HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Selling management system for restaurant

TRAN DUC VIET
viet.td198275@sis.hust.edu.vn

Major: Information Technology

Supervisor:	MsC. Nguyen Duy Hiep	______________
		Signature
Department:	Computer Science	
School:	Information and Communications Technology	

HANOI, 06/2024

ACKNOWLEDGMENTS

With heartfelt gratitude, I extend my deepest thanks to all who have been instrumental in the completion of this thesis. Your collective support and guidance have been the cornerstone of my academic journey and the driving force behind this achievement.

To my beloved family, I owe an immeasurable debt of gratitude. Your unwavering love, support, and belief in my abilities have been my anchor throughout this challenging yet rewarding process. Through late nights of study and moments of doubt, your encouragement has been a constant source of strength, propelling me forward and inspiring me to reach for excellence.

I am profoundly thankful to my esteemed teachers, whose wisdom, dedication, and insightful feedback have been crucial to my academic growth and the realization of this thesis. Your passion for knowledge and commitment to nurturing young minds have not only imparted valuable lessons but also ignited in me a fervent desire to push beyond my perceived limitations.

A special note of thanks goes to MsC. Nguyen Duy Hiep, whose mentorship has been pivotal in the success of this thesis. Your expert guidance, constructive criticism, and unwavering support have been instrumental in elevating the quality of my work and broadening my academic horizons. Your dedication to my growth as a scholar has been truly inspiring.

To my dear friends and colleagues, I extend my warmest appreciation. Your camaraderie, support, and shared experiences have made this journey not just bearable, but genuinely enjoyable and memorable. The stimulating discussions, collaborative study sessions, and moments of levity we shared have enriched this experience in ways I could not have anticipated.

This thesis stands as a testament to the collective effort and support of everyone mentioned above. I am profoundly grateful for your contributions and involvement in making this endeavor a reality. Your impact extends far beyond the pages of this thesis, shaping my personal and professional growth in immeasurable ways. Thank you for being an integral part of this significant chapter in my academic life.

PLEDGE

Student's Full Name: . . . .
Student ID: . . . .
Contact Phone Number: . . . .
Email: . . . .
Class: . . . .
Program: . . . .

I, Tran Duc Viet, commit that the Graduation Thesis (GT) is my own research work under the guidance of MsC. Nguyen Duy Hiep The results presented in the GT are truthful and are my own achievements, not copied from any other works. All references in the GT, including images, tables, data, and quotations, are clearly and fully cited in the bibliography. I take full responsibility for any violations of the school's regulations concerning plagiarism.

Hanoi,

Student

ABSTRACT

In today's rapidly evolving digital landscape, businesses across industries are embracing software solutions to optimize operations and reduce costs. However, despite the abundance of management software available, there remains a significant gap in tailored solutions addressing the specific operational needs of restaurants.

Particularly, the ordering, order placement, and payment processes in many restaurants still rely heavily on manual methods. Upon arrival, patrons are presented with menus, and staff awaits their selections, subsequently recording orders on paper. This antiquated approach often results in misplaced orders or mix-ups between tables, leading to numerous service disruptions and errors. Furthermore, kitchen staff may struggle to prioritize dish preparation, while service personnel may experience confusion when multiple tables order the same dish simultaneously. Such inefficiencies not only prolong service times but also invite customer complaints, tarnishing the restaurant's reputation and service standards.

Presently, only a handful of restaurants have adopted management software solutions for these crucial processes. Typically, it is only big or renowned restaurant chains like Manhwa, Wulao, Haidilao, etc., that possess the financial means to invest in such systems. Consequently, smaller and mid-sized restaurants are compelled to allocate significant resources toward management and operational expenses to enhance efficiency and service quality. While the integration of modern technology into service operations holds promise for efficiency gains, error reduction, and enhanced customer experiences, it presents a considerable hurdle for many smaller-scale dining establishments.

In summary, this thesis aims to develop a comprehensive selling management system for small and medium-sized restaurants. The system will comprise a mobile app for efficient order handling and kitchen preparation sequencing, and a website for menu and user management, dish preparation oversight, and invoice management, including functionality for settling unpaid bills. This solution seeks to bring digital efficiency to smaller restaurant operations, enhancing their competitiveness in the modern dining landscape.

Student

(Signature and full name)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Expression
AOT	Ahead-Of-Time
API	Application Programming Interface
CLI	Command Line Interface
DAO	Data Access Object
DI	Dependency Injection
DTO	Data Transfer Object
F&B	Food and Beverage
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IoC	Inversion of Control
IT	Information Technology
JAR	Java ARchive
JIT	Just-In-Time
JWT	JSON Web Token
RDBMS	Relational Database Management System
SDK	Software Development Kit
SEO	Search Engine Optimization
SQL	Structured Query Language
UI	User Interface
WORA	Write Once, Run Anywhere

CHAPTER 1. INTRODUCTION

1.1 Motivation

Starting from the current situation, despite the rapid advancements in technology, many dining establishments continue to grapple with significant challenges in overseeing the intricate processes of ordering, placing orders, and settling payments. This persistent struggle not only engenders complexities and inaccuracies within service protocols but also exerts a palpable impact on the reputation and service quality of these establishments. Notably, this predicament extends beyond the confines of the restaurant sector, prompting a broader examination into the efficacy of managing and conducting operations within service-oriented enterprises.

Efforts aimed at addressing these challenges promise multifaceted advantages that transcend mere enhancement for restaurant operations, extending to encompass advantages for patrons and the service sector at large. The introduction of a contemporary and streamlined management system tailored for the sales pipeline within dining establishments holds immense potential to bolster workforce productivity, mitigate service-related discrepancies, and reduce operational overheads. As a result, restaurant staff can redirect their focus toward delivering prompt and impeccable service to customers. Moreover, dining establishments stand to achieve enhanced control over the preparation and distribution of culinary offerings, thereby nurturing an enriched customer experience and elevating overall satisfaction levels.

Furthermore, the integration of technological solutions into the restaurant landscape opens pathways for innovation in management and operational paradigms across diverse service sectors. The versatility of a restaurant management system renders it applicable not only within the culinary domain but also within ancillary industries such as hospitality, cafes, and self-service eateries. This underscores the transformative potential of this initiative, not solely within the gastronomic realm but also in catalyzing comprehensive advancements across the broader service industry.

In conclusion, despite the challenges in current operational frameworks, implementing advanced management systems designed for restaurant environments shows potential for improving efficiency, elevating service quality, and fostering innovation across service sectors. This strategic approach not only tackles immediate operational obstacles but also forms the basis for sustained growth and competitiveness within the changing landscape of service-oriented businesses.

1.2 Objectives and scope of the graduation thesis

Nowadays, in Viet Nam, the demand for the solutions of restaurant management is increasing, which reflects the developments and changes in the service industry. So, many restaurants are looking for management tools and software to improve performance, reduce errors, and enhance customer experience.

However, there are some popular restaurant management in Viet Nam such as CukCuk[1], PosApp[2], SapoFnb[3], and iPos[4], which provide many features like table management, ordering, warehouse management, and revenue reporting features with delivery, order management, branch management, etc. With a User-friendly and intuitive interface, they mainly cater to medium and large restaurants. This creates a gap when applied to small and medium-sized restaurants because investment costs in these systems are often high and complex features may not suit their scale and needs. In order of that, Small and medium-sized restaurants need simpler management solutions that save costs but still ensure efficiencies in managing daily operations such as ordering payments, and warehouse management.

From the overall analysis and evaluation of the advantages and limitations of some software as above, this thesis concentrates on creating a selling management system for small and medium restaurants. This system focuses on optimizing the restaurant's ordering and serving processes, while also providing seamless payment functionalities for restaurants. The systems have some features like user management, menu management, ordered dish management, cashier management, and ordering management.

1.3 Tentative solution

Based on clearly defined goals in section 1.2 and research into features of existing restaurant management applications, this thesis aims to develop a comprehensive system comprising a web management platform and a mobile ordering application designed to run on an Android virtual machine and locally.

The system is structured by three main components: back-end, front-end, and mobile app. The back end handles system operations, while the front end and mobile app manage user interaction interfaces. Specifically, the front end utilizes the Angular framework for building the management interface, and the mobile app uses the Flutter framework and Dart language for the ordering interface. The back-end is developed using the Spring Boot framework in Java. The thesis will utilize

[1]`https://www.cukcuk.vn/`
[2]`https://posapp.vn/`
[3]`https://www.sapo.vn/phan-mem-quan-ly-nha-hang.html`
[4]`https://ipos.vn/`

a MySQL database to store system data.

The primary contributions of this thesis include: (i) delivering a system consisting of a website and a mobile application, and (ii) implementing the following features: login, user management, menu management, ordered dish management, cashier management, and ordering management.

1.4 Thesis organization

The structure of this graduation thesis is as follows:

Chapter 2 examines the thesis requirements, divided into four sections: (i) status survey, (ii) functional overview, (iii) functional description, and (iv) nonfunctional requirements.

Chapter 3 covers the technologies employed in this thesis, split into two parts. The first part discusses the background theory and its application in the solution, while the second part provides an overview of each technology used in the thesis.

Chapter 4 describes the system design, including architecture design, database design, and interface design.

Chapter 5 details my primary contributions, the significant challenges encountered during development, and the solutions implemented to overcome them.

Finally, Chapter 6 concludes the thesis, summarizing the thesis results and discussing future work directions.

CHAPTER 2. REQUIREMENT SURVEY AND ANALYSIS

In the opening chapter, the motivations for embarking on this thesis are detailed, the scope and methodology for fulfilling the requirements are outlined, and an overview of the thesis structure is provided. In this chapter, current market applications similar to this system are reviewed, and their strengths and weaknesses are analyzed. Additionally, an overview of the application's features is presented, and key use cases are highlighted.

