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

AITCS

Vol. 5 No. 2 (2024) 1483-1502
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management

Jonathan Joseph[1], Mohd Hamdi Irwan Hamzah[1]*

[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.2024.05.02.080

Article Info

Received: 13 June 2024
Accepted: 30 October 2024
Available online: 15 December 2024

Keywords

QR Food Ordering System, Pablo's, Web-based, Scan&Order, Virtual Waitlist

Abstract

Pablo's restaurant at 49 Bishop Street in George Town, Malaysia, is revolutionising dining with its "Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management." This project introduces a web-based solution to improve operational efficiency and customer satisfaction by eliminating the existing system's long wait times and poor order handling. Its main objectives are to develop, build, and test an enhanced meal ordering and virtual waiting list system. PHP and MySQL were used to build the agile, object-oriented system. Its mobile-responsive digital menu and Order Progression Bar allow customisable ordering and live tracking. The system improves restaurant efficiency by improving order processes and communication. Future upgrades will focus on refining the ordering process to maintain Pablo's restaurant competitive advantage. This complete approach ensures Pablo's leadership in innovation, customer satisfaction, and operational excellence.

1. Introduction

Pablo's restaurant owners Pavi and Edmond's entrepreneurial path changed the culinary world. Starting with online food marketing and delivery, it developed to a busy food truck at Gala House, Penang, and a restaurant in George Town, Malaysia. Manual order processing and food order management during peak hours have caused operational issues as the restaurant has grown. To address these difficulties, a cutting-edge Self-Service Food Ordering System with Virtual Waitlist Management is being developed.

Inefficient manual processes at Pablo's include handwritten orders and paper records for order processing. This method causes delays, inaccuracies, and poor data management. The irregular staff allocation during peak hours complicates waiting management and lowers customer satisfaction. The restaurant also struggles with reporting and data analysis, which hinders decision-making. Pablo's manual order processing and reservation management operations are hampered by delays, inefficiencies, and inaccuracies. Staffing imbalances at peak hours impede service and distort responsibility distribution. The lack of sophisticated reporting tools limits the restaurant's ability to assess vital business data and make informed decisions. This cumulative effect causes longer wait times, order errors, and worse customer satisfaction. Thus, Pablo's reputation and revenue are at risk in a competitive business.

The Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management project aims to develop and implement a tailored food ordering system using object-oriented and web-based approach. Key users include the management, waitstaff, kitchen personnel, and restaurant clients. The system incorporates essential modules like user login, self-service food ordering, virtual waitlist management, and data analytics to enhance operational efficiency and customer satisfaction. The primary goal is to create an effective system that allows seamless interaction with the menu, enabling order placement, customization of meals, and special requests. Post-

implementation, the system is expected to improve restaurant operations, reduce wait times, and offer real-time data analytics for informed decision-making[1]. Its scalability ensures adaptability in an ever-changing business environment, facilitating future expansion. The significance lies in revolutionizing Pablo's business operations with faster order processing, efficient waitlist management, and real-time data analytics, establishing it as a technologically advanced and customer-centric restaurant for long-term.

To tackle the research issue methodically, the paper has numerous sections. Section 2 includes a complete literature analysis of relevant earlier research and current implementations. In Section 3, the system development technique is detailed, providing useful insights into system analysis and design. Section 4 then analyzes and discusses the system outcomes, providing valuable insights. Section 5 concludes with a summary of the paper's main findings and research proposals. This methodological structure ensures a logical and observant analysis of the topic, including a literature review, practical application, and insightful recommendations for future research.

2. Related Work

This section provides an overview of the self-serve food ordering system, virtual waitlist management system, web-based application system, and conducts a comparative analysis of the proposed system and the existing system.

2.1 Self-Serve Food Ordering System and Virtual Waitlist Management

The integration of self-service food ordering systems, represented by QR code-based solutions, has dramatically revolutionized the food service business in an era distinguished by technology breakthroughs and increasing consumer expectations. Customers can efficiently peruse menus, customize orders, and improve their eating experience using these technologies, which use intuitive interfaces and QR code scanning [2]. The simplified method eliminates the need for traditional waiter services, enhancing efficiency and lowering labor expenses. Simultaneously, the implementation of virtual queue management systems that use algorithms such as First-In-First-Out (FIFO) further revolutionizes restaurant operations [3]. These systems improve customer experience and streamline seating arrangements by allowing clients to add themselves to waiting lists using internet-connected devices such as smartphones. The FIFO method ensures waitlist fairness by gathering real-time information and decreasing customer uncertainty [4]. Furthermore, virtual queue management systems give significant data insights that feed marketing tactics and facilitate data-driven decision-making. This combined combination of self-service food ordering and virtual waitlist management corresponds to the industry's trend towards increased efficiency, customer happiness, and adaptive management methods.

2.2 Web-based application (WebApp)

Web apps are essential to the digital world in this age of rapid technological advancement. Smartphone adoption and HTML5, CSS3, and JavaScript APIs have made these apps rival native ones in design, functionality, and multimedia integration [5]. Web apps work on PCs, smartphones, and tablets without downloads [6]. In the age of telecommuting, real-time communication, cloud storage, and shared documents have transformed cooperation. Web applications' personalized experiences, secure transactions, and efficient interfaces have optimized tasks and transformed sectors in electronic commerce. Responsive design and third-party integration improve user experiences, but web app security and performance optimization remain issues. Even with these obstacles, web apps shape digital interactions and service delivery in a changing technological context. The proposed system adopts web app technology for its versatility, personalized experiences, and efficiency in electronic commerce despite challenges in security and performance optimization.

2.3 Study of Existing Related Systems

This section will evaluate three existing systems that are similar to the system being created. An examination is performed on the attributes and functions of the existing systems, followed by a juxtaposition with the self-service food ordering system that was created. Three existing systems that have been chosen for comparison are Beep QR by Storehub, Qashier.com, and MYQRMenu Malaysia. Table 1 presents a comparison between the current system and the system that has been designed.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1483-1502

Table 1 *Comparison between Existing Systems and Developed System*

Features	Qashier.com	MYQrMenu	Beep QR Order by StoreHub	Pablo's Food Ordering System
Register and login module	√	√	√	√
Menu browsing and ordering	√	√	√	√
Virtual waitlist management	X	X	√	√
User-friendly interface	√	√	√	√
Ordering processing workflow	√	√	√	√
Admin and Configuration module	√	√	√	√
Sales Monitoring Module	√	√	√	√
Data Analytics and reporting	√	X	√	√
Web-app compatibility	√	√	√	√
Mobile app based	X	X	√	X
System adaptability	√	√	√	√
Internet connection reliability	√	√	√	√
Order Progression Bar	X	X	X	√
Programming Language	Phyton, JS	PHP, Java	JavaScript, PHP	PHP, JavaScript
Database	MySQL	MySQL	MySQL	MySQL

The original Pablo's restaurant system had long lines that made it hard for customers, staff, and the cook to communicate, resulting in blunders and wasted time. However, Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management offers several benefits. Mobile meal ordering reduces wait times. Real-time WhatsApp updates from the imaginary backlog make customers happier. Orders are immediately known to the kitchen and personnel, improving interaction and service. Having many login choices helps managing orders, payments, and customer interactions easier and more efficient for administrators and staff.

3. Methodology

The development process is improved by using a methodology to design, build, test, and maintain software or systems. Proposed system uses agile technique. Agile emphasises iterative development, collaboration, and customer feedback throughout the project life cycle. It allows adaptability and flexibility to changing needs [7]. Sprints, short, incremental cycles of agile development, involve users. Planning, requirement analysis, design, implementation, testing, and deployment are Agile's key phases. This iterative process allows regular functional increment releases, allowing rapid changes in response to user feedback and ensuring the system meets user needs. **Figure 1** displays the Agile paradigm and **Table 2** lists software development tasks. **Appendix A** shows the project's Gantt Chart, which shows tasks and deadlines for each phase. This project runs from 8/10/2023 until 27/6/2024.

