commerce system. Then, Section 2.3 focuses on Web-based application review, while Section 2.4 discusses three types of existing systems that are studied and compared with the developed system.

2.1 Framework Laravel

Laravel was developed and founded by Taylor Oatwell in July 2011. It was developed more than 5 years after CodeIgniter was released [2]. It comes with a powerful tools and provide application architecture. Laravel is a PHP based web-framework for developing. It is for creating advanced web applications with elegant syntax. Furthermore, it incorporates a variety of technologies such as ASP.NET MVC, CodeIgniter, and Ruby on Rails, and much more. This framework is an open-source framework. It saves developers a lot of time and helps them reduce the thinking and planning to develop the whole website from scratch. Additionally, the application's security is handled by Laravel. Hence, all its features can help the developer to create a stunning website.

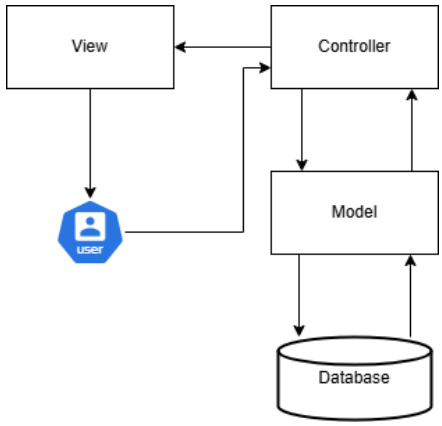

Fig. 1 *The concept of MVC in Laravel's Framework*

Fig 1 illustrates the model-view-controller (MVC) architectural pattern. MVC, as the name suggests, is a software architecture that separates domain from the rest the user interface. It does separate the application into three parts, which are the model, the view, and the controller. The model manages the behaviors and data of the application, such as the implementation of the Database. It is about the data management of the application. Beside, the view can be as the interface of the application. It will process the data from the model into a form that is suitable for the user interface. In addition, the controller is like a method or function. This controller works as the function that can be used for the application. It receives user input and makes calls to model objects to perform as intended actions.

2.2 E-Commerce System

The primary related concept is e-commerce platforms, as CampusEats enables users to order products or food online, manage their shopping cart, and process payments. Concepts like shopping cart functionality, product catalog management, user reviews, and customer feedback are part of the e-commerce system. This feature includes a shopping cart, order processing, product catalog, and booking. According to Annaraud and Berezina,

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

[3], when customers order food online, their experience with the entire process is influenced by the experience with the website of the online food ordering [3].

2.3 Web-based application(Web-App)

The current rise of technology, web applications have significantly evolved, The World Wide Web was once seen as a static repository of information accessible to many users [4]. However, that perception has changed rapidly over time. Today, web-based applications operate on a client-server architecture, where the client refers to the device or application requesting information from a web server typically, this means the user's web browser.

2.4 Comparison of System

Table 1 illustrates these common and unique features, showcasing that all systems offer login and order processing, adding and displaying cart items, and a product catalog. However, CampusEats, does not offer branch availability, reflecting its campus-centric approach.

Table 1 *System's comparison between each features*

Features/System	Myfave	popmeals	foodpanda	CampusEats
Login System	√	√	√	√
Process Order	√	√	√	√
Add item to cart	√	√	√	√
Display cart item	√	√	√	√
Search Function	√	X	√	√
Branches Availability	√	√	√	X
Estimated Arriving Time	√	√	√	√
Display Product Catalogue	√	√	√	√

Table 1 compares the core features of four food ordering systems: MyFave, PopMeals, FoodPanda, and CampusEats. While all platforms provide basic functionalities such as user login, order processing, cart management, and access to product catalogs, they primarily cater to urban or city-based markets. These existing systems often include features like branch availability and wide geographic coverage, which, although beneficial in larger commercial areas, are less relevant and may add unnecessary complexity for a campus-focused environment

Moreover, most of the compared systems are not tailored for students' specific daily routines and needs, such as limited meal hours, budget constraints, or campus-based promotions. These platforms also lack features that support localized vendor management within a confined community like a university.

CampusEats addresses these gaps by eliminating unnecessary features such as branch selection and introducing campus-relevant modules, such as streamlined ordering method, and simplified UI for quick ordering . Thus, the need for CampusEats arises from the lack of a system specifically designed to support food ordering and management within a campus.

3. Methodology

A methodology is a systematic approach and more organized strategy used to develop, improve, verify, and maintain software or systems. It offers a set of ideas to optimize the learning experience to develop the CampusEats system. This section explains how the use of the Agile model is applied in this project, detailing the activities carried out in each phase to ensure the system meets its objective effectively.

System Development Life Cycle (SDLC) model is the process of developing the system with proper analysis, design, implementation and maintenance to improve the quality of the system [5]. In this project, because of the deadline is already known, there might be changes due to feedback of stakeholder and to align with user needs. SDLC model is shown in Fig 2.

Fig. 2 *SDLC Model [8]*

Based on fig 2 above, it provides a clear explanation of the phases of SDLC. A clear visual representation of it must start from planning until deployment. Each of the tasks will be explain in more details in table 2 as follows.

Table 2 *Software development activities and corresponding task*

Phase	Task	Output
Planning & Requirement Analysis	1) Proposed the project 2) Determine the project schedule, activities and output.	1) Project proposal 2) Develop Gantt chart 3) Literature review on comparing the existing system with the proposed system.
Defining the requirements And designing	1) Brief discussion with the stakeholder to collect some information about the current traditional method of ordering food. 2) To figure out the software requirement.	1) User requirements. 2) Hardware and software requirements. 3) Use Case Diagram, Activity Diagram, Sequence Diagram, Class diagram
Development Phase	1) Code the system based on the requirement gathered. 2) Developing interface that is easy to understand.	1) System code based on the requirements 2) Connecting to the Database. 3) Planning on the coding management, fixing any errors.
Testing	1) Finding the defect if any defect in the system. 2) Testing every interface, functional testing.	1) Provide report defects are found, checking on module for the defects. 2) To make sure quality of the system.
Deployment	1) Show the stakeholder for the complete system. 2) Provide user training to the stakeholder. 3) Ensure system is free from errors.	1) System operational in good condition. 2) User guideline to use the system.