2.1 Status survey

In Vietnam, the adoption of restaurant management and service software is developing unevenly. Large restaurant chains and high-end establishments have widely implemented these systems, while the majority of small and medium-sized restaurants have yet to deploy them due to high initial investment costs and difficulties in staff training. However, the digitalization trend in the F&B industry is driving demand for software solutions, especially cloud-based systems integrated with artificial intelligence.

The market still has significant growth potential, with expectations of a considerable increase in the adoption of this technology among small and medium-sized restaurants in the near future. Currently, there are several widely used restaurant management software solutions in Vietnam, with SapoFnB and PosApp standing out prominently.

SapoFnB, developed by Sapo, offers modern features such as order management, inventory control, integrated payments, and delivery management. PosApp is known for its user-friendly interface and diverse functionalities, ranging from sales management and table booking to revenue analysis and marketing. Both solutions help restaurants optimize management practices, improve operational efficiency, and enhance customer experiences in Vietnam's vibrant food service industry. Table 2.1 is a comparison table of two restaurant management software PosApp and SapoFnB.

Criteria	**PosApp**	**SapoFnb**
Interface	- Modern and user-friendly. - Multi-platform(web, mobile)	- Intuitive and easy to use. - Supports multiple devices
Features	- Order and delevery management. - Bracnh management. - Integrated online payments.	- Order and deleviry management. - Inventory management and revenue reporting. - Staff and customer management.
Customization	- Flexible customization. - Easy integration with other software.	- Moderate customization. - Integration with Sapo ecosystem.
Cost	- Reasonable cost. - Various service packages suitable for different restaurant sizes.	- Moderate cost. - Service packages suitable types of businesses.
Customer Support	- Professional support. - Support via chat, phone, and email.	- Dedicated and professional support. - Online and on-site support.
Scalability	- Good for restaurants with multiple braches.	- Good scalability within the Sapo ecosystem.
Disadvantages	- Interface may be challenging for non-tech-savvy-users. - High maintenance costs.	- Moderate cost but still high for small restaurants. - Features may not suit small-scale restaurants

Table 2.1: Comparison of two well-know restaurant management system which are PosApp and SapoFnb

2.2 Functional Overview

2.2.1 General use case diagram

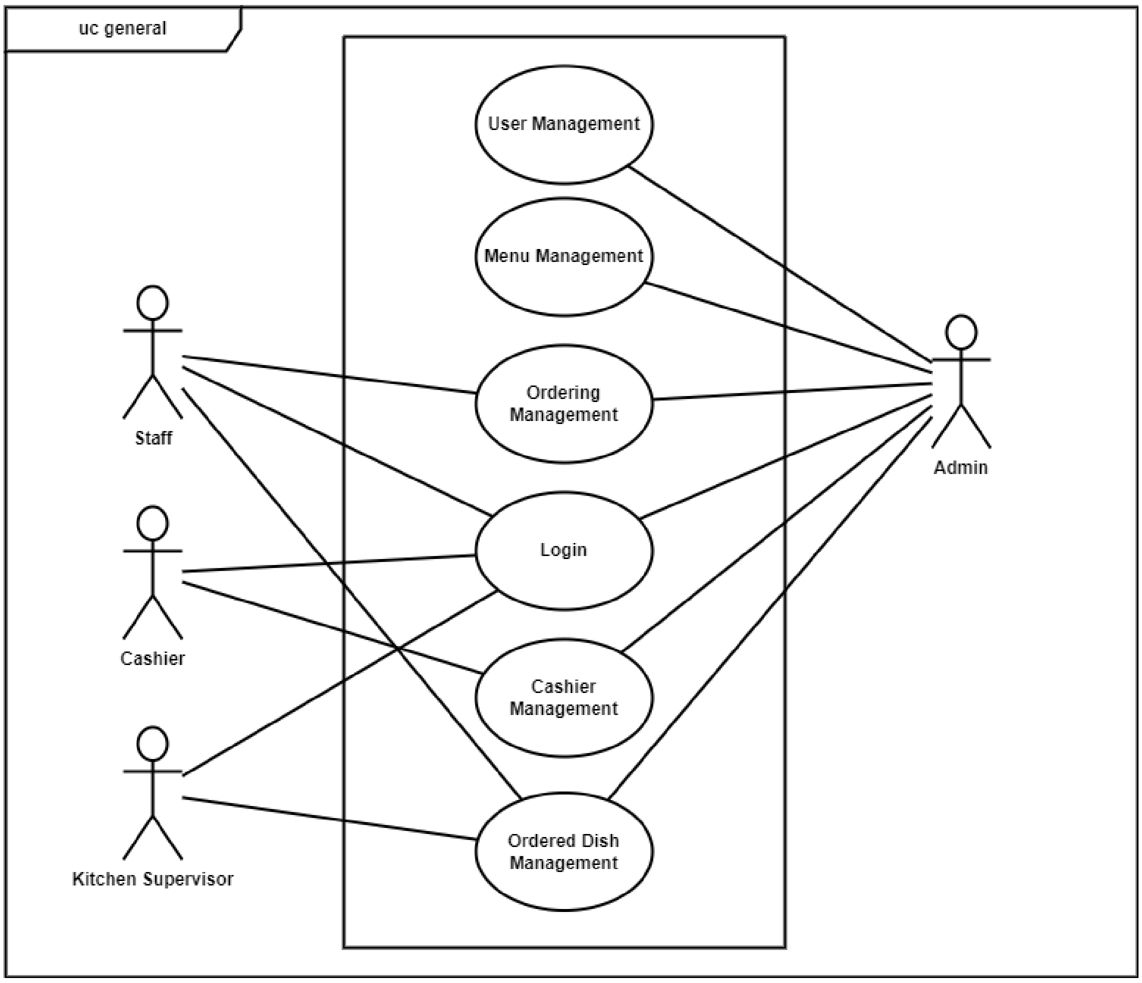


Figure 2.1: General use case diagram

Figure 2.1 describes the general use cases in the system. There are four actors including Staff, Cashier, Kitchen Supervisor, and Admin.

2.2.2 Detailed use case diagram for "Ordering Management"

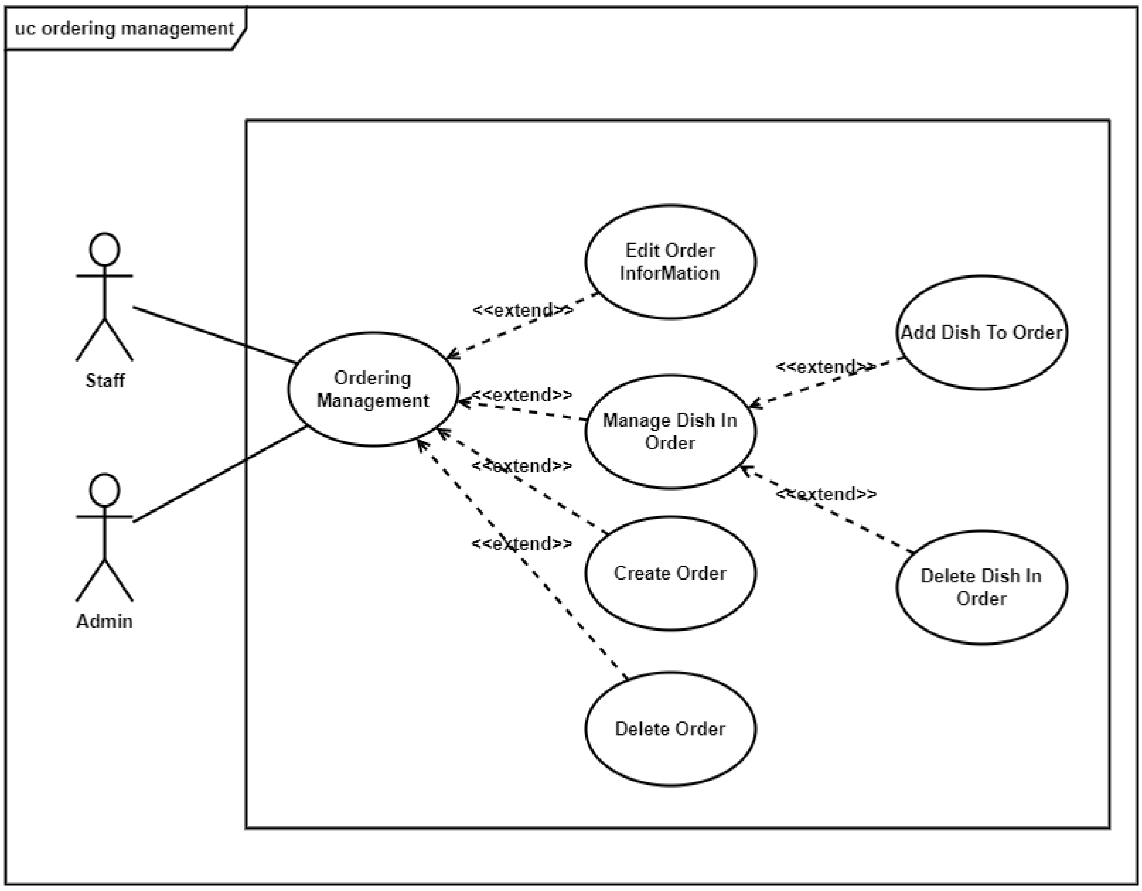


Figure 2.2: Detailed ordering management use case

Figure 2.2 is a use case diagram for ordering management. The staff and the admin can (i) edit order information when needed, (ii) create a new order, (iii) delete an order, (iv) add dishes to order, and (v) delete dishes that are in the order.