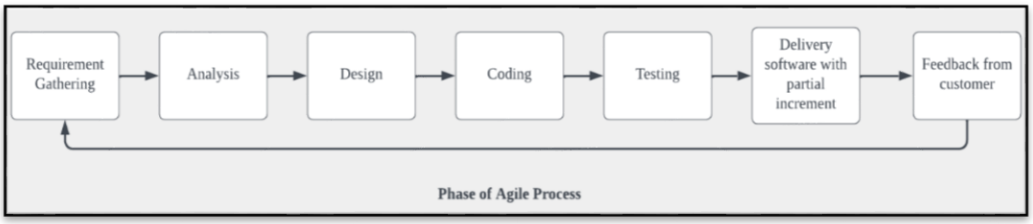


Fig.1 ***Agile Model***

Table 2 *System development activities and tasks*

Phase	Task	Output
Requirement Phase	• Analyze user requirements. • Collect information through interviews • Develop Gantt Chart for project scheduling	• Detailed user requirements • Project proposal • Gantt chart • Literature review
Analysis Phase	• Interview with the user- owner of the restaurant • Analysis of hardware and software requirements	• User requirements • Hardware and software requirements • Sequence diagram, activity diagram, case diagram
Design Phase	• Design wireframes of the system • Design database • Design a user-friendly UI/UX	• Object-oriented approach selected • Wireframe of the system's flow • User interface for the systems
Testing Phase	• Test the system for defects and requirements compliance • Perform interface, functional, and security testing • Resolve defects	• System testing results and defect reports • Test outcomes and security assessment • Defect resolution and quality assurance
Delivery Phase	• Deploy the system to customer environments • Conduct user training and system orientation	• Installed system for customer use • Users trained to operate the system

3.1 System Requirements

The process of requirements analysis is the study, definition, and recording of user expectations and requirements for a software system to address a problem. Functional, non-functional, user, software, and hardware requirements make up system requirements. List of necessary functions of the system is provided in the functional requirements. The system under development, the intended audience for the programme, and the requirements writing approach of the organisation all influence these specifications [8]. Not operations, but non-functional requirements define system behaviour. Usability, security, performance, and other quality-related non-functional needs are among them [9]. The functional requirements of the suggested system are listed in **Table 3**, and its non-functional requirements in **Table 4**.

Table 3 *Functional requirements*

No	Module	Description
1	Order Processing Module	• Streamline the order processing workflow from placement of delivery • Ensuring accuracy in handling customer orders • Facilitate efficient communication and coordination among Pablo's staff involved in the order fulfilment process.
2	Admin and Configuration	• Allow administrators to manage tables, menus, staff, and sales. • Provide a centralized platform for overseeing and configuring key business elements • Enable easy and intuitive navigation for administrators

Table 3*(cont)*

No	Module	Description
3	Sales Monitoring and Data Analytics Module	• Administrators can access sales reports • Monthly sales records can be sorted for analysis • Generates analytical insights into customer preferences and operational performance
4	User Login	• Implements a secure login system with role-based access • Authenticates restaurant staff based on their roles • Ensures a seamless and secure login experience for Pablo's Restaurant Admin and Staff
5	Payment Module	• Develops an intuitive interface for customers to proceed with payments independently • Enables customers to complete transactions seamlessly • Enhances the overall user experience for customers during the payment process
6	Virtual Waitlist Management	• Efficiently manages waiting lists during peak hours • Optimizes restaurant seating capacity by handling customer waitlists effectively • Provides tools and features for Pablo's Restaurant Manager and Staff to manage virtual waitlists

Table 4 *Non-functional requirements*

No	Requirements	Description
1	Performance	• The system should be responsive and provide quick order processing to ensure a seamless customer experience • The system should be able to handle peak load of orders during busy hours without significant degradation in performance
2	Operational	• The system should be available 24/7, allowing customers to place orders at anu time during operational hours • The system should provide an intuitive and user-friendly interface for both restaurant staff and customers to minimize training time and errors
3	Security	• The system should be securely storing user login credentials, especially those of administrators using encryption techniques • The system should implement secure data transmission protocols to protect customer and business data.
4	Cultural and political	• The system should with local regulations and international regulations regarding data privacy
5	Compatibility	• The system should be compatible with a wide range of modern web browsers, including but not limited to Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari • The system should be consistent across different browsers to accommodate the diverse preferences of customers.

The user requirements are the demand and need of the user for the system's functionality. **Table 5** illustrates the system's user requirements.

Table 5 *User Requirements*

No	User Requirements
1	Both administrators and waitstaff must have a registered account with a valid ID and password to access the system.
2	Customers should have the ability to browse the menu and place orders through a self-service interface, which includes the option to scan a QR code for website access.

Table 5 *(cont)*

No	User Requirements
3	The staff should efficiently manage waiting lists during peak hours using the virtual waitlist management system.
4	Administrators should be able to manage menus, staff, and sales of the system
5	Administrators should be able to generate reports, including those on sales and customer preferences.
6	Customers should be able to make payments for their orders directly through the system.
7	Kitchen staff should be notified of new orders and be able to confirm receipt of the orders.
8	Kitchen staff should have the capability to generate receipts for completed orders.
9	Customers should be notified when a table becomes available.
10	The system should incorporate security measures, such as encrypted storage and secure data transmission.
11	Administrators should have the capability to create, read, update, and delete items from the food menu.
12	Administrators and staff members should have the ability to log out from the system.

3.2 System Analysis

System analysis includes system development and flow to determine functionality. This section discusses the proposed system's design and analysis. An object-oriented method is used to analyse and design the proposed system and create Use Case Diagrams, Sequence Diagrams, Class Diagrams, and Activity Diagrams. Pablo's restaurant's self-service meal ordering system with virtual waitlist is diagrammed. It highlights customers, kitchen staff, and admin. This method permits immediate ordering and payment. Kitchen personnel use the virtual waitlist and cook. Administration manages staff, generates sales data, and contacts the virtual queue. **Figure 2** depicts Pablo's system's Use Case Diagram.

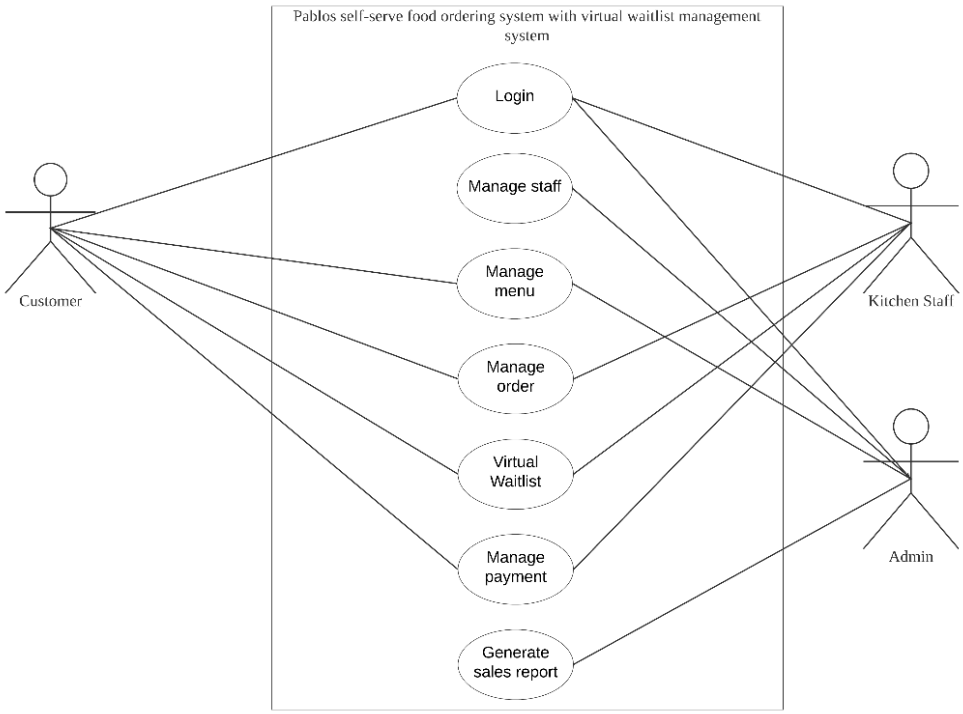


Fig.2 *Use Case Diagram*

3.2.1 Class Diagram

Development of system code is facilitated by the class diagram. Class name, attributes, and relationship are displayed in the class diagram. Use of object-oriented methods is seen in the class diagram in **Figure 3**. Class diagrams, thus, are essential to system development. The class diagram will outline the relationships, function list, attribute list, and class name used in creating Pablo's Virtual Waitlist Management Self-Serve Food Ordering System.