Table 2 gives a clear picture of the software development process and the tasks involved at each stage. It breaks down the project into steps such as planning and requirement analysis, defining the requirement and designing, development phase, Testing, and deployment. It will make it easier to understand how everything fits together. Each activity is matched with specific tasks to keep the process organized and on track. This detailed approach ensures that everyone involved knows what needs to be done.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

3.1 Analysis and Design

This chapter discusses the analysis and design in the development of the proposed system which is CampusEats food ordering system. This system requirement analysis, which includes user requirements, hardware and software requirements, and functional and non functional requirements.

The CampusEats food ordering system adopts an Object-Oriented Programming (OOP) approach, where the system is developed during the analysis and design phases. This development process is later represented through OOP diagrams, including the Use Case Diagram, Sequence Diagram, Class Diagram, and Activity Diagram, which collectively serve as representations of the database systems.

3.2 User Requirements Analysis

User requirement analysis bridges developers and users by identifying needs from traditional food shop methods before modernizing with CampusEats. Section 4.1 ensures the system addresses user needs and traditional shortcomings for project success.

Table 3 *User Requirement*

No	User Requirements
1	Admin, Manager and Customer should be required to have a registered account with a valid ID and password to access the system.
2	Customer should be able to browse the menu, place orders, do searching, and edit their information such as name, address, number phone and others.
3	The Manager efficiently handle the order that submit by customers.
4	Manager should be able to manage order, manage shop, manage menu of the system.
5	Admin should be able to generate reports, including sales, and customer preferences.
6	Customer should be able to make payments for their orders.
7	Manager should be able to see orders make by the customer.
8	Manager should be able to generate receipts for completed orders.
9	Manager should be able to create, read, update, and deletion of food menu.

Based on table 3 the user requirements for the system are as follows: Admin, Manager, and Customer must register with a valid account to access the system. Customers can browse menus, place orders, search, and update personal details. Managers handle customer orders, manage the shop, and oversee the menu. Admins generate reports on sales and customer preferences, while Managers can view orders, generate receipts, and manage the food menu, including creating, updating, and deleting items. Customers can also make payments for their orders.

3.2.1 Functional Requirements

A functional requirement is the ability of a system to perform a specific modules. In CampusEats there will be 8 modules in total. The functional requirements for the CampusEats are shown in Table 4 as follows.

Table 4 *Functional Requirement of CampusEats*

No	Module	Description
1	User Login	• Streamlined the process of user authentication. • Login & Registration • Password Change
3	Manage Menu	• Customers can submit orders, track order status, and manage their orders. • Shipping information fetched from user profile. • Managers or food stall owners can perform CRUD operations for the menus.

Table 5 *Continued Functional Requirements*

No	Module	Description
3	Manage Promotion	• Managers or food stall owners can perform CRUD operations for the promotion. • Managers or food stall owners can manage promotion, edit promotion on desired menu.
4	Manage Order	• Customers are able to view menu, perform order, and can look for the receipt. • Manager can view customer's order, update status, and confirm order made by the customer.
5	Manage Payment	• Manager is able to oversee the payment made by the customers • Customers are able to choose choices of payment options.
6	Manage Sales Report	• The CampusEats manager can see the dashboard based on popular products, total sales, total spender, and all about the performance of the system sales.
7	Manage Feedback	• Customers can provide feedback on buying product (After Customer complete the ordering until O_Status=5;= "picked up". • Managers or food stall owners can view customer feedback and make improvements based on it.
8	Manage System	• Admin able to oversees the system, user , and system setting. • Admin able to sees the system logs, payment, and login logs. • Admin are able to manage setting for payment, Frequently Asked Questin, Email Notification, Term and Condition and overall system for CampusEats.

3.3 Use Case Diagram

A use case diagram is the representation of the overall system. The main purpose of a use case diagram is to visualize use cases in the system [6]. It consist of few elements such as actors, use cases and relationships. The primary actors is usually on the left side of the use case diagram. The use case diagram not only shows single use case, but a series of use cases for a given system, and each use case can be described in sequence diagram, and activity diagram. The use case diagram combine both UML notation and its narrative text.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