2.2.3 Detailed use case diagram for "User Management"

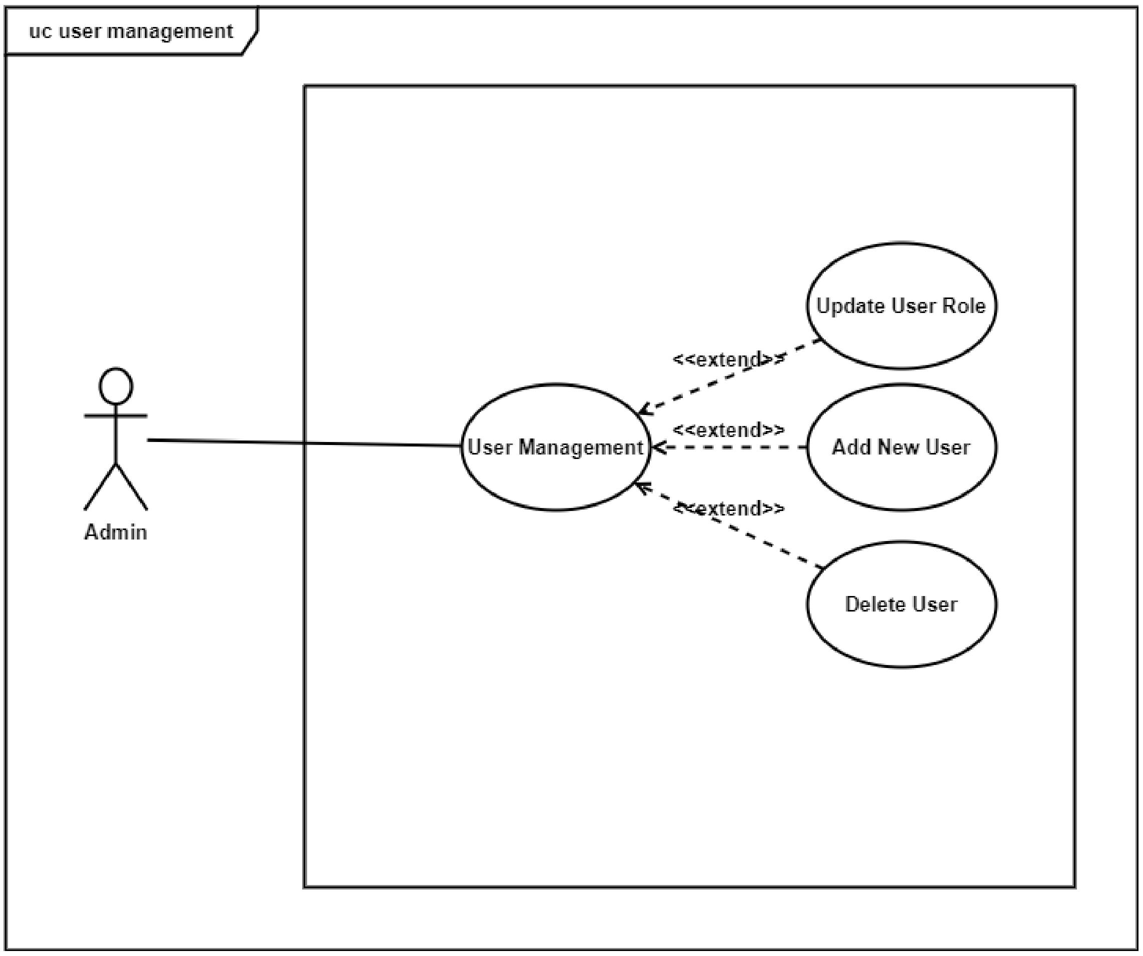


Figure 2.3: Detailed user management use case

Figure 2.3 is a use case diagram for user management. The admin can (i) add a new user, (ii) update the role of the user, and (iii) delete the user.

2.2.4 Detailed use case diagram for "Menu Management"

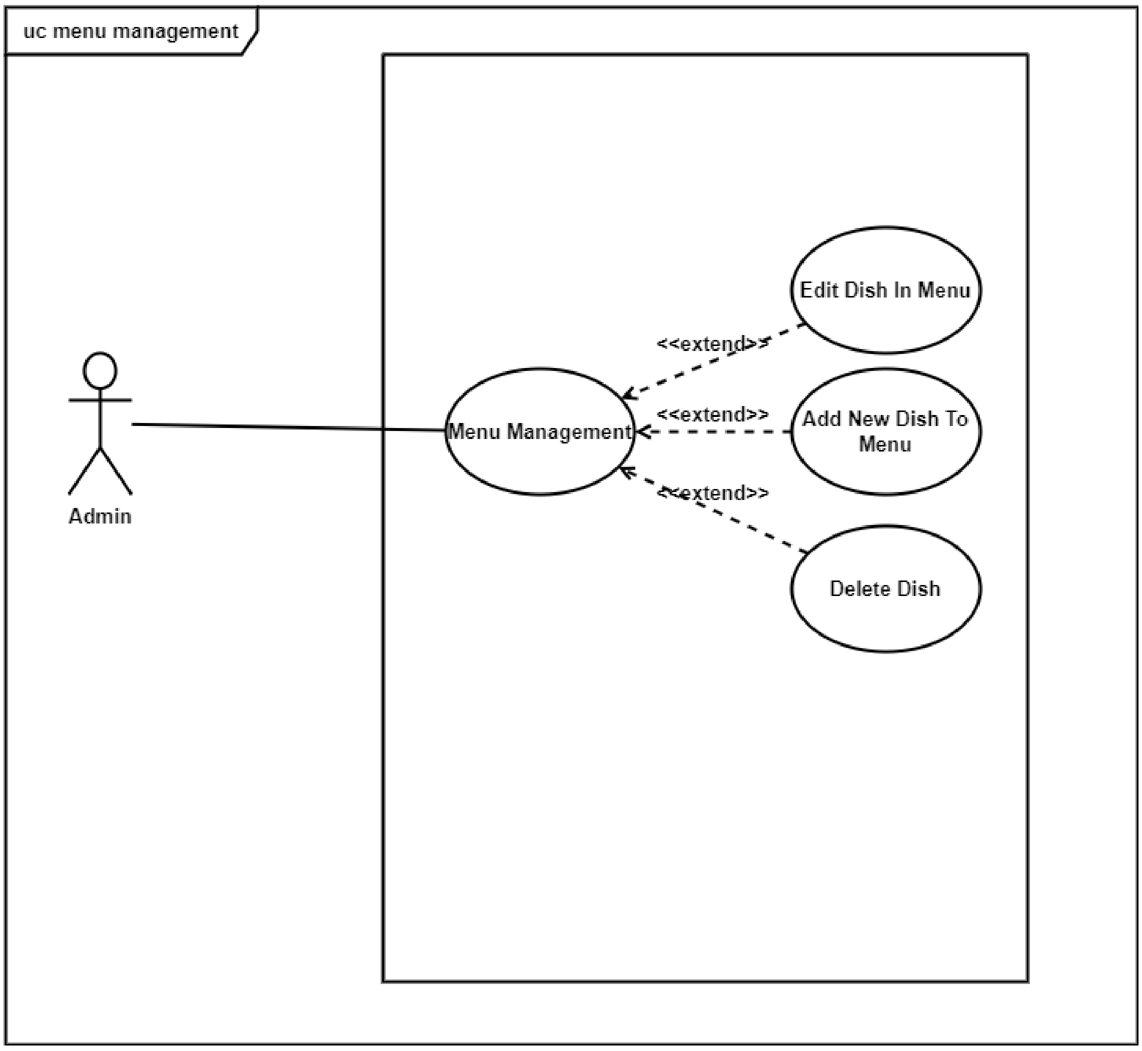


Figure 2.4: Detailed menu management use case

Figure 2.4 is a use case diagram for menu management. The admin can (i) add a new dish to the menu, (ii) update the dish in the menu, and (iii) delete the dish from menu.

2.2.5 Detailed use case diagram for "Cashier Management"

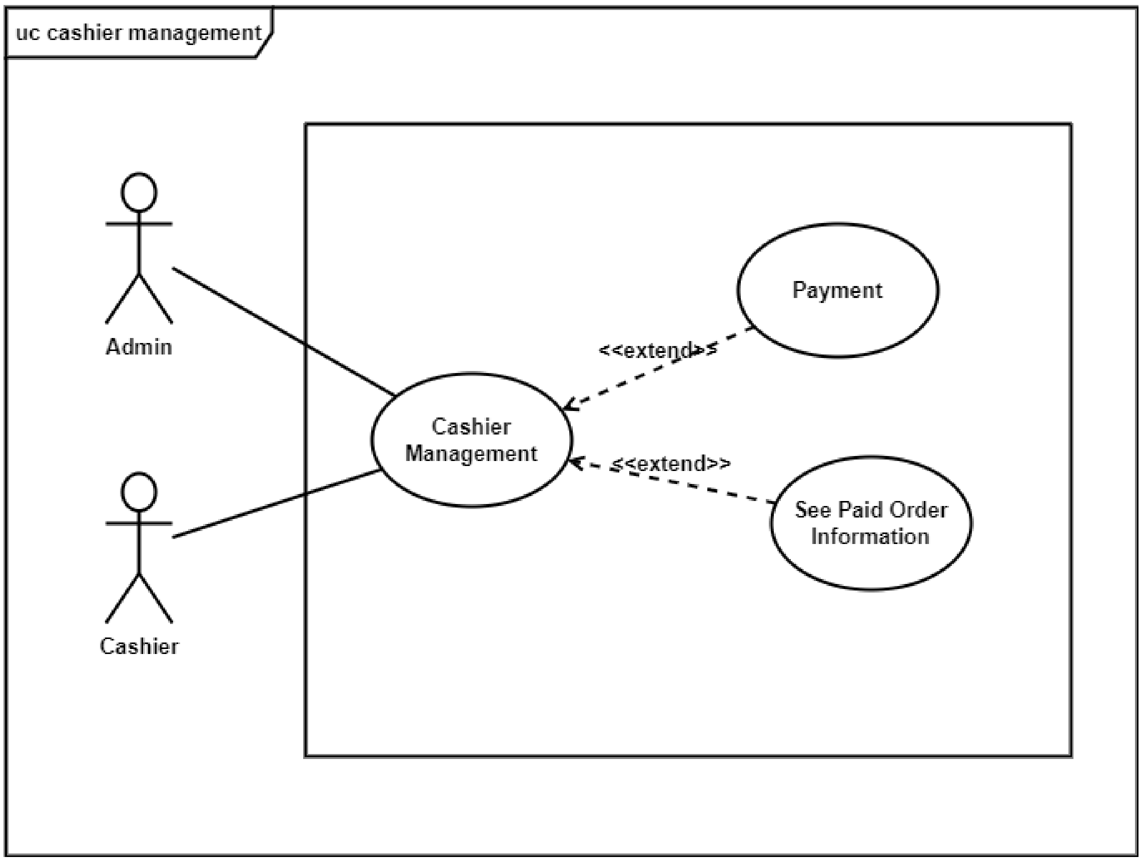


Figure 2.5: Detailed cashier management use case

Figure 2.5 is a use case diagram for cashier management. The cashier and admin can (i) make payment for an order, and (ii) see paid order information.