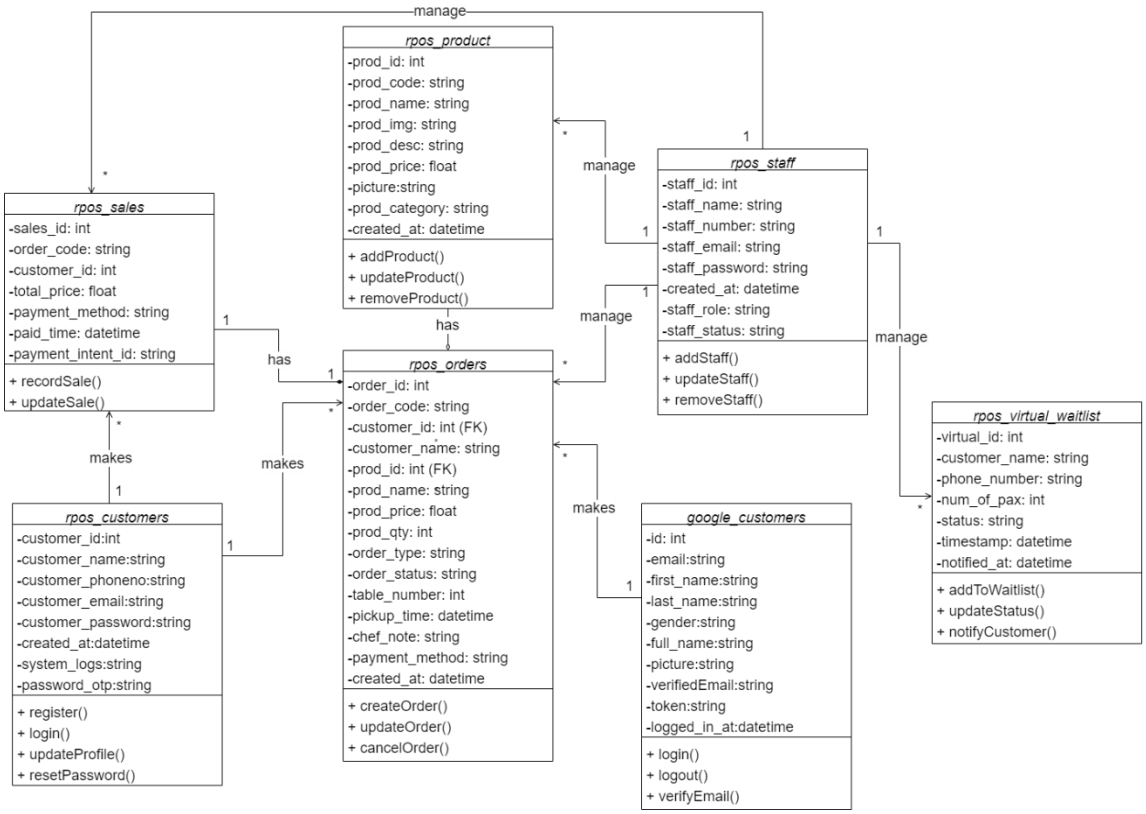


Fig.3 *Class Diagram*

Appendix B, **Appendix C** and **Appendix D** show the flowcharts of customer, admin, and culinary staff for the Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management. The system's flow is illustrated in the flowchart from inception to completion.

3.3 System Design

System architecture, components, interfaces, and data are designed based on the proposed system's demands. This phase creates system architecture and interface wireframes. The system architecture outlines its modules, connections, interactions, and how they operate together to achieve goals. Lucidchart, an online design tool, developed **Figure 4** suggested system architecture.

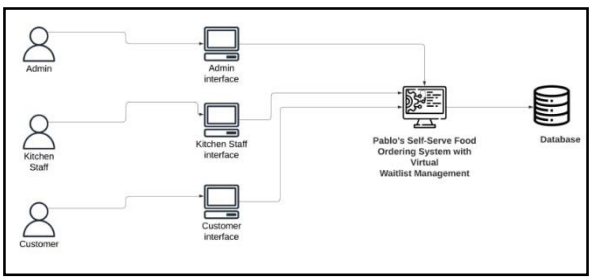


Fig.4 *System Architecture of Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management*

Tables from the database that have been designed and derived from the class diagram are listed below. They were developed with Microsoft SQL Server 2014.

i. `google_customers (id (PK), email, first_name, last_name, gender, full_name, picture, verifiedEmail, token, logged_in_at).`

ii. `rpos_customers (customer_id(PK), customer_name, customer_phoneno, customer_email, customer_password, created_at, system_logs, password_otp).`

iii. `rpos_orders (order_id(PK), order_code(Index), customer_id(Index), customer_name, prod_id(Index), prod_name, prod_price, prod_qty, order_type, order_status, table_number, pickup_time, chef_note, payment_method, created_at).`

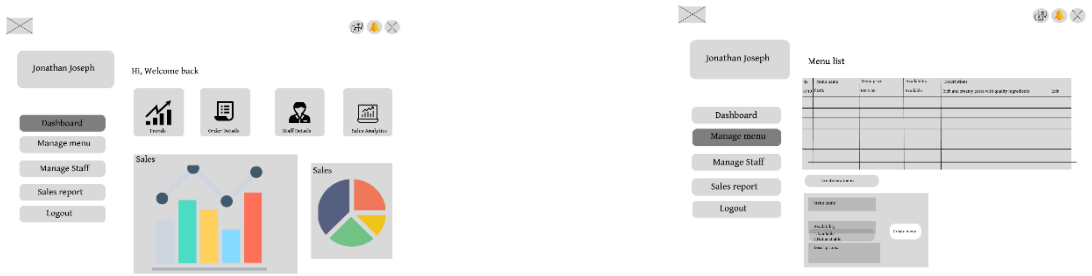
iv. `rpos_products (prod_id(PK), prod_code, prod_name, prod_img, prod_desc, prod_price, prod_category, created_at).`

v. `rpos_sales (sales_id(PK), order_code(Index), customer_id, total_price, payment_method, paid_time, payment_intent_id).`

vi. `rpos_staff (staff_id(PK), staff_name, staff_number, staff_email, staff_password, created_at, staff_role, staff_status).`

vii. `rpos_virtual_waitlist (virtual_id(PK), customer_name, phone_number, num_of_pax, status, timestamp, notified_at).`

Figure 5, interface of admin dashboard 5(a), Figure 5(b), **Figure 6**, interface of customer dashboard 6(a), Figure 6(b), and **Figure 7** show the wireframe of the customer interface and admin dashboard for Pablo's Self-Serve Food Ordering System with Virtual Waitlist Management, which were created using Figma, another online design tool.



(a) **(b)**

Fig.5 *Interface of Admin dashboard (a)Main dashboard;(b) Manage menu interface*

(a) **(b)**

Fig.6 *Interface of Customer dashboard (a) Customer main dashboard;(b) Product description page*

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1483-1502

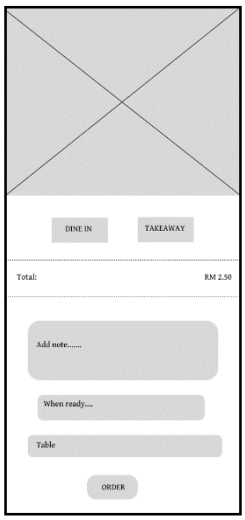


Fig.7 *Order Summary Interface*

4. Results and Discussion

The results of the system's implementation are illustrated in this section. The front end of the system is created using HTML, CSS, and JavaScript, while the back end is created using the PHP language. The integrated development environment was Microsoft Visual Studio Code, and the server and database modules were provided by XAMPP. User Acceptance Testing (UAT) was conducted, and the results are documented in **Appendix E**, **Appendix F**, and **Appendix G**.