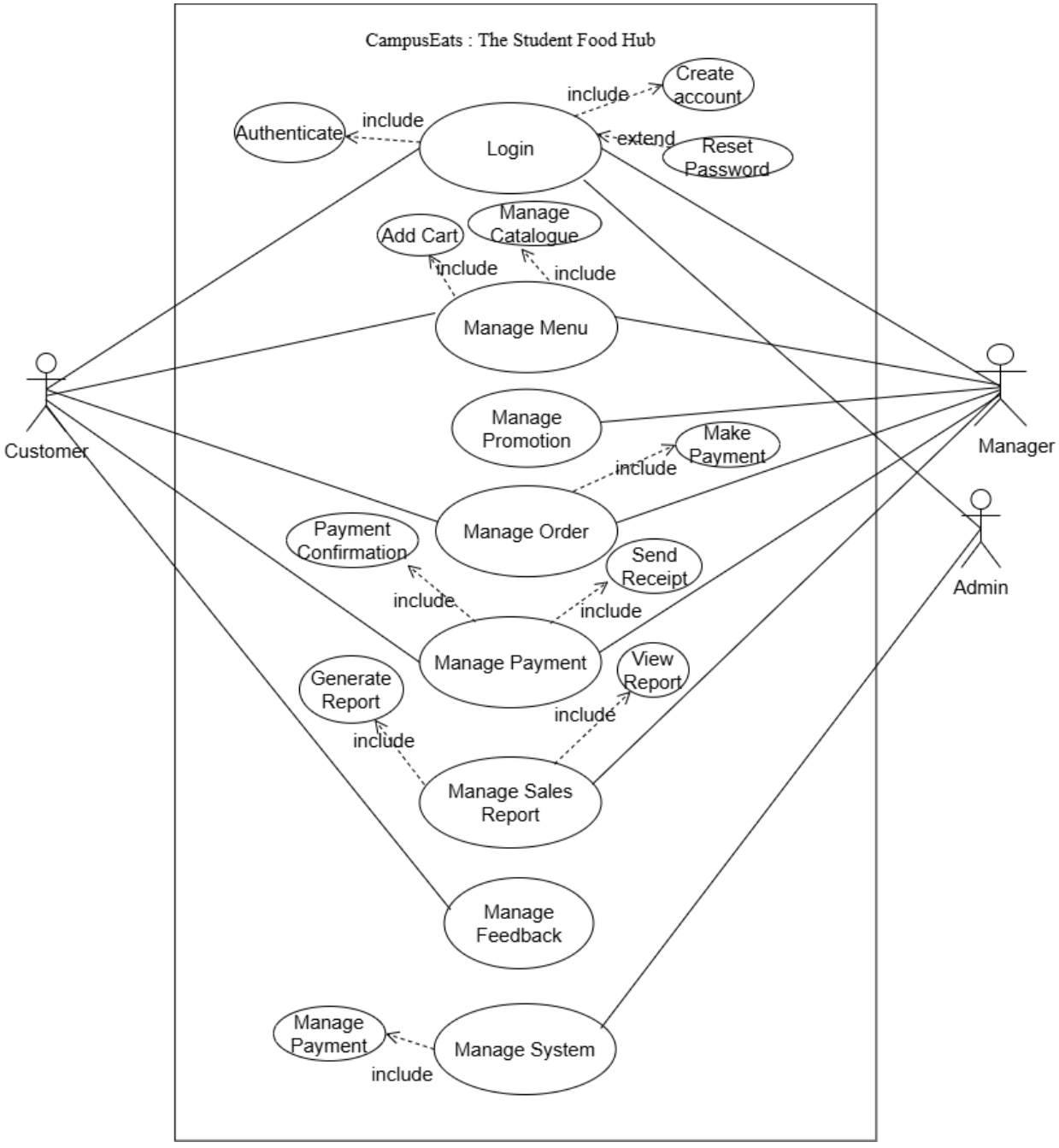

Fig. 3 *Use Case Diagram*

Based on Fig 3, the system involves three key actors: the customer, the admin, and the manager. Among these, customers play a crucial role in the system. To access the system, customers are required to register before proceeding to place an order. They actively participate in order management by browsing the menu to create new orders, tracking the progress of their orders, and confirming their order statuses.

3.4 Sequence Diagram

Before accessing the system, the customer must enter their email and password. Once logged in successfully, they can perform activities such as ordering food, manage profile and others. The customer has the option to log out of their account when they are done with their activities. Fig 4 represents the sequence diagram for the customer manage order. Customer submit order while manager oversees the order and update the order status for the customer. Then the Customer will be able to see their order status based on the update made by the manager.