2.2.6 Detailed use case diagram for "Ordered Dish Management"

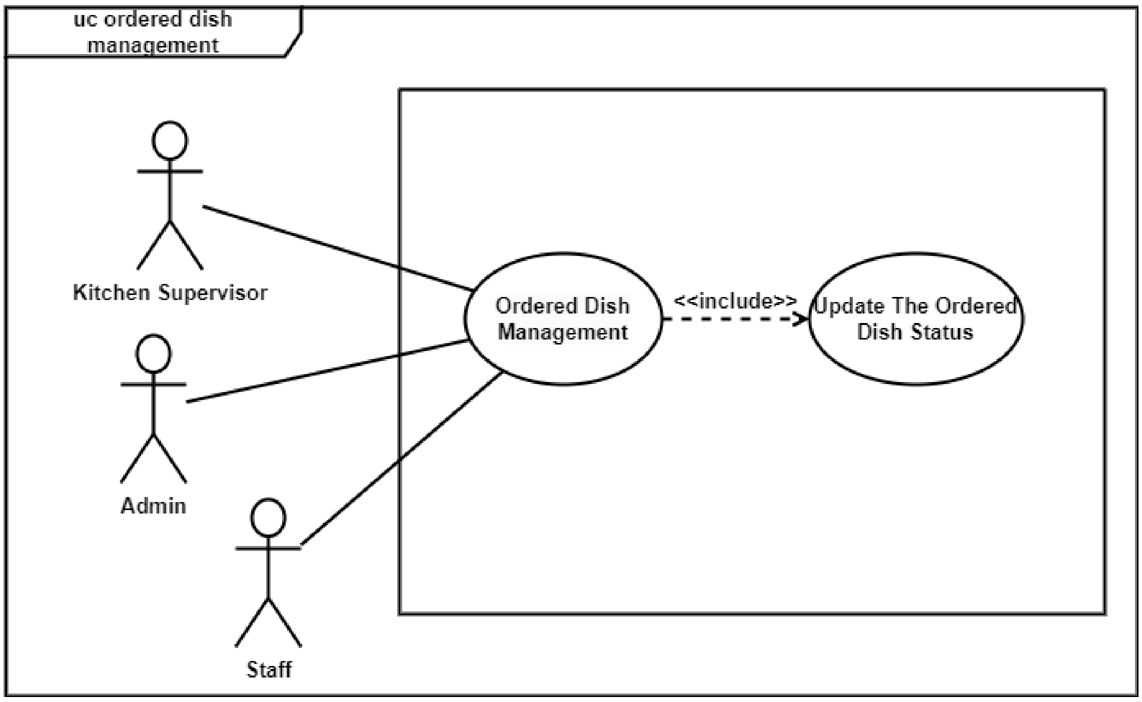


Figure 2.6: Detailed ordered dish management use case

Figure 2.6 is a use case diagram for ordered dish management. The admin, kitchen supervisor, and staff can update the status of the ordered dish,

2.3 Functional description

2.3.1 Description of use case User Management

Table 2.2 describes the use case "User Management".

Code	**UC01**
Use Case	User Management
Actors	Admin
Description	Allows an admin to manage user accounts in the system, including adding new user, update the user role, delete user
Precondition	The admin must be logged in and the device need to be connected to the internet.
Trigger	Admin wants to manage user accounts.
Main Flow	1. Admin select user management in the sidebar. 2. System navigates to the user management page. 3. System displays a list of user accounts in the system 4. Admin selects an action. 5. System confirms the action and updates the list of user accounts. 6. Admin review the updated user list.
Alternative flow	1.1. System detects invalid input data 1.2. Admin corrects input data. 2.1. System encounters an error while doing the action. 2.2. Admin retries the action.
Post-condition	User accounts are successfully managed as per the admin's actions

Table 2.2: Description of use case user management

2.3.2 Description of use case Menu Management

Table 2.3 describes the use case "Menu Management".

Code	**UC02**
Use Case	Menu Management
Actors	Admin
Description	Allows an admin to manage dishes in menu, including adding a new dish, updating the dish, deleting the dish.
Precondition	The admin must be logged in and the device need to be connected to the internet.
Trigger	Admin wants to manage dish in menu.
Main Flow	1. Admin select menu management in the sidebar. 2. System navigates to the menu management page. 3. System displays a list of dishes in the menu. 4. Admin selects an action. 5. System confirms the action and updates the list of dishes. 6. Admin review the updated dishes list.
Alternative flow	1.1. System detects invalid input data 1.2. Admin corrects input data. 2.1. System encounters an error while doing the action. 2.2. Admin retries the action.
Post-condition	Dishes in menu are successfully managed as per the admin's actions

Table 2.3: Description of use case menu management

2.3.3 Description of use case Ordering Management

Table 2.4 describes the use case "Ordering Management".

Code	**UC03**
Use Case	Ordering Management
Actors	Admin, staff
Description	Allows admin and staff to manage an order of a table in the restaurant, including adding a new order, updating a order, deleting a order, and managing dishes of the order
Precondition	The admin and staff must be logged in to the mobile app and the device need to be connected to the internet.
Trigger	Admin, staff wants to manage orders.
Main Flow	1. Admin, staff successfully login to the app. 2. System navigates to the order list screen. 3. System displays a list of ongoing orders in the menu. 4. Admin, staff selects an action. 5. System confirms the action and updates the list of orders and the detailed information in the detailed order screen. 6. Admin, staff review the updated order list or order detail information.
Alternative flow	1.1. System detects invalid input data 1.2. Admin, staff corrects input data. 2.1. System encounters an error while doing the action. 2.2. Admin, staff retry the action.
Post-condition	Orders are successfully managed as per the admin and staff actions

Table 2.4: Description of use case ordering management

2.3.4 Description of use case Cashier Management

Table 2.5 describes the use case "Cashier Management".

Code	**UC04**
Use Case	Cashier Management
Actors	Admin, cashier
Description	Allows admin and cashier to manage payment of an order in the restaurant, including payment of orders, and seeing Paid order information.
Precondition	The admin and cashier must be logged in to web page app and the device need to be connected to the internet.
Trigger	Admin, cashier wants to manage payment or seeing the paid order.
Main Flow	1. Admin, cashier select the actions in the list of functions of cashier management in the sidebar. 2. System navigates to the appropriate page. 3. System displays a list of orders that done ordering. 4. Admin, cashier selects an action. 5. System confirms the action and updates the list of orders and the order. 6. Admin, staff review the updated order list or order.
Alternative flow	1.1. System detects invalid input data 1.2. Admin, cashier corrects input data. 2.1. System encounters an error while doing the action. 2.2. Admin, cashier retry the action.
Post-condition	Orders are successfully updated with the payment information as per the admin and staff actions

Table 2.5: Description of use case cashier management

2.3.5 Description of use case Ordered Dish Management

Table 2.2 describes the use case "Ordered Dish Managment".

Code	**UC05**
Use Case	Ordered Dish Management
Actors	Admin, staff, kitchen supervisor
Description	Allows admin, kitchen supervisor, and staff to manage the status of ordered dishes.
Precondition	The admin, kitchen supervisor, and staff must be logged in to web page or mobile app and the device needs to be connected to the internet.
Trigger	Admin, kitchen supervisor, staff.
Main Flow	1. Admin, kitchen supervisor select the Ordered dish management in the sidebar or admin and staff select the serving dish list in the drawer in app mobile. 2. System navigates to the appropriate page or screen. 3. Admin, staff, and kitchen supervisor select an action. 4. System confirms the action and updates the ordered dish status. 5. Admin, staff, and kitchen supervisor review the updated status of ordered dish.
Alternative flow	1.1. System detects invalid input data 1.2. Admin, cashier corrects input data. 2.1. System encounters an error while doing the action. 2.2. Admin, cashier retry the action.
Post-condition	Orders are successfully updated the ordered dish status as per the admin, kitchen supervisor and staff actions

Table 2.6: Description of use case ordered dish management

2.4 Non-functional requirement

2.4.1 General requirements

The application must ensure that the main functions operate accurately. Additionally, the application needs to provide users with efficient methods for querying and searching data. The data display on the interface must be flexible and capable of generalization. The system should be designed to be maintainable and easily expandable in the future.

2.4.2 User interface requirements

When designing the user interface, attention should be paid to the user experience, ensuring the interface is simple, easy to use, and highly interactive. Additionally, when the screen size changes, data tables should automatically adjust to fit the different screen sizes on various devices.

2.4.3 Security requirements

To ensure security when using the application, the system performs: (i) password encryption user before saving to the database, (ii) do not save important information user as password on the browser's localstorage which only saves the login session's user.

CHAPTER 3. METHODOLOGY

3.1 Backend development technology

3.1.1 Java programming language

Java [1] is one of the popular object-oriented programming languages. It is widely used in software development, web development, game development, and mobile applications. Java was initiated by James Gosling and his colleagues at Sun Microsystems in 1991. Initially, Java was created to write software for household devices and was called Oak. Java was officially released in 1994, and in 2010, Oracle acquired it from Sun Microsystems.