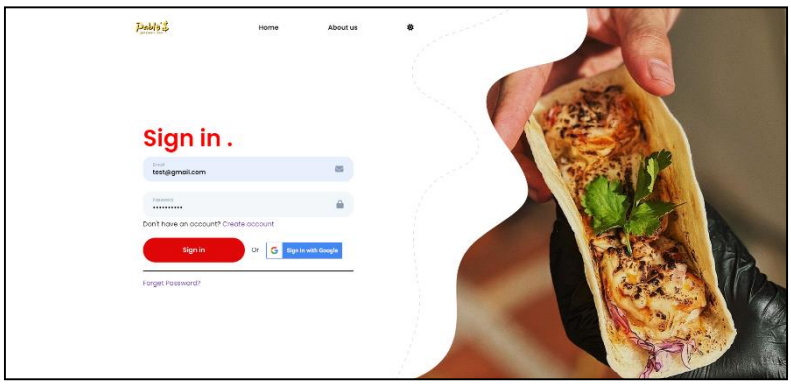


Fig.8 *Interface of Customer login page*

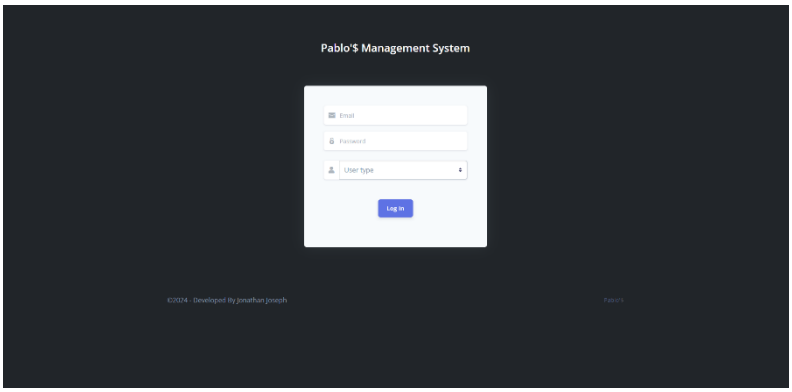


Fig.9 *Interface of Admin login page*

Figure 8 shows Pablo's Self-serve Food Ordering's Customer Login page; **Figure 9** shows the Admin login page. **Table 6** shows the Pablo self-service food ordering system's customer/administrator login module. Customers can log in with Google or email and password on the login page. There are other ways to create an account and recover a forgotten password. The admin login screen requires email and password, then Staff or

Admin from a dropdown menu. System checks user roles and highlights incorrect credentials or role selections. The login module's standardised login experience lets employees quickly access their dashboards.

Table 6 *Test cases for login page*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M1-1	To check whether customers can log in with email and password	The customer should be able to log in with email and password	The customer has successfully logged in with email and password	Pass
M1-2	To check whether customers can create and register an account	The customer should be able to create and register a new account	The customer has successfully created and registered a new account	Pass
M1-3	To check whether customers can log in using Google Sign-In	The customer should be able to log in using Google Sign-In	The customer has successfully logged in using Google Sign-In	Pass
M1-4	To check whether the system sends an OTP for password recovery	The system should send an OTP to the customer's email for password recovery	The system sent an OTP to the customer's email for password recovery	Pass
M1-5	To check whether admin and staff can log in through the same interface	Both admin and staff should be able to log in through the same interface	Admin and staff have successfully logged in through the same interface	Pass
M1-6	To check whether the system authenticates users based on their role	Users should be authenticated based on their role and redirected accordingly	Users were authenticated based on their role and redirected accordingly	Pass
M1-7	To check whether the system shows an error for incorrect username and password	The system should show an error message for incorrect username and password	The system showed an error message for incorrect username and password	Pass
M1-8	To check whether the system shows an error for incorrect user type	The system should show an error message for incorrect user type	The system showed an error message for incorrect user type	Pass
M1-9	To check whether the system shows an error when staff with inactive status login into the system	The system should show an error message when staff with inactive status login into the system	The system showed an error message when staff with inactive status logs in	Pass

Pablo's HRM helps administrators handle employee data. Staff names, emails, responsibilities, and statuses are all editable, delectable, and viewed by administrators. To manage system access, set staff members' active or inactive statuses. Employees can be found by using the filter search feature of the module. Efficiency is guaranteed by the range of functions, which simplify personnel management. The Manage Staff module interface is shown in **Figure 10**; its test cases are listed in **Table 7**.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1483-1502

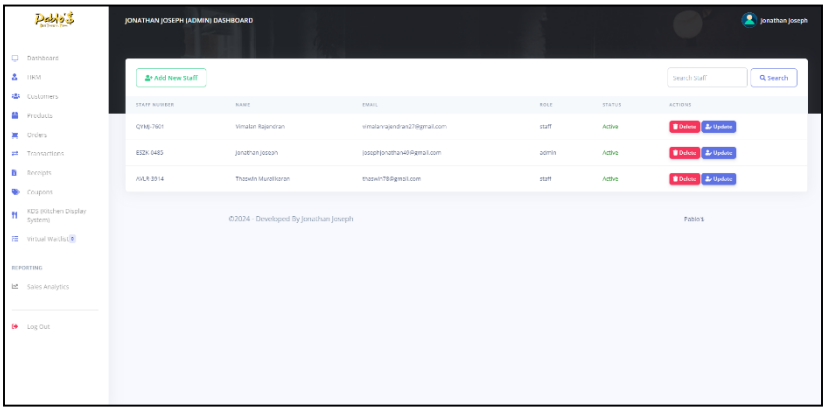


Fig.10 *Interface of Manage Staff Module*

Table 7 *Test cases for Manage staff module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M2-1	To check whether the admin can view the staff list	The admin should be able to view the staff list	The admin successfully viewed the staff list	Pass
M2-2	To check whether the admin can sort the staff table by name	The admin should be able to sort the staff table by name	The admin successfully sorted the staff table by name	Pass
M2-3	To check whether the admin can sort the staff table by staff ID	The admin should be able to sort the staff table by staff ID	The admin successfully sorted the staff table by staff ID	Pass
M2-4	To check whether the admin can create new staff members by inserting staff name, email address, and password	The admin should be able to create new staff members with the specified details	The admin successfully created new staff members with the specified details	Pass
M2-5	To check whether the admin can set roles and statuses for staff as active or inactive	The admin should be able to set roles and statuses for staff	The admin successfully set roles and statuses for staff	Pass
M2-6	To check whether the admin can update staff details	The admin should be able to update staff details	The admin successfully updated staff details	Pass
M2-7	To check whether the admin can delete staff	The admin should be able to delete staff	The admin successfully deleted staff	Pass

Within Pablo's management system, the Manage Menu module enables the administrator to effectively oversee the product inventory. By letting the administrator see, add, change, and remove products, this module ensures that the product options are always updated and accurately represent the menu. **Table 7** lists the test cases for the Manage Menu module; **Figure 11** shows its interface.