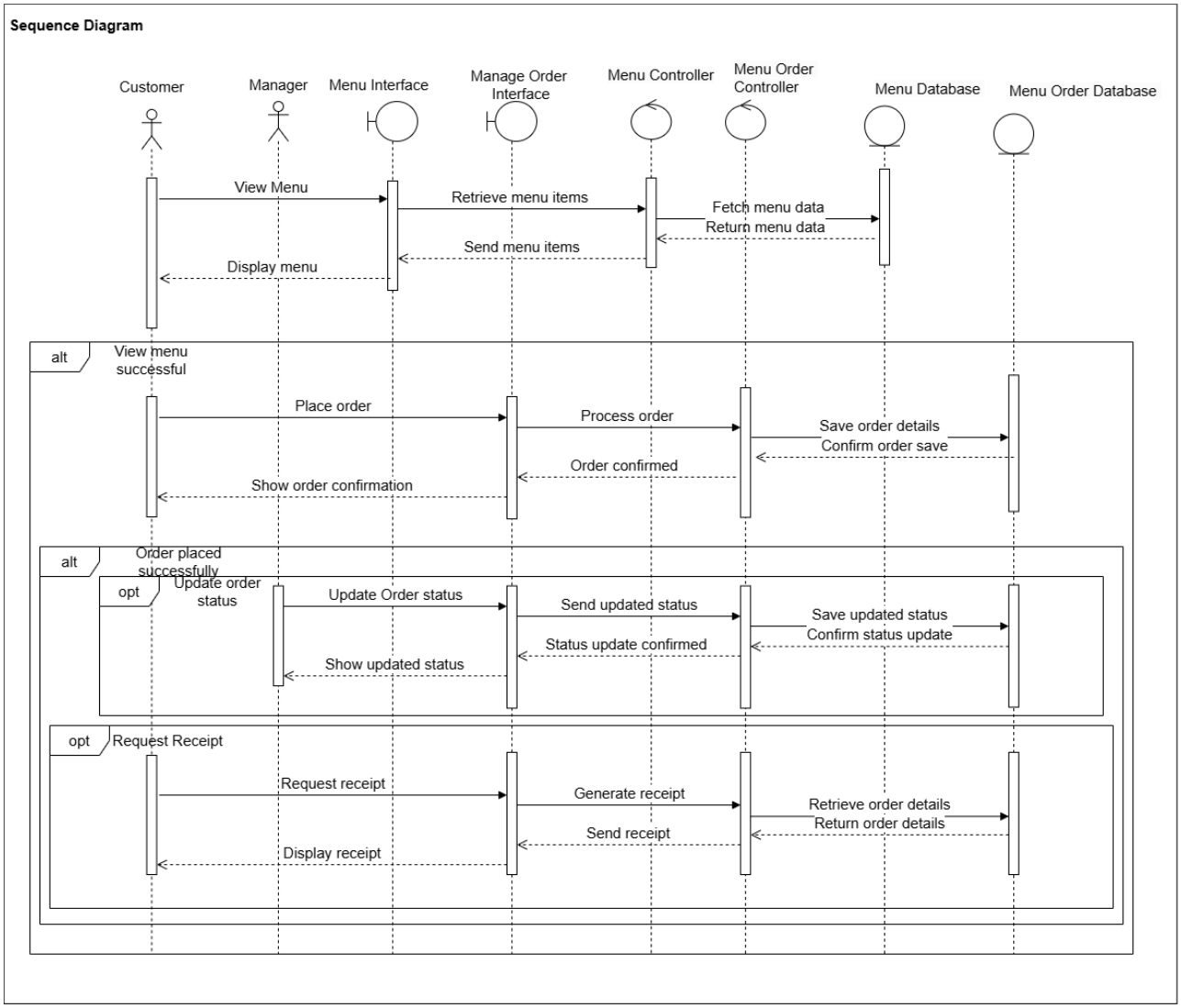

Fig. 4 *Sequence Diagram for manage order*

3.5 Class diagram

The class diagram is the most commonly used diagram to illustrate, design, and specify how the different entities in a software system relate between them [7]. Class diagram are crucial components in the development of this system. It also shall provide additional clarification regarding the class name, attribute list, function list, and relationship types that were implemented during the development of CampusEats.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

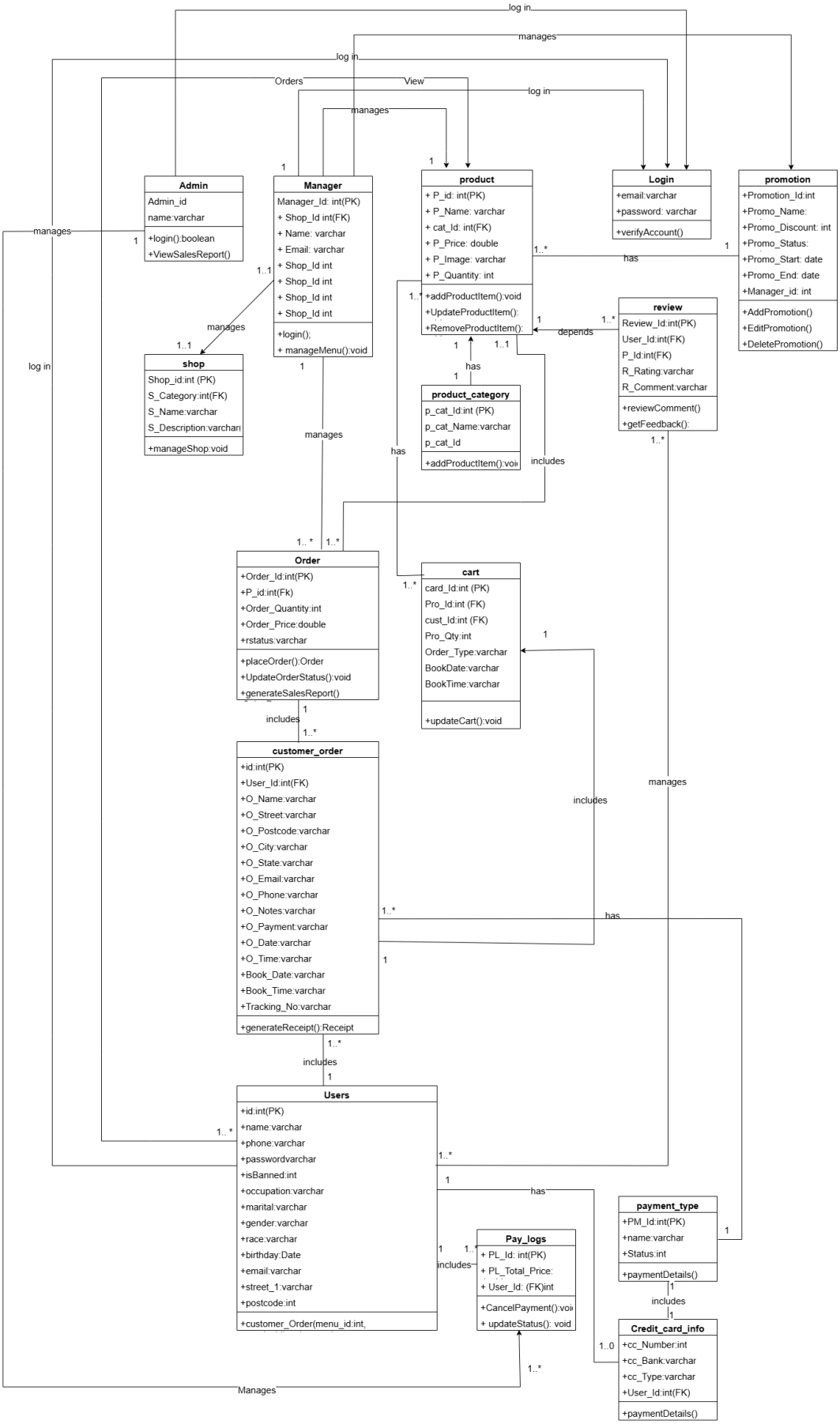

Fig. 5 *Class diagram of CampusEats*

Fig 5 shows use case diagram for CampusEats. This class diagram is the overview of how the system will interact with the system and the various functionalities it offers.

3.6 System Design

System design present the Interface design of the system and functionality. Section 4.4 will outlines the overview of the system interface.

3.6.1 Interface Design

The design of the system as in Fig 6, Fig 7, Fig 8, Fig 9, and Fig 10.