Java was created with the principle of "Write Once, Run Anywhere" (WORA). Software programs written in Java can run on any platform through an execution environment, provided that the appropriate execution environment supports that platform.

The Java programming language has the following notable advantages: (i) Object-oriented - In Java, everything is an object, making it easy to extend and maintain. (ii) Platform-independent - A program written in Java can run well in various environments. (iii) Multi-threaded - Java supports multi-threaded programming to perform concurrent tasks. (iv)Safe - Java has strict requirements for data types, and data must be explicitly declared. (v)Secure - Java has a security manager to define the access rights of classes. With these advantages, Java is chosen to be the programming language for the back-end development of this selling management system for restaurants.

3.1.2 Spring Boot Framework

Java Spring Boot [2] is a powerful and widely adopted framework for Java application development, designed to simplify the creation of Spring-based applications, particularly microservices. Built on top of the Spring Framework, Java Spring Boot inherits its robust features such as Dependency Injection (DI), Inversion of Control (IoC), and seamless integration with various technologies. One of its standout features is automatic configuration, which eliminates the need for extensive setup. Spring Boot configures components based on project dependencies and predefined settings, reducing configuration time and potential errors. Spring Boot also supports powerful tools like Spring Boot Starter, which simplifies adding dependencies by providing default configurations for common features such as web services, data access, security, and more. Additionally, Spring Boot Actuator offers monitoring and management tools to easily monitor and maintain application health. Another key capability is its ability to create standalone applications. Spring Boot applications can be packaged as executable JAR or WAR files and run directly on any Java-supported platform without needing additional middleware like application servers, streamlining deployment, and operations. Furthermore, Spring Boot excels in developing microservices, facilitating the construction of scalable and distributed systems. With Spring Cloud extensions, developers can build complex microservices architectures with features like service discovery, load balancing, and distributed configuration management. Through the above advantages, this system used this framework to build the server for the selling management system for restaurant.

3.2 Front-end development technology

3.2.1 Typescript

TypeScript [3] is a popular superset of JavaScript that brings static typing and modern programming features to the JavaScript language. Developed and maintained by Microsoft, TypeScript was first released in 2012. It is widely used in software development, particularly for building large-scale applications, web development, and server-side development. TypeScript was designed by Anders Hejlsberg, who is also known for creating the C# language.

TypeScript enhances JavaScript with optional static types, allowing developers to catch errors at compile time rather than at runtime. This feature makes TypeScript a powerful tool for developing large and complex applications, where early detection of errors can significantly improve code quality and maintainability.

The TypeScript language offers the following notable advantages: (i) Static Typing - TypeScript provides optional static typing, which helps catch errors early and improves code reliability. (ii) Object-oriented Programming - TypeScript supports classes, interfaces, and other object-oriented principles, making it easier to build and maintain large applications. (iii) Compatibility with JavaScript - TypeScript is a strict syntactical superset of JavaScript, meaning any valid JavaScript code is also valid TypeScript code. (iv) Tooling and Editor Support - TypeScript is supported by a wide range of editors and IDEs, offering features like code completion, navigation, and refactoring. (v) Active Community and Ecosystem - TypeScript has a large and active community, and it integrates seamlessly with popular frameworks and libraries such as React, Angular, and Vue.

Due to these advantages, TypeScript has been chosen as the programming language for the front-end development of this restaurant selling management system, ensuring a robust, scalable, and maintainable codebase.

3.2.2 Angular

Angular [4] is an open-source JavaScript framework developed by Google to support building user interfaces for web applications. Unlike embedding JavaScript into HTML using attributes like AngularJS, Angular is a powerful framework that enables developers to create web applications more easily and with clearer structure.

Angular uses TypeScript for writing code, making application development more flexible and easier to maintain. TypeScript is a superset of JavaScript that extends its features, such as static typing and support for object-oriented programming.

The fundamental building block of Angular is components, similar to ReactJS. Angular components allow embedding HTML and TypeScript together, creating flexible and understandable user interfaces.

The benefits of using Angular include: (i) Ease of writing code with TypeScript and Angular's special syntax; (ii) Ability to divide applications into independent components, facilitating efficient application management and development; (iii) Powerful development and debugging tools supported by a large community and active development from Google; (iv) Strong support for SEO due to server-side data rendering mechanisms that improve the application's searchability.

Based on these advantages, Angular framework has been chosen to build and develop the user web interface of selling management system for restaurant.

3.3 Mobile app development technology

3.3.1 Dart programming language

Dart [5] is a programming language developed by Google, designed to build modern applications across various platforms. Introduced in 2011, Dart has evolved into a powerful tool for developing web, mobile, and desktop applications.

Dart boasts several strengths: (i) Modern and Flexible: Dart supports both object-oriented and functional programming, allowing developers to choose the approach that best suits their projects. (ii) Cross-platform: With Dart, you can write code once and run it on multiple platforms such as web (via Flutter), mobile (iOS and Android), desktop, and even server-side. (iii) High Performance: Dart is optimized to deliver excellent performance, including support for both Just-In-Time (JIT) and Ahead-Of-Time (AOT) compilation. (iv) Community and Support: Dart has a large and active community of users and developers, with extensive learning resources, libraries, and development tools. (v) Flutter framework: Dart is the primary language used for developing applications with Flutter - a popular UI framework from Google that enables building beautiful and interactive user interfaces across platforms.

With these advantages, Dart is chosen to be the programming language for the mobile development of this selling management system for restaurant.

3.3.2 Flutter Framework

Flutter [6] is an open-source UI framework developed by Google for crafting natively compiled applications for mobile, web, and desktop from a single codebase. Launched in 2017, Flutter has gained significant popularity among developers for its fast development cycles, expressive and flexible UI components, and excellent performance.

One of Flutter's standout features is its use of Dart as its programming language. Dart's productivity and capabilities, such as Just-In-Time (JIT) compilation during development and Ahead-Of-Time (AOT) compilation for production-ready apps, contribute to Flutter's efficiency. This approach allows developers to create visually rich and responsive interfaces that run smoothly on various platforms.

Flutter's widget-based architecture makes it easy to build custom UI components and seamlessly integrate them into applications. Widgets in Flutter are not only highly customizable but also enable hot reload, a feature that significantly speeds up the development process by instantly reflecting code changes in the running app.

Moreover, Flutter provides extensive support for Material Design and Cupertino widgets, ensuring consistent and native-like experiences across Android and iOS platforms. Its growing ecosystem of packages and plugins further enhances its capabilities, making it a preferred choice for building modern, cross-platform applications with high-performance user interfaces.

Based on these advantages, Flutter framework has been chosen to build and develop mobile app of selling management system for restaurant.

3.4 MySQL

MySQL [7] is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL), developed, distributed, and supported by Oracle Corporation. MySQL runs on various platforms including Linux, UNIX, and Windows, and is commonly used in conjunction with web applications.

RDBMS stores data in different tables and establishes relationships using Primary Keys and Foreign Keys. MySQL's notable features include: (i) open-source nature, making it free for developers to use; (ii) compatibility with multiple operating systems and languages like Java, PHP, and C++; (iii) support for large databases, handling up to 50 million rows or more per table. The default table file size limit is 4 GB, extendable to a theoretical limit of 8 TB if supported by the operating system; (iv) customizable under the open-source GPL license, allowing developers to modify MySQL to fit specific environments.

Due to these advantages, MySQL is highly suitable as the primary database for this selling management system for restaurant.

CHAPTER 4. DESIGN, IMPLEMENTATION, AND EVALUATION

4.1 Architecture design

4.1.1 Software architecture selection

The system is structured using a client-server architecture, where the client is responsible for providing the user interface, capturing user inputs, and sending processing requests to the server. The server's primary role is to handle the necessary data processing based on these requests to facilitate the display of information on the client interface. Data exchange between the client and server occurs through well-defined APIs (Application Programming Interface).

Figure 4.1 Show the picture of the Client-Server architecture

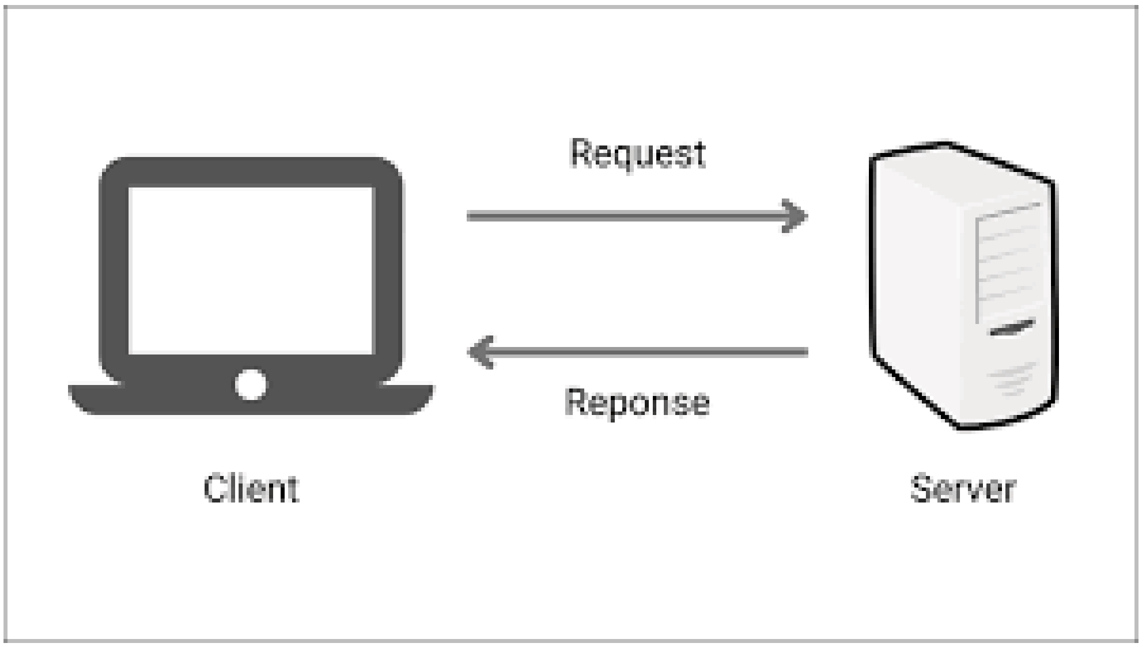


Figure 4.1: Client-Server

Specifically, when a user initiates an action on the client interface, the client sends the corresponding request to the server via the provided API. The server then processes the request, which might involve database access or storage. After completing the processing, the server sends the result back to the client, which then displays the result on the user interface.