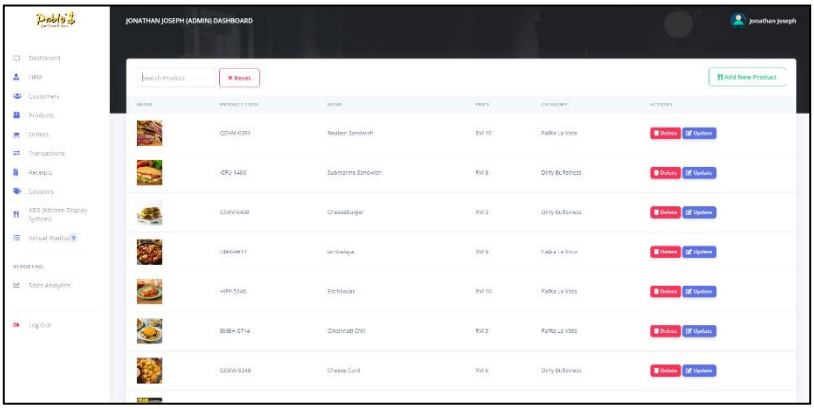


Fig.10 *Interface of Manage Menu Module*

Table 7 *Test cases for Manage staff module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M3-1	To check whether the customer can view the menu with image, name, price, and descriptions	The customer should be able to view the menu with image, name, price, and descriptions	The customer successfully viewed the menu with image, name, price, and descriptions	Pass
M3-2	To check whether the customer can filter the menu by category	The customer should be able to filter the menu by category	The customer successfully filtered the menu by category	Pass
M3-3	To check whether the admin can view the list of products	The admin should be able to view the list of products	The admin successfully viewed the list of products	Pass
M3-4	To check whether the admin can add new products	The admin should be able to add new products	The admin successfully added new products	Pass
M3-5	To check whether the admin can delete products	The admin should be able to delete products	The admin successfully deleted products	Pass
M3-6	To check whether the admin can update existing products	The admin should be able to update existing products	The admin successfully updated existing products	Pass
M3-7	To check whether the admin can filter products by searching for the product name	The admin should be able to filter products by searching for the product name	The admin successfully filtered products by searching for the product name	Pass

The Pablo management system's Manage Order module uses a WebSocket server to provide real-time updates on client orders. Immediately following a customer's purchase, the Manage Order module's user interface (UI) is updated to show the new order. Users can change, cancel, forward the updated order to the kitchen, and after the order is finished, create an invoice for the client. This module guarantees efficient order administration and promotes efficient contact with the kitchen, therefore raising the standard of service provided generally. **Figure 12** depicts the user interface of the Manage Order module, while **Figure 13** showcases the KDS (Kitchen Display System) where chefs acquire the order in real-time. Moreover, **Table 8** shows the test cases connected to the Manage Order module.

Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1483-1502

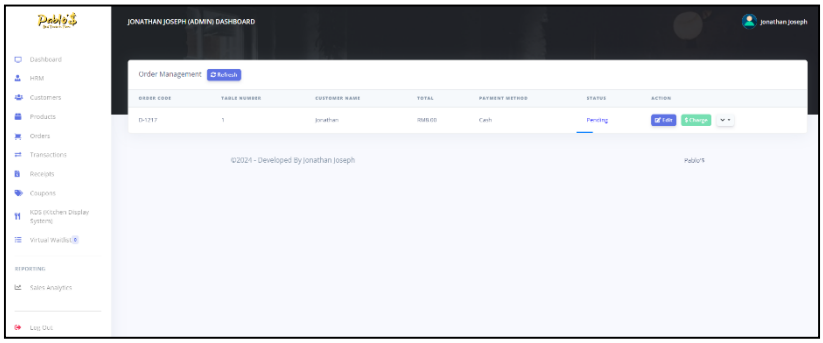


Fig.12 *Interface of Manage Order Module*

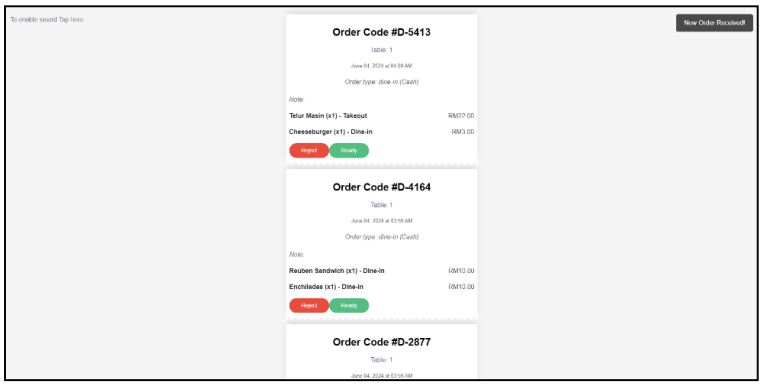


Fig.13 *Interface of Kitchen Display System*

Table 8 *Test cases for Manage Order module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M4-1	To check whether the customer can view added products in the cart	The customer should be able to view added products in the cart	The customer successfully viewed added products in the cart	Pass
M4-2	To check whether the customer can edit order quantity and remove products in the cart	The customer should be able to edit order quantity and remove products in the cart	The customer successfully edited order quantity and removed products in the cart page	Pass
M4-3	To check whether the customer can successfully place an order	The customer should be able to place an order successfully	The customer successfully placed an order	Pass
M4-4	To check whether the customer can see the order progression bar and get real-time order status updates	The customer should be able to see the order progression bar and get real-time order status updates	The customer successfully saw the order progression bar and got real-time order status updates	Pass
M4-5	To check whether the customer can go back to the food menu page	The customer should be able to go back to the food menu page	The customer successfully went back to the food menu page	Pass
M4-6	To check whether the staff receive new orders in real-time	The staff should be able to receive new orders in real-time	The staff successfully received new orders in real-time	Pass

In Pablo's management system, the Virtual Waitlist module offers views from both the admin and customer sides. To add their information and join the virtual queue, customers interact with the user interface shown in **Figure 14**. Easily managing and keeping an eye on the waiting entries, administrators use the interface depicted in **Figure 15**. Moreover, **Table 9** lists the test cases connected to the Virtual Waitlist module.

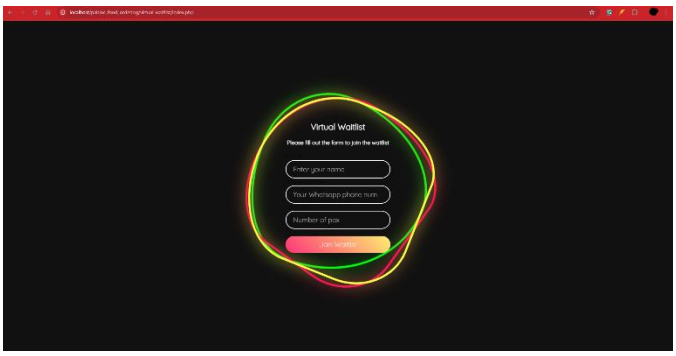


Fig.14 *Interface of Virtual Waitlist module (Customer view)*

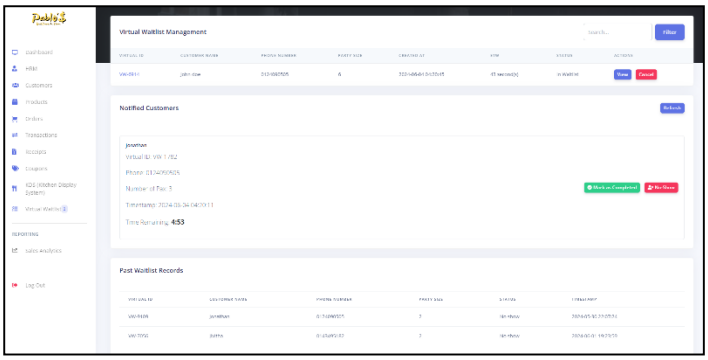


Fig.15 *Interface of Virtual Waitlist module (Admin view)*

Table 9 *Test cases for Virtual Waitlist module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M5-1	To check whether the customer can join the virtual waitlist through the virtual waitlist interface	The customer should be able to join the virtual waitlist through the virtual waitlist interface	The customer successfully joined the virtual waitlist through the virtual waitlist interface	Pass
M5-2	To check whether the customer can get notifications through WhatsApp	The customer should receive notifications through WhatsApp	The customer successfully received notifications through WhatsApp	Pass
M5-3	To check whether the customer is notified when the table is ready through WhatsApp	The customer should be notified when the table is ready through WhatsApp	The customer was successfully notified when the table was ready through WhatsApp	Pass
M5-4	To check whether the customer receives a no-show message if they do not show up upon being notified	The customer should receive a no-show message if they do not show up upon being notified	The customer successfully received a no-show message	Pass
M5-5	To check whether the staff can view waiting customers	The staff should be able to view waiting customers	The staff successfully viewed waiting customers	Pass

Table 9 *(cont)*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M5-6	To check whether the staff can notify customers when the table is ready	The staff should be able to notify customers when the table is ready	The staff successfully notified customers when the table was ready	Pass
M5-7	To check whether the staff can mark a customer's status as completed	The staff should be able to mark a customer's status as completed	The staff successfully marked a customer's status as completed	Pass

Pablo's Manage Payment function in his self-service food ordering system expedites online transactions. Users who choose "Online Payment" at checkout are taken straight to Stripe's safe and easy-to-use transaction page. Customers find convenience in Stripe's FPX and credit card options as shown in **Figure 16**. On a success page, payment confirmation shows. A digital and email receipt made possible by technology gives customers a thorough record of their purchases. Online payments are made safe and quick with this streamlined process, which increases customer satisfaction. After payment, WebSocket pushes real-time order details to the admin page. **Table 10** displays the Manage Payment module test scenarios.