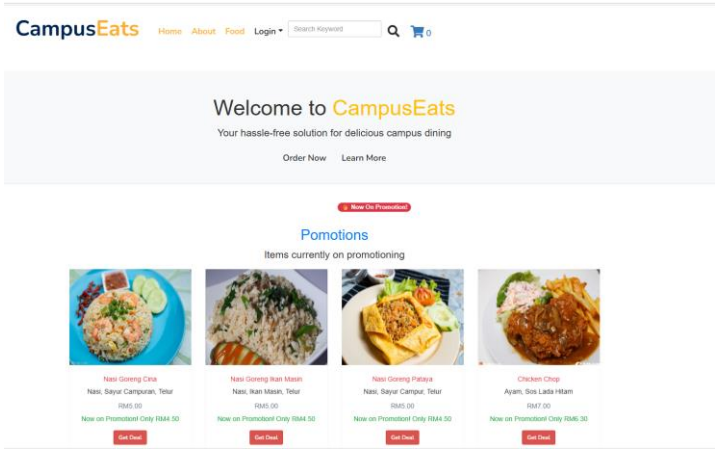

Fig. 6 *Homepage Interface*

Fig 6 shows the homepage with a simple navigation menu (Home, About, Login, Search) and a dynamic cart icon. A central slideshow displays dish images to encourage menu exploration.

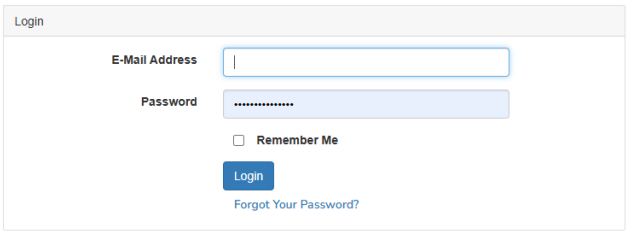

Fig. 7 *Login Interface*

Fig 7 shows the login interface with fields for email, password, and a login button. A "Remember Me" checkbox allows users to save credentials for future sessions.

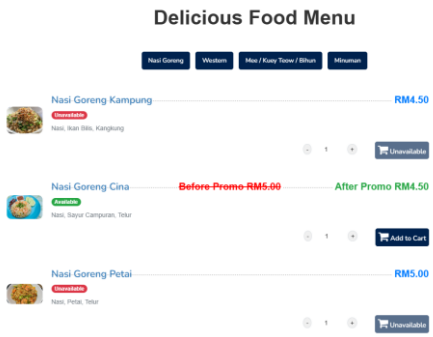

Fig. 8 *User choose food Interface*

Fig 8 shows the interface for the menu. The menu is categorized (e.g., Nasi Goreng, Western, Mee) with images for easy identification. Users can adjust quantities with plus/minus symbols, and the textbox updates in real time.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

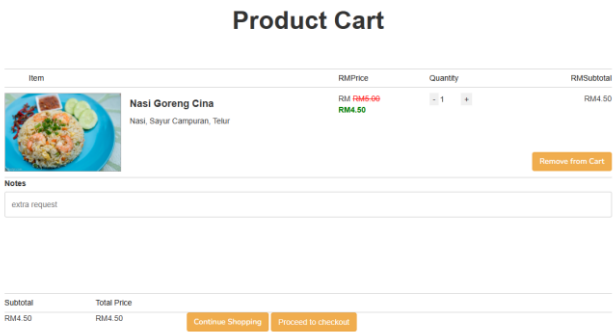

Fig. 9 *Cart Interface*

Fig 9 shows the CampusEats cart interface, allowing users to view updates on their menu selections.

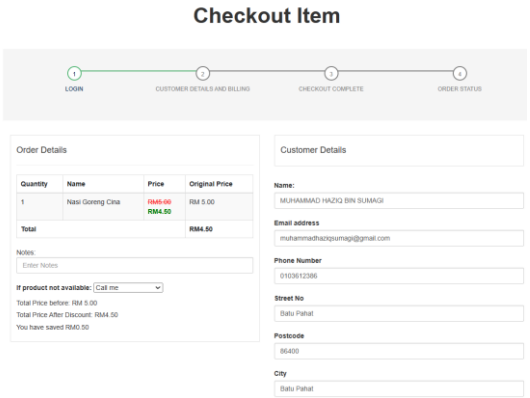

Fig. 10 *Checkout Interface*

Fig 10 shows the checkout interface of CampusEats. Users can input the required fill such as customer details. Users also have to choose payment options.

4. Implementation and testing

CampusEats consists of various modules that work together to provide a seamless user experience. During the implementation phase of system development, the coding of each main module is presented to demonstrate its integration with the database and how it supports the system's goals. The implementation highlights how each module interacts with the database to ensure that data is properly stored, retrieved, and updated. Fig below is one of the modules in CampusEats. In this section, the example of the primary module is outlined to give an overview to the system.

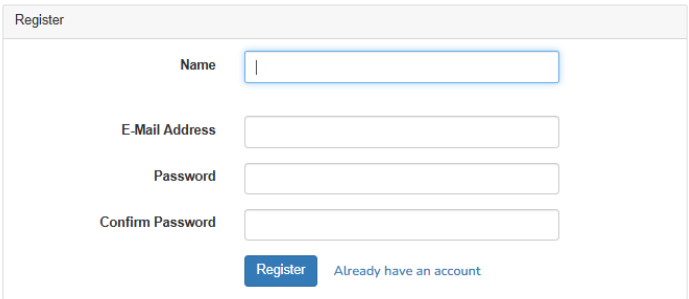

Fig. 11 *Interface Registration for the user*

```
<div class="card">
    <div class="card-header">{{ __(key: 'Register') }}</div>

    <div class="card-body">
        <form method="POST" action="{{ route(name: 'register') }}">
            @csrf

            <div class="form-group row">
                <label for="name" class="col-md-4 col-form-label text-md-right">{{ __(key: 'Name') }}</label>

                <div class="col-md-6">
                    <input id="name" type="text" class="form-control @error('name') is-invalid @enderror" name="name" value="{{ old(key: 'name') }}" required autocomplete="name" autofocus>

                    @error('name')
                        <span class="invalid-feedback" role="alert">
                            <strong>{{ $message }}</strong>
                        </span>
                    @enderror
                </div>
            </div>

            <div class="form-group row">
                <label for="email" class="col-md-4 col-form-label text-md-right">{{ __(key: 'E-Mail Address') }}</label>

                <div class="col-md-6">
                    <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old(key: 'email') }}" required autocomplete="email">

                    @error('email')
                        <span class="invalid-feedback" role="alert">
                            <strong>{{ $message }}</strong>
                        </span>
                    @enderror
                </div>
            </div>
```