The client-server architecture offers several benefits: (i) remote access - using the client-server model, users can remotely access data, send and receive files, or easily search for information, (ii) the model can prevent network congestion, (iii) it ensures data integrity during incidents, and (iv) it can function as long as a common communication protocol is used, without needing a shared platform. Hence, it is chosen to develop the system. And in this particular system, the client consists of a management web page and a mobile app used for customer order placement.

4.1.2 Overall design

a) Web Package Design

Figure 4.2 shows an overview structure for web client's package design.

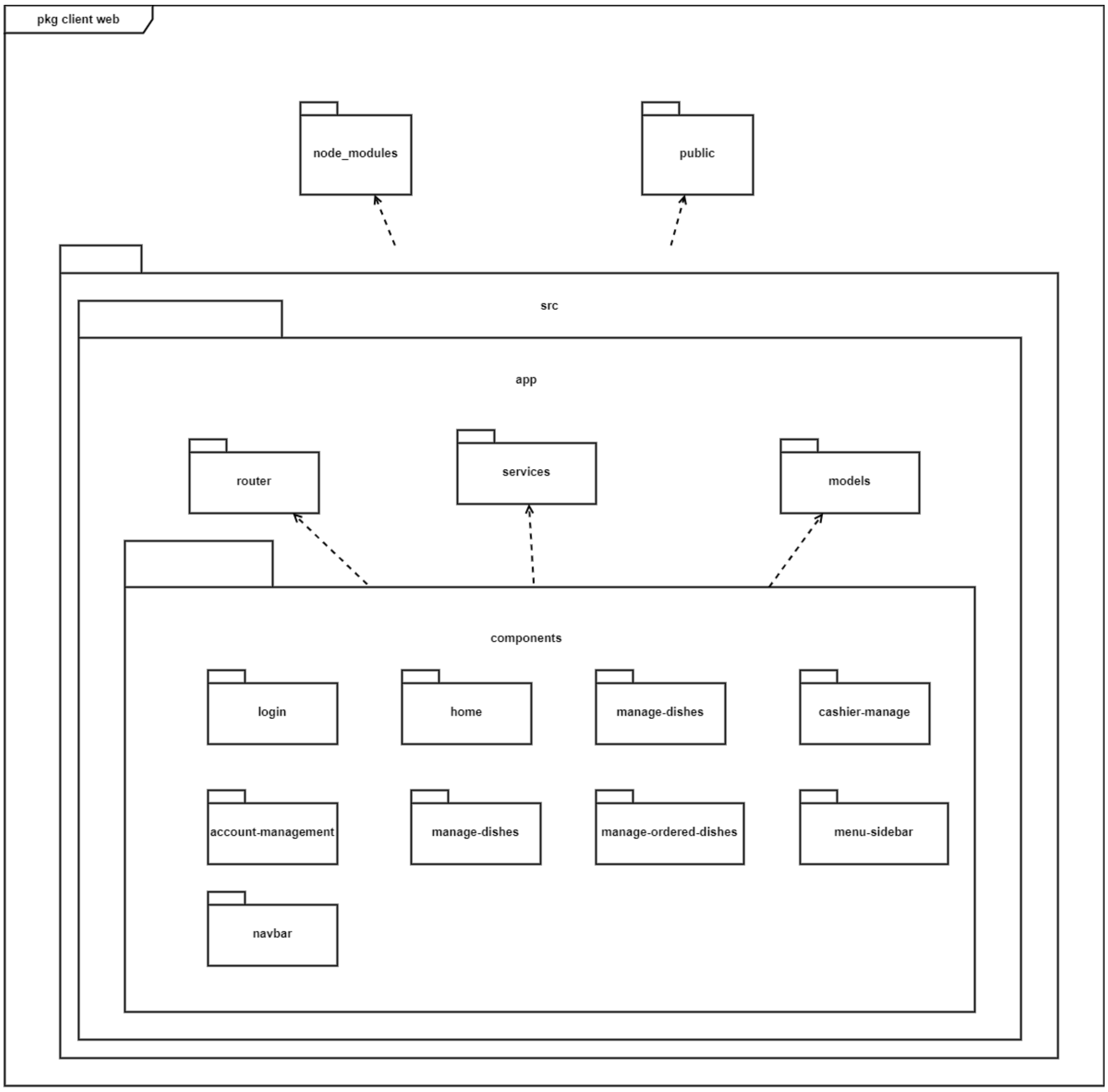


Figure 4.2: Web general package design

`components` contain all the interface components of the web.

`router` contain all the routes of the web.

`services` contain all the request methods to server.

`modules` contain all the interfaces use in this web client.

`nodes_modules` contain all the libraries and dependencies that this web client use

b) Mobile Package Design

Figure 4.3 shows an overview structure for mobile package design.

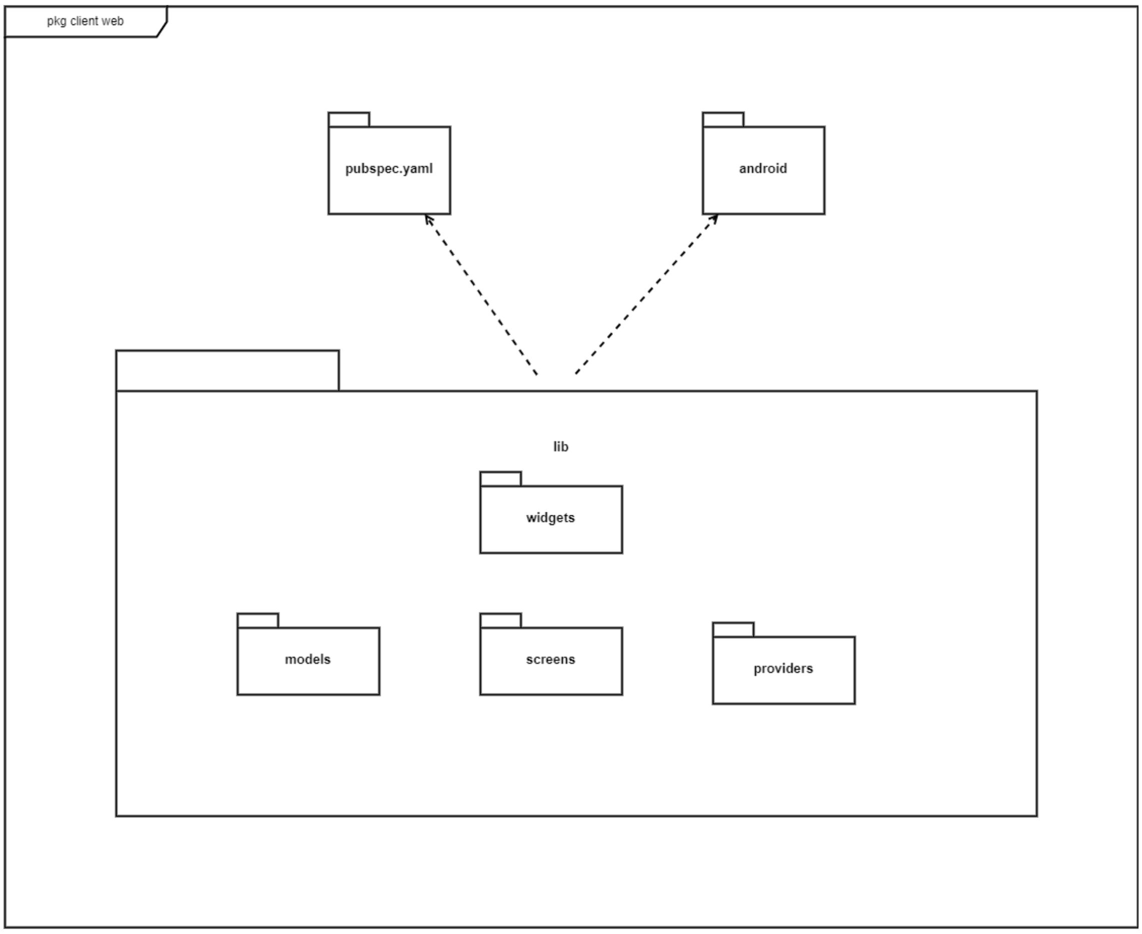


Figure 4.3: Mobile general package design

`android` contains the source code and resources needed to build Flutter applications for the Android platform.

`pubspec.yaml` contains all libraries and dependencies of this app.

`screens` contain all the widget screens that are used to show content.

`widgets` contain all the widget items used in screens to show content.

`models` contain all the entity classes used in this mobile app.

`providers` contains all the shared methods.

c) Backend Package Design

Figure 4.4 shows an overview structure for the backend's package design.