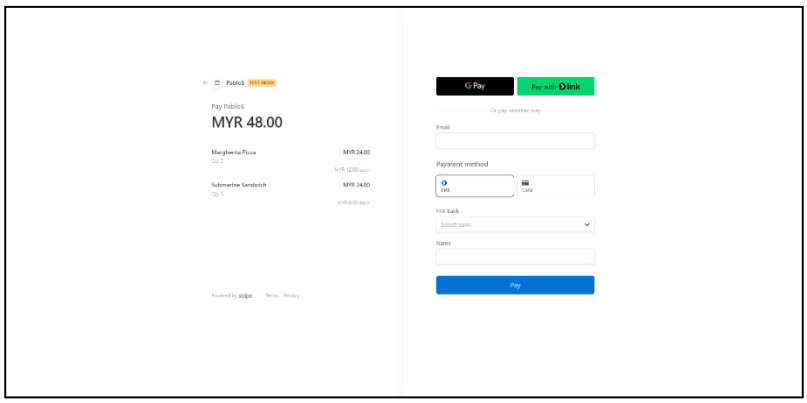


Fig.16 *Interface of Payment module*

Table 10 *Test cases for Virtual Waitlist module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M6-1	To check whether the customer can pay at the counter through the interface	The customer should be able to select and pay at the counter through the interface	The customer successfully selected and paid at the counter through the interface	Pass
M6-2	To check whether the customer receives an email receipt after the payment is completed	The customer should receive an email receipt after the payment is completed	Payment was processed and the customer received an email receipt.	Pass
M6-3	To check whether the staff can charge customers who pay at the counter	The staff should be able to charge customers who pay at the counter	The staff successfully charged customers who paid at the counter	Pass
M6-4	To check whether the staff can process refunds for cash/QR and online payments	The staff should be able to process refunds for cash/QR and online payments	The staff successfully processed refunds for cash/QR and online payments	Pass

The Sales Reporting module of Pablo's management system provides thorough information on the sales performance of the business. Administrators alone can access this module, which offers comprehensive data in chart form including sales, revenue, top-selling and least-selling dishes, and regular customers. Using patterns and trends in the development of a company, administrators can make better selections. With the module's sorting choices by today, week, month, and year, administrators can concentrate on time frames. Administrators can produce sales reports in PDF format as well for sharing or more research. **Table 11** lists the test cases connected to the Sales Reporting module, and **Figure 17** shows its interface.

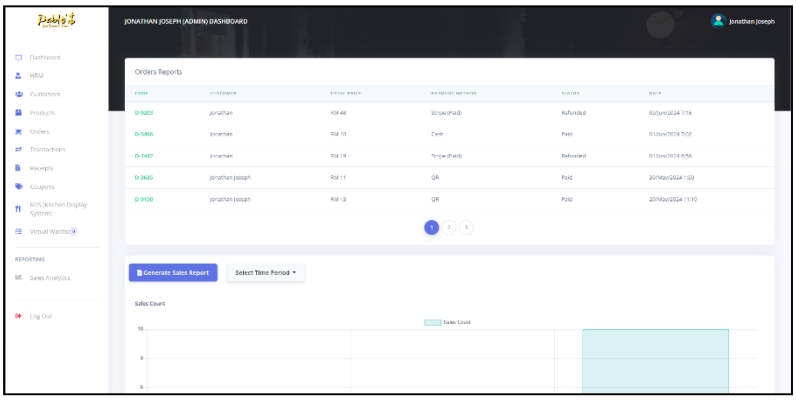


Fig.17 *Interface of Sales reporting module*

Table 11 *Test cases for Sales reporting module*

Test Case ID	Description	Expected Result	Actual	Pass/Fail
M7-1	To check whether the admin can view all sales orders	The admin should be able to view all sales orders	The admin successfully viewed all sales orders	Pass
M7-2	To check whether the admin can view sales trends and patterns through charts	The admin should be able to view sales trends and patterns through charts	The admin successfully viewed sales trends and patterns through charts	Pass
M7-3	To check whether the admin can generate and view sales reports in PDF format	The admin should be able to generate and view sales reports in PDF format	The admin successfully generated and viewed sales reports in PDF format	Pass

Conclusion

Pablo's Self-serve food ordering system with Virtual waitlist management has designed, built, tested, and is now operational his Self-Serve Food Ordering System with Virtual Waitlist Management. The system provides quicker wait times, more effective virtual queue with WhatsApp updates, and mobile meal ordering, all of which increase customer satisfaction. Restaurant efficiency is increased by the order, payment, and customer administration that is streamlined by real-time order updates and many admin login choices. Because it depends so much on the internet, the system cannot be used offline and could be challenging to use in places with bad connection. Integration of messaging systems and payment gateways, together with user aversion to ordering QR codes, call for ongoing upkeep and user education. A loyalty point incentive programme to boost income and client retention, better channels for customer support to raise user satisfaction, and sophisticated analytics to impact operational and marketing strategies are among the future upgrades planned. Pablo's will keep its system improvements creative and competitive.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein 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

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

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

References

[1] E. E.Abel and E. Obeten, "Restaurant Customer Self-ordering System: A Solution to Reduce Customer/Guest Waiting Time at the Point of Sale," *Int J Comput Appl*, vol. 111, no. 11, 2015, doi: 10.5120/19583-1332.

[2] C. Batra, G. Pathak, S. Gupta, S. Singh, C. Gupta, and V. Gupta, "Foodie: An Automated Food Ordering System," 2020. doi: 10.1007/978-981-15-3647-2_40.

[3] M. Jaiswal, "Software Architecture and Software Design," *SSRN Electronic Journal*, 2021, doi: 10.2139/ssrn.3772387.

[4] G. L. Intal, J. D. Payas, L. M. Fernandez, and B. M. Domingo, "Restaurant Information System (RIS) with QR Code to Improve Service Operations of Casual Fine Dining Restaurant," in *2020 IEEE 7th International Conference on Industrial Engineering and Applications, ICIEA 2020*, 2020. doi: 10.1109/ICIEA49774.2020.9102036.

[5] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile software development: Methodologies and trends," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, 2020, doi: 10.3991/ijim.v14i11.13269.

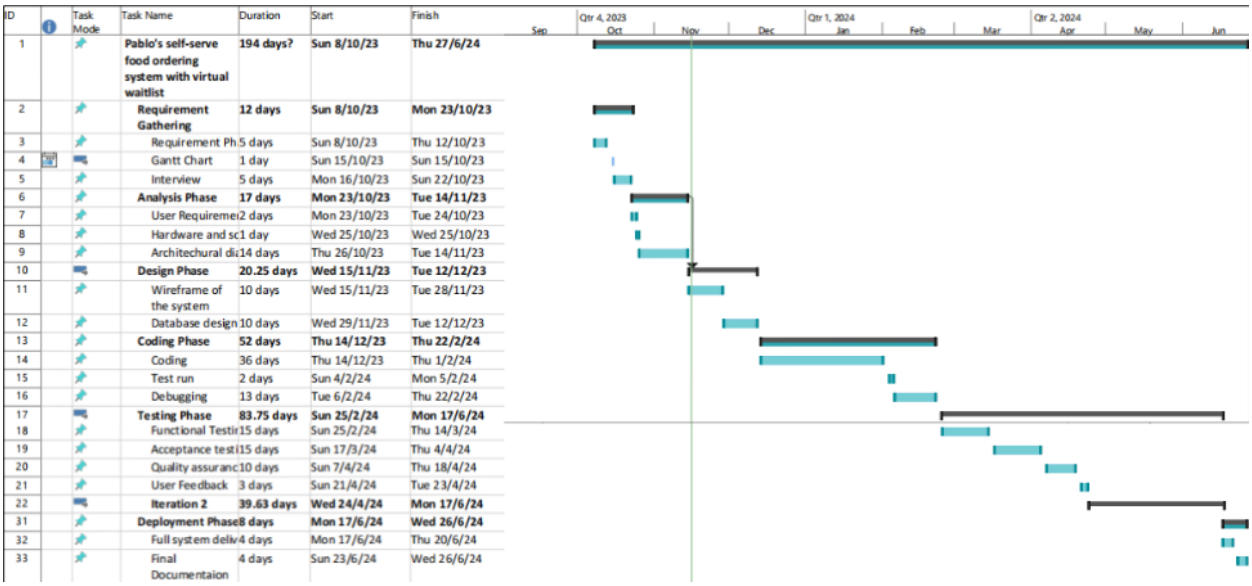
[6] Q. A. Shreda and A. A. Hanani, "Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3052921.