Fig. 12 *Coding for registration user*

Fig 11 is the interface for registration user. Fig 12, represents a user registration form in a Laravel-based web application, using Blade templating and Bootstrap for layout and styling. It includes fields for the user's name and email address, wrapped within a card layout for better visual structure. The form uses the POST method and submits data to Laravel's predefined register route, which handles the user registration logic. To protect the form from cross-site request forgery attacks, Laravel's @csrf directive is used. Each input field is set up to display validation feedback using Laravel's @error directive. If the validation fails, the old input value is retained with the old() function, and an error message is shown beneath the field with a red border to indicate the issue. Labels are aligned using Bootstrap's grid system, and the form is designed to be accessible and responsive. The {{ __('Register') }} syntax is used to support multi-language translation

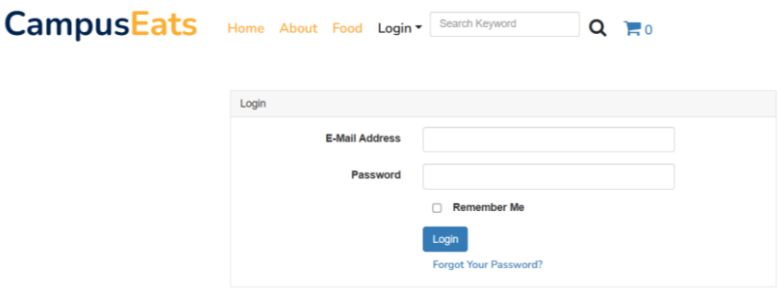

Fig. 13 *Login interface*

```
<div class="card">
    <div class="card-header">{{ __('Login') }}</div>

    <div class="card-body">
        <form method="POST" action="{{ route(name: 'login') }}">
            @csrf

            <div class="form-group row">
                <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>

                <div class="col-md-6">
                    <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ old(key: 'email') }}" required autocomplete="email" autofocus>

                    @error('email')
                        <span class="invalid-feedback" role="alert">
                            <strong>{{ $message }}</strong>
                        </span>
                    @enderror
                </div>
            </div>

            <div class="form-group row">
                <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>

                <div class="col-md-6">
                    <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" required autocomplete="current-password">

                    @error('password')
                        <span class="invalid-feedback" role="alert">
                            <strong>{{ $message }}</strong>
                        </span>
                    @enderror
                </div>
            </div>
```

Fig. 14 *Login interface coding*

Fig 13 is the interface for login. Fig 14 shows the snippet of code for login. Inside the form, the first input field collects the user's email address. It is labelled clearly and styled using Bootstrap's grid system. The old('email') function retains the user's input in case of validation errors. If the input fails validation, Laravel

automatically displays an error message using the @error directive and highlights the field with a red border by adding the is-invalid class.

The second input field is for the user's password. Similar to the email field, it is validated using Laravel's error-handling system. Any validation message related to the password is shown under the input field using the @error directive. The autocomplete="current-password" attribute helps the browser suggest saved passwords to the user.

4.1 Testing Result

The test results will primarily concentrate on functional testing across various modules. This comprehensive evaluation ensures that each module operate as intended, meeting functional requirements seamlessly. The CampusEats undergo inspection to make sure it work as it intended to. Table 6, table 7, and table 8 show the testing of functionality for the modules.

Table 6 *Functionality Testing*

No	Test Case ID	Test Case Description	Expected Result	Actual Result	Actual Output
1	TC-01-001	Verify that the system displays the login page.	User is redirected to main page.	User successfully register and go the main page	Pass
2	TC-01-002	Verify login with incorrect credentials.	Error message shown on the incorrect credentials	Error message shown: "Invalid credentials."	Pass
3	TC-01-003	Verify that the login form validates empty fields.	Validation errors displayed.	Validation errors appear	Pass
4	TC-01-004	Verify successful user registration	Account is created, redirected to login or dashboard.	Registration successful	Pass

Table 7 *Continued Functional Testing*

No	Test Case ID	Test Case Description	Expected Result	Actual Result	Actual Output
5	TC-01-005	Verify registration with already used email/username.	Error shown	Error: "The email has already been taken."	Pass
6	TC-01-006	Verify password reset functionality.	Reset email sent to user's email	Email sent	Pass
7	TC-02-001	Verify that the system allows authorized users to add menu items.	New menu item appears in list	Menu item added and listed correctly	Pass
8	TC-02-002	Verify that the system allows authorized users to update menu items.	Updated item is shown	Item is updated in menu list	Pass
9	TC-02-003	Verify that the system allows authorized users delete menu items.	Item is removed from menu	Menu no longer shows the deleted item	Pass
10	TC-02-004	Verify that the system allows authorized users making order	Item is added to the list and ready to make an order to the selected item list	Menu is added to the list, and ready to make a payment	Pass
11	TC-03-001	Ensure Manager can create and manage promotions.	Promo appears in list	Promo added successfully	Pass
12	TC-03-002	Ensure Manager can create and manage promotions.	Updated promo shown	Changes saved	Pass
13	TC-03-003	Ensure Manager can create and manage promotions.	Promo Removed	Promo no longer visible	Pass
14	TC-04-001	Confirm orders can be placed	Order is submitted and visible in order panel	Order listed under Manage Orders	Pass
15	TC-04-002	Confirm orders can be tracked	Status reflects correctly	Status changed	Pass
16	TC-05-001	Test payment processing and confirmation	Payment confirmed	Payment successful, order status updated	Pass
17	TC-05-002	Test payment processing and confirmation	Payment marked as confirmed	Payment record updated	Pass