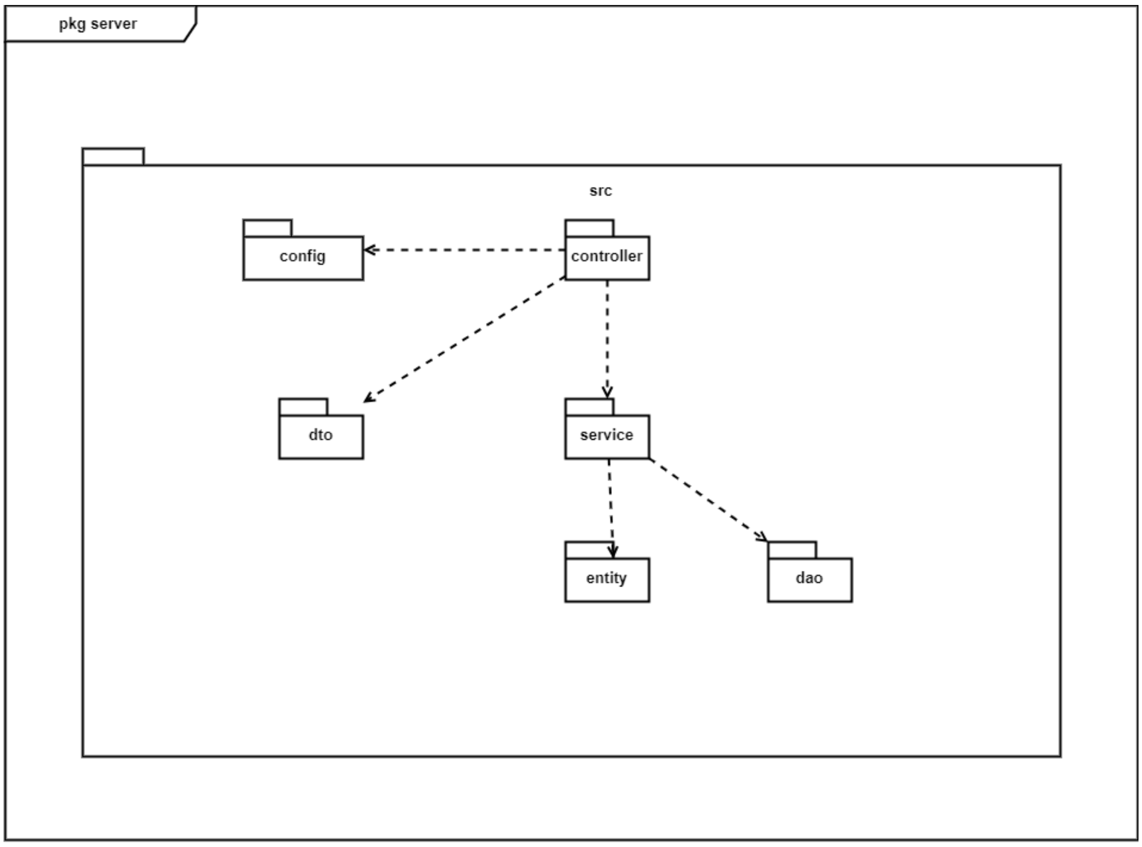


Figure 4.4: Backend general package design

`controller` contains all the controller classes that are responsible for receiving requests from the client and forwarding them to the service for processing

`service` contains business logic methods for the system.

`Entity` contains multiple classes that define the data structure.

`dto` contains multiple classes that define structures of data exchanged between the client and the server.

`dao` is the package that directly handles data operations with the database.

4.1.3 Detailed package design

The application is built based on a client-server architecture, where each object and operation is implemented on both the client and server sides. On the client side, each object and operation consists of three main parts: components, services, and models. On the server side, each object and operation is structured around five main files: (i) controller, (ii) entity, (iii) service, (iv) DTO (Data Transfer Object), and (v) DAO (Data Access Object). Some typical detailed package designs of the system are described in Figure 4.5, and Figure 4.6.

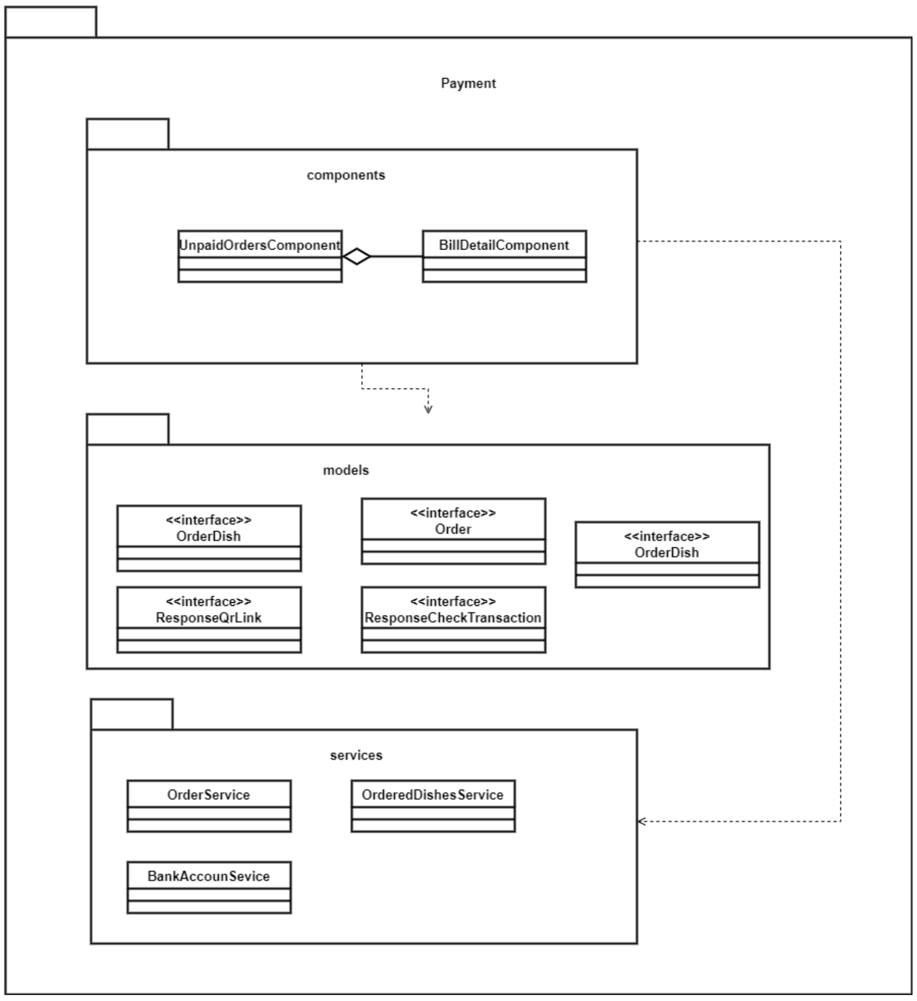


Figure 4.5: Detailed package for payment in web client

Figure 4.5 is the detailed package for payment in the web client. There are three packages: (i) components, (ii) services, and (iii) modules.

The `components` package includes two classes: (i) UnpaidOrdersComponent - to display a list of orders that have been placed and are awaiting payment, and (ii) BillDetailComponent - to display details of an order.

The `services` package contains three classes: (i) OrderService class - responsible for handling order creation requests and fetching lists of orders and order details from the server, (ii) OrderDishesService class - responsible for handling requests to fetch the list of dishes included in an order, and (iii) BankAccountService class - responsible for handling requests to generate QR codes and process payments.

The `models` package contains 5 interfaces used to configure objects in the classes within the components and services packages.

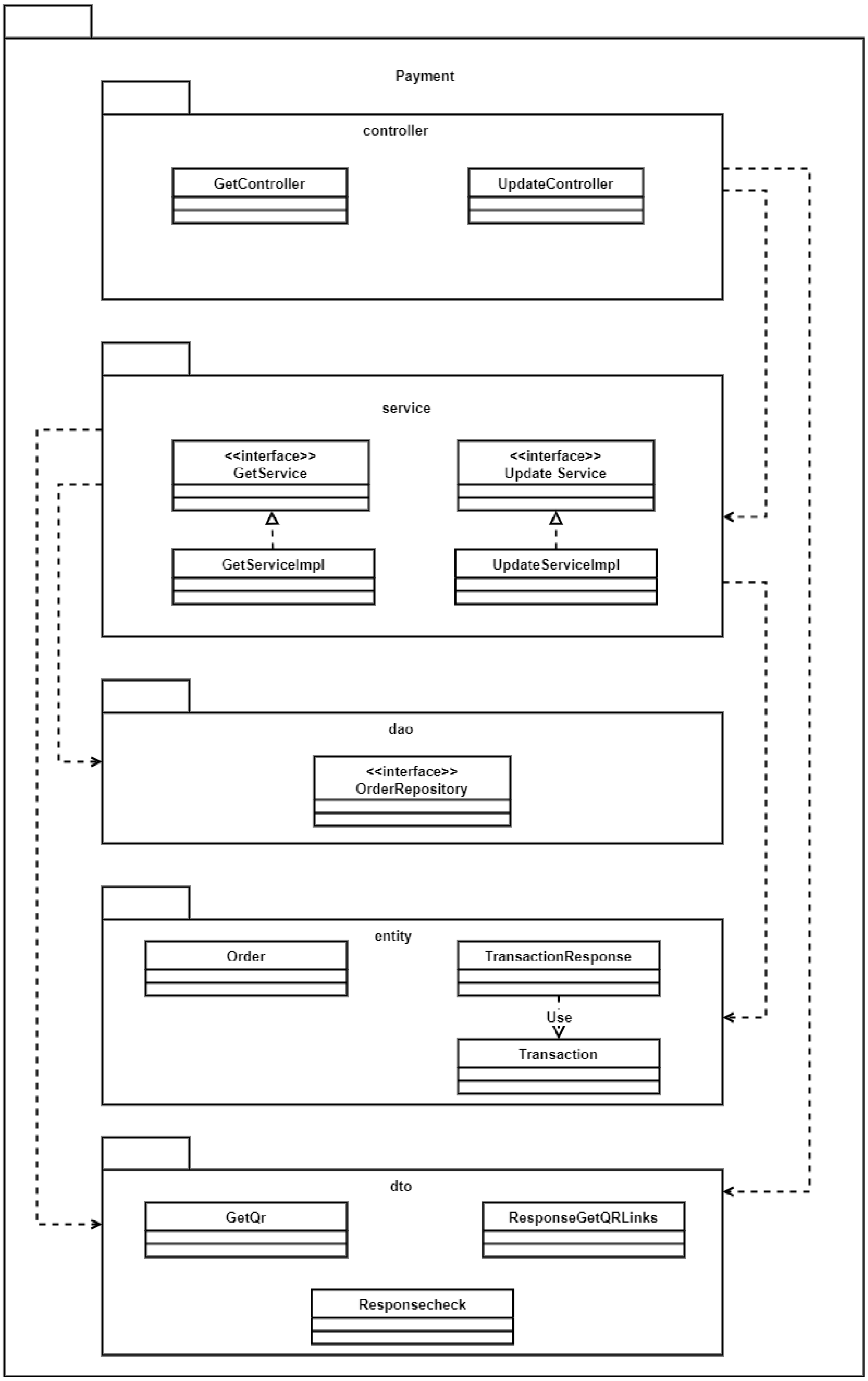


Figure 4.6: Detailed package for payment in backend

Figure 4.6 is the detailed package for payment in the backend. There are five packages: (i) controller, (ii) services, (iii) dao, (iv) entity, and (v) dto.