[7] R. Fauzan, I. Krisnahati, B. D. Nurwibowo, and D. A. Wibowo, "A Systematic Literature Review on Progressive Web Application Practice and Challenges," *IPTEK The Journal for Technology and Science*, vol. 33, no. 1, 2022, doi: 10.12962/j20882033.v33i1.13904.

[8] W. Jobe, "Native Apps Vs. Mobile Web Apps," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 7, no. 4, 2013, doi: 10.3991/ijim.v7i4.3226.

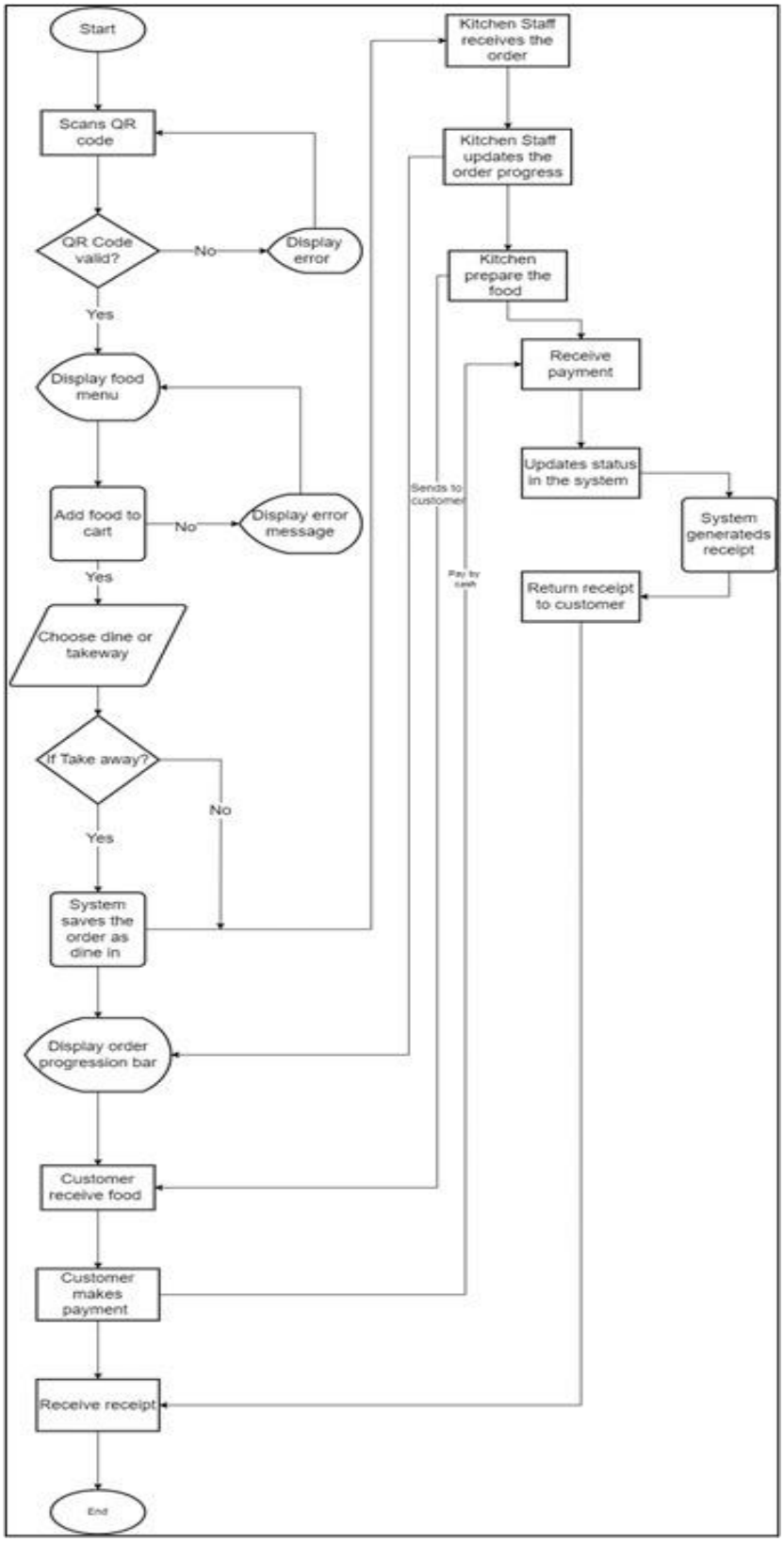
[9] L. A. Austria, "Queue Management Practices of Quick ServiceRestaurants (QSR) in Lipa City, Philippines," *Part II Asia Pacific Journal of Multidisciplinary Research*, vol. 3, no. 5, 2015.