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 631-648

Table 8 *Continued Functionality Testing*

No	Test Case ID	Test Case Description	Expected Result	Actual Result	Actual Output
18	TC-06-001	Verify report generation functionality.	Sales report is displayed	Sales report appears	Pass
19	TC-07-001	Test ability of customers to submit feedback.	Feedback submitted	Confirmation shown	Pass
20	TC-07-002	Test ability of Manager to view feedback.	Feedbacks displayed	List of feedbacks appears	Pass
21	TC-08-001	Verify system settings management	Settings saved	Config updated	Pass
22	TC-08-002	Verify system settings management	Changes saved	User access updated	Pass

Table 6, table 7, and table 8 shows the test case ID for each of the modules. This comprehensive evaluation encompasses the functionality of the system. By prioritizing functional testing, CampusEats can continuously improve and establish a strong and trustworthy use, and support the overall quality and reliability of the systems.

4.2 User Acceptance Form

The user acceptance of the system is gathered via an online survey questionnaire created using Google Forms. The questionnaire has been shared to respondents for the purpose of user acceptance testing. The results obtained from the user acceptance form are categorized into two category. The first category is related to the User Interface (UI), while the second category focuses on the User Experience (UX). Fig 15 shows the survey of the overall system interface

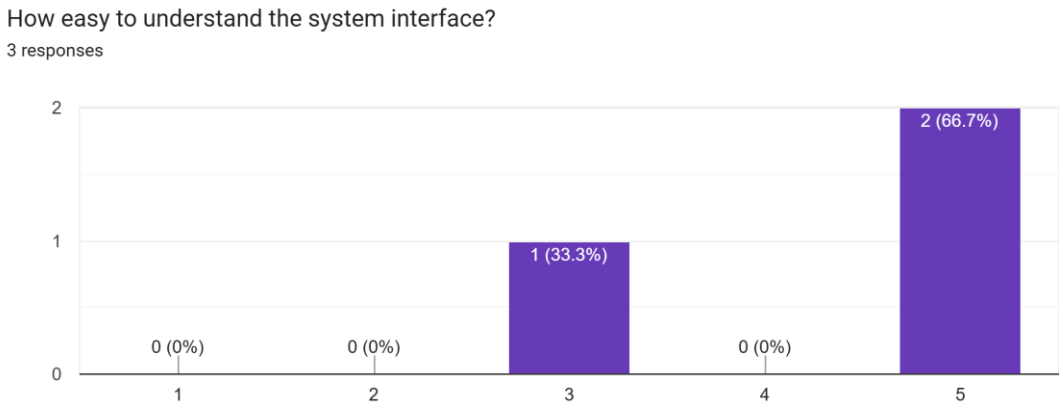

Fig. 15 *Overall system interface responses*

Fig 15 shows the result of feedback gathered from the google form. For this result 2 user review on highly satisfied (66.7%) with the system interface, while 1 student response on the neutral (33.3%). This is because each of the user have different perspectives. CampusEats implement simplify versions of view for the user who are using the system. The goals of CampusEats are to caters every user so the user can use the system without hassle.

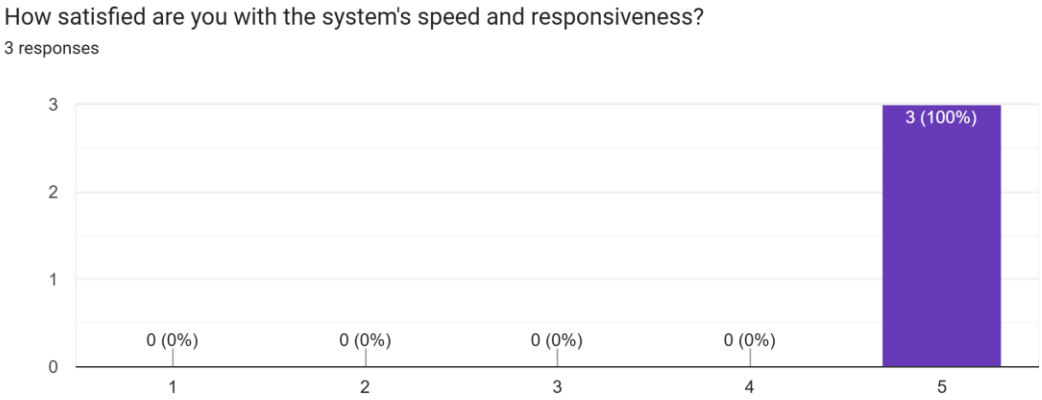

Fig. 16 *System Speed and responsiveness responses*

Fig 16 shows that users are satisfied with the system's speed and responsiveness (100%). This aligns with the system's goal of eliminating the traditional food ordering method by providing CampusEats as a faster and more responsive solution for placing orders.