The `controllers` package includes 2 classes: (i) GetController class - which provides an API to get QR codes, and (ii) UpdateController class - which provides an API to process payments.

The `service` package includes 2 classes that are implementations of two interfaces: (i) GetServiceImpl class - contains methods to handle QR code retrieval logic, and (ii) UpdateServiceImpl class - contains methods to handle payment logic and update orders upon successful payment.

The `dao` package includes the OrderRepository interface to access and modify data in the database.

The `entity` package contains the Order, TransactionResponse, and Transactions classes used in the payment authentication methods in the UpdateService.

The `dto` package contains the GetQr, ResponseGetQRLinks, and ResponseCheck classes to configure the data passed to the API and the data returned from the API used in the Service and Controller classes.

4.2 Detailed design

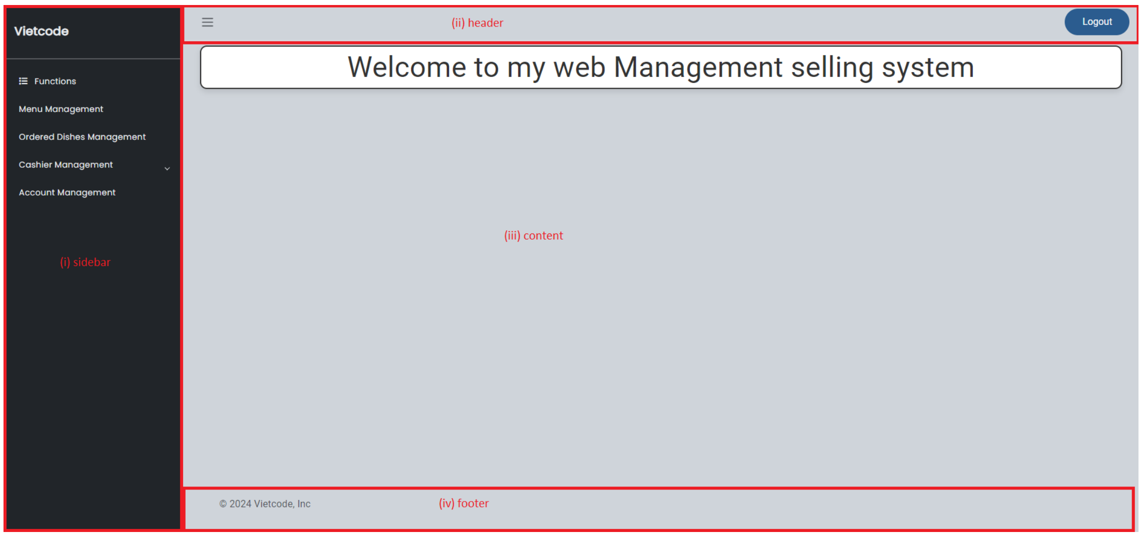
4.2.1 User interface design

a) web interface design

The web interface is crafted to be optimal for both laptop and desktop screens. The interface should present all essential information to the user while concealing unnecessary details. For data that occupies a large display area, the information can be temporarily hidden or revealed only upon user request. The application interface is also designed to be highly interactive, providing responsive feedback when users engage with the system.

The selling management system for restaurant uses a color scheme tailored to different types of data and contexts of use. For example, blue is used for adding new data, yellow for editing or updating, and red for deleting data. The application also employs user-friendly color tones, avoiding the overuse of anxiety-inducing colors like red and yellow.

Regarding the overall interface design, the structure of all pages will include four main components: (i) sidebar - the navigation menu for service functions. (ii) header - contains login, logout, expand and close sidebar button. (iii) content - the main content of the page. (iv) footer - the bottom part of the page. The layout of the application interface is presented in Figure 4.7.



Welcome to my web Management selling system

Figure 4.7: Layout of page

b) Mobile interface design

The mobile interface is optimized for smartphones. It presents all the necessary information clearly while minimizing unnecessary details. For data that requires a large display area, information can be collapsed or expanded based on user interaction. The mobile interface is designed to be highly responsive, ensuring smooth and intuitive feedback as users interact with the system.

The mobile interface of the restaurant's point of sale management system uses a color palette appropriate to a variety of data types and usage contexts. For example, purple-black is used to add new data, yellow to edit or update, and red to end an order or to delete something. The application uses gentle tones, avoiding using too many colors that can cause anxiety such as bright red or yellow. This ensures a pleasant user experience while maintaining functionality. Regarding the overall interface design, the structure of all pages will include two main components: (i) appbar which contain name of the screen and some action if have. (ii) content which í the main content ò the screen. The layout of the application interface is presented in Figure 4.8.

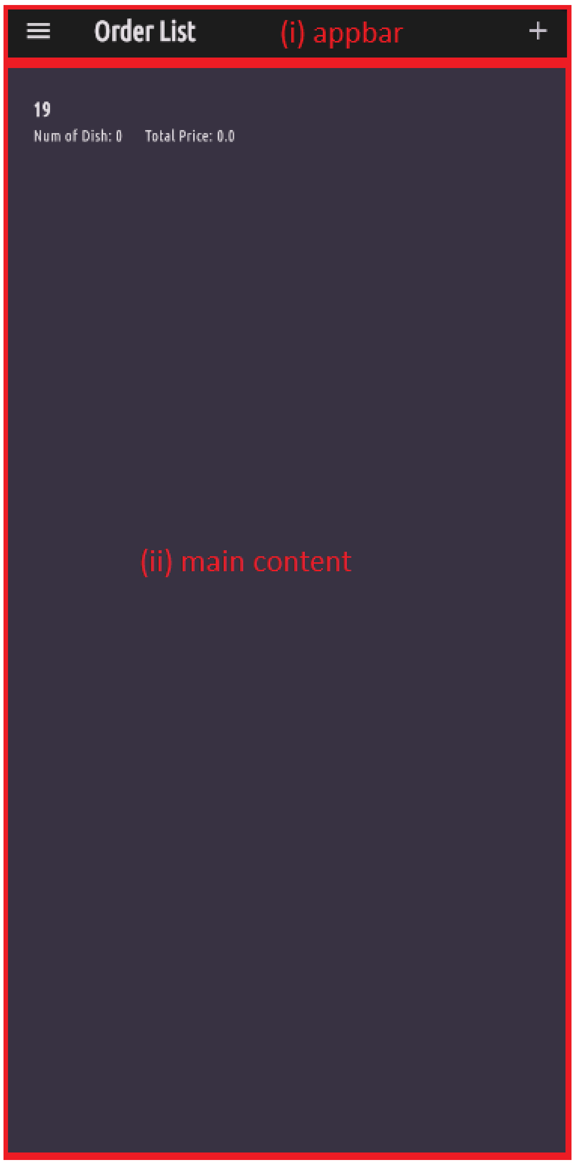


Figure 4.8: Layout of screen

4.2.2 Layer design

Figure 4.9 and Figure 4.10 present the layer design of two important layers for the application.

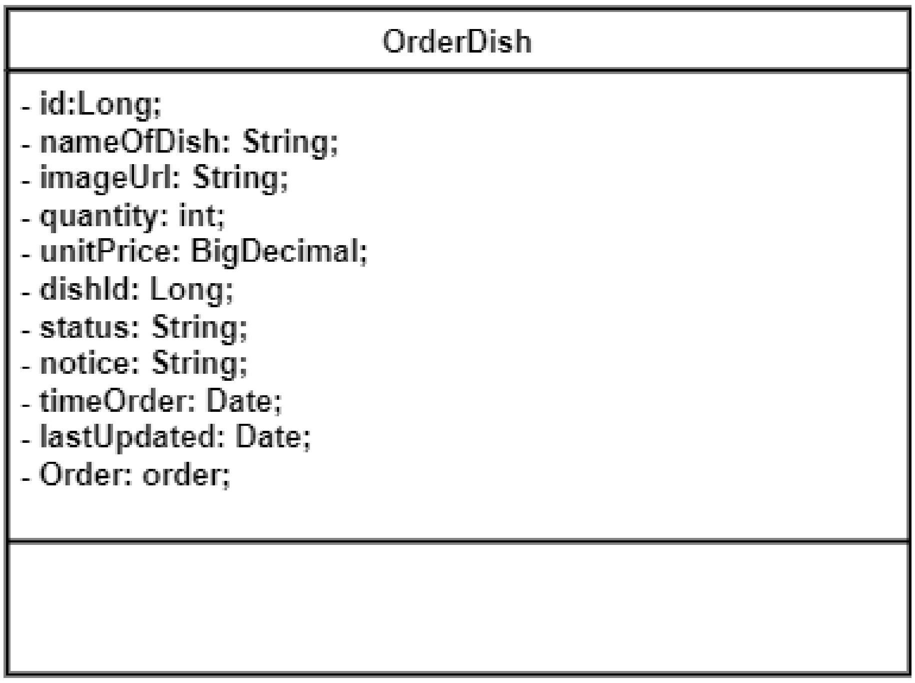


Figure 4.9: Detailed "OrderDish" class