Appendix A: Gantt Chart



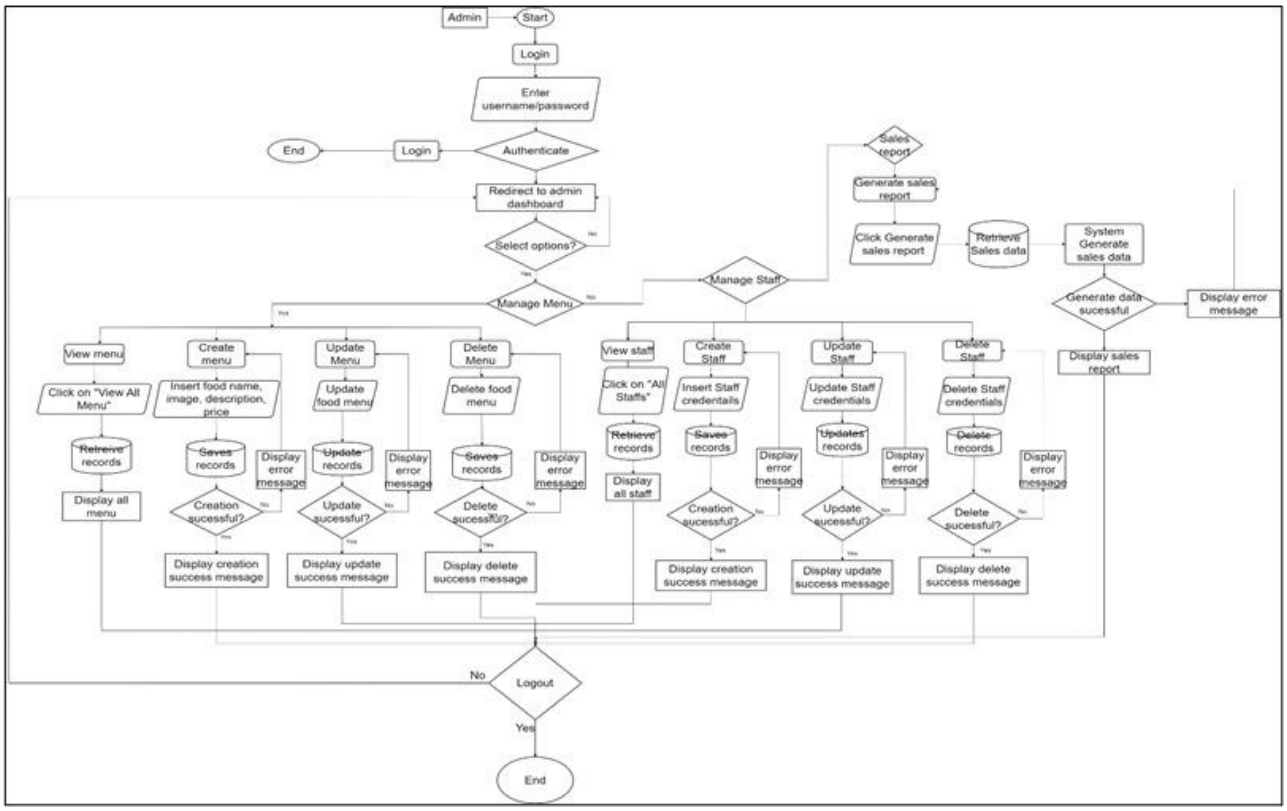
ID	Task Mode	Task Name	Duration	Start	Finish
1		**Pablo's self-serve food ordering system with virtual waitlist**	**194 days?**	**Sun 8/10/23**	**Thu 27/6/24**
2		**Requirement Gathering**	**12 days**	**Sun 8/10/23**	**Mon 23/10/23**
3		Requirement Ph	5 days	Sun 8/10/23	Thu 12/10/23
4		Gantt Chart	1 day	Sun 15/10/23	Sun 15/10/23
5		Interview	5 days	Mon 16/10/23	Sun 22/10/23
6		**Analysis Phase**	**17 days**	**Mon 23/10/23**	**Tue 14/11/23**
7		User Requireme	2 days	Mon 23/10/23	Tue 24/10/23
8		Hardware and sc	1 day	Wed 25/10/23	Wed 25/10/23
9		Architechural di	14 days	Thu 26/10/23	Tue 14/11/23
10		**Design Phase**	**20.25 days**	**Wed 15/11/23**	**Tue 12/12/23**
11		Wireframe of the system	10 days	Wed 15/11/23	Tue 28/11/23
12		Database design	10 days	Wed 29/11/23	Tue 12/12/23
13		**Coding Phase**	**52 days**	**Thu 14/12/23**	**Thu 22/2/24**
14		Coding	36 days	Thu 14/12/23	Thu 1/2/24
15		Test run	2 days	Sun 4/2/24	Mon 5/2/24
16		Debugging	13 days	Tue 6/2/24	Thu 22/2/24
17		**Testing Phase**	**83.75 days**	**Sun 25/2/24**	**Mon 17/6/24**
18		Functional Testi	15 days	Sun 25/2/24	Thu 14/3/24
19		Acceptance testi	15 days	Sun 17/3/24	Thu 4/4/24
20		Quality assuranc	10 days	Sun 7/4/24	Thu 18/4/24
21		User Feedback	3 days	Sun 21/4/24	Tue 23/4/24
22		**Iteration 2**	**39.63 days**	**Wed 24/4/24**	**Mon 17/6/24**
31		**Deployment Phase**	**8 days**	**Mon 17/6/24**	**Wed 26/6/24**
32		Full system deliv	4 days	Mon 17/6/24	Thu 20/6/24
33		Final Documentaion	4 days	Sun 23/6/24	Wed 26/6/24

Appendix B: Flowchart for Customers

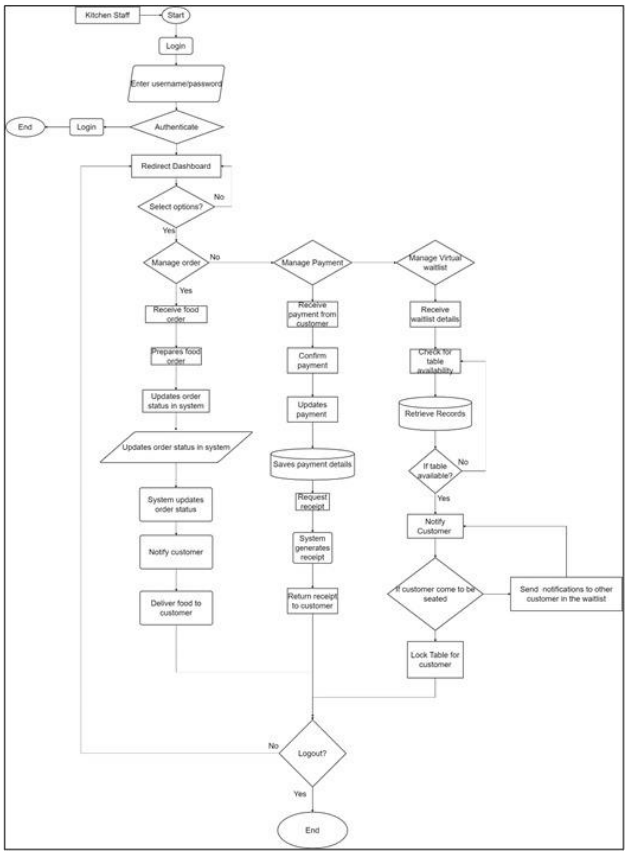


Applied Information Technology and Computer Science Vol. 5 No. 2 (2024) p. 1483-1502

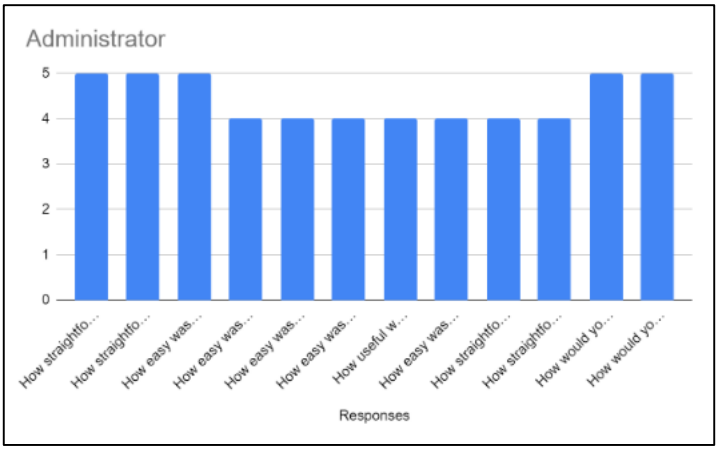
Appendix C: Flowchart for Admin



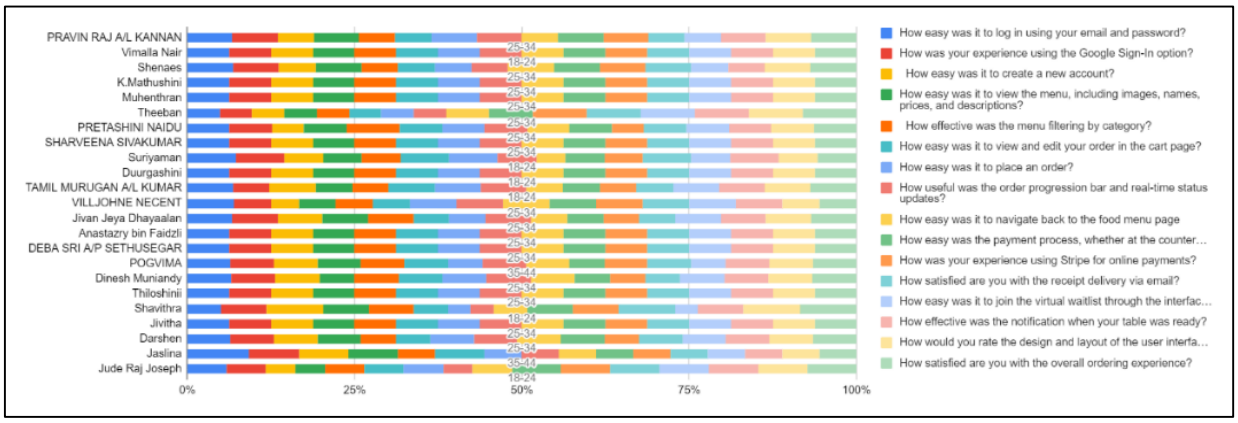
Appendix D: Flowchart for Kitchen Staff



Appendix E: Bar chart of the UAT Result (Admin)



Appendix F: Bar chart of the UAT Results (Customers)



Appendix G: Bar chart of the UAT result (Staff)