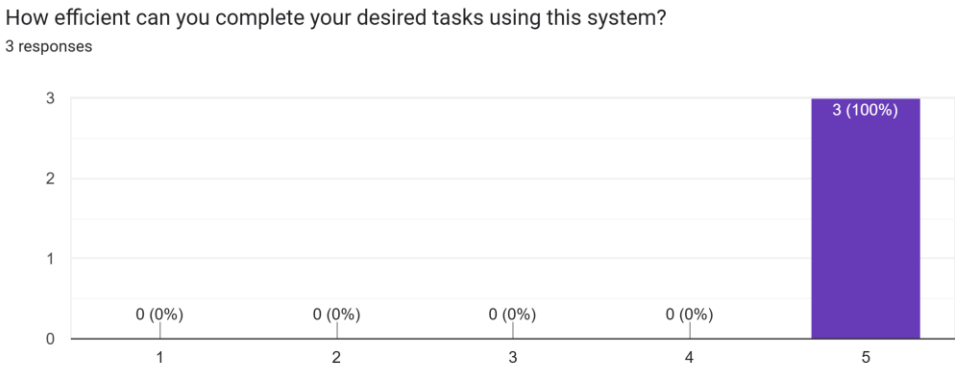

Fig. 17 *Effiency of complete desired task reponses*

As in fig 17 shows the all of the respondents reported being highly satisfied (100%) with how efficiently they could complete their desired tasks using the CampusEats system. This indicates that the system effectively supports users in performing key functions with minimal difficulty or delay.

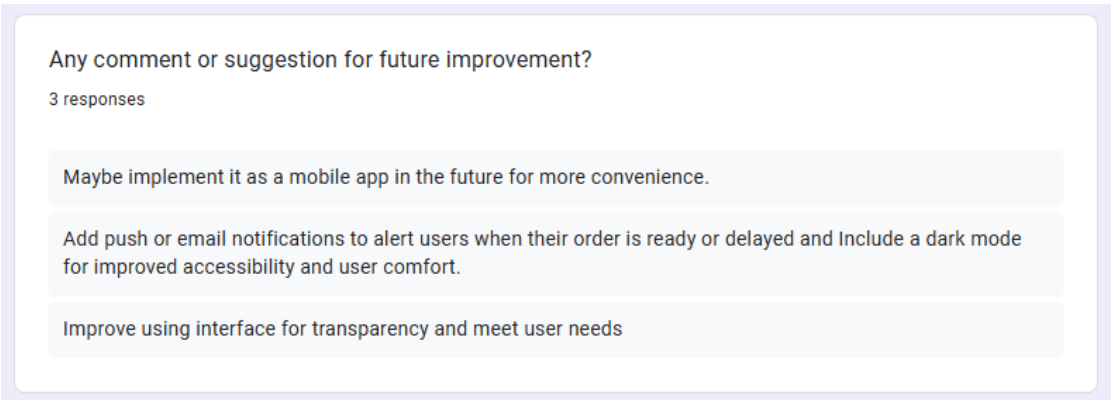

Fig. 18 *Comment or suggestions response*

Based on the feedback from respondents as in fig 18, several suggestions for future improvement were provided. One common recommendation was to implement the system as a mobile application to enhance

convenience and accessibility, especially for users who are frequently on the move. Another suggestion emphasized the addition of push or email notifications to inform users when their orders are ready or delayed, which would improve communication and overall user satisfaction. Additionally, respondents proposed interface enhancements such as the inclusion of a dark mode for better visual comfort and accessibility, as well as general improvements to the user interface to ensure greater transparency and a more user-friendly experience.

5. Conclusion

The CampusEats Web-Based Food Ordering System significantly enhances the meal ordering process for both students and campus food owner. By replacing manual, walk-in ordering with a streamlined digital platform, the system introduces key operational benefits and improves overall user experience. One of the most notable advantages of CampusEats is its ability to handle real time order management. Users can browse food menus, place orders, and receive order status updates without needing to queue physically. This feature saves time for customers and reduces congestion at the food shop, especially during peak hours such as lunch and dinner. Another advantage is the promotion and discount management module, which allows Food owner to create time-limited promotions or offer student specific discounts. This feature boosts customer engagement while providing a flexible pricing strategy for food owner. The user authentication and registration module ensures secure and personalized access. Students can easily sign up, log in, and manage their profile information, while food vendors and administrators have separate access roles to manage their respective data. This ensures the system maintains both privacy and efficiency.

Despite its many strengths, the current version of the CampusEats system presents several limitations that could impact its scalability and broader usability in the long term, including the system currently lacks of comprehensive mobile optimization. While the web interface is responsive on most modern browsers, additional mobile-specific adjustments are needed for seamless performance on smaller screens.

To further enhance the CampusEats platform, adding features such as multi-language support can significantly enhance user experience and inclusivity. Lastly, ongoing user feedback collection and iterative design improvements should remain a priority, as they ensure the system evolves based on real world needs and continues to meet user expectations.

Acknowledgement

The author would like to thank the Faculty of Computer Science and Information Technology, University Tun Hussien Onn Malaysia for its support.

Conflict Of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design:** Muhammad Haziq, Mohd Hamdi; **data collection:** Muhammad Haziq; **analysis and interpretation of results:** Muhammad Haziq, Mohd Hamdi, **draft manuscript preparation**: Mohd Hamdi, All authors reviewed the results and approved the final version of the manuscript.

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

References

[1] Ahmad, D. (2018). The food delivery battle has just begun in Malaysia. Retrieved from https://ecinsider-my.blogspot.com/2018/02/food-delivery companies-malaysia.html

[2] Bagwan, M. K., & Ghule, P. S. (2019). A modern review on Laravel-PHP framework. IreJournals, 2(12), 1-3

[3] Annaraud, K., & Berezina, K. (2020). Predicting satisfaction and intentions to use online food delivery: what really makes a difference?. Journal of Foodservice Business Research, 23(4), 305-323.

[4] Jazayeri, M. (2007). Some trends in web application development. In Future of Software Engineering (FOSE'07) (pp. 199-213). IEEE.

[5] Jindal, T. (2016). Importance of Testing in SDLC. International Journal of Engineering and Applied Computer Science (IJEACS), 1(02), 54-56.

[6] Siau, K., & Lee, L. (2004). Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML. Requirements engineering, 9, 229-237.

[7] Al-Fedaghi, S. (2017). Diagramming the class diagram: toward a unified modelling methodology. arXiv preprint arXiv:1710.00202.

[8] GeeksforGeeks. (2025, March 25). SDLC (Software Development Life Cycle). GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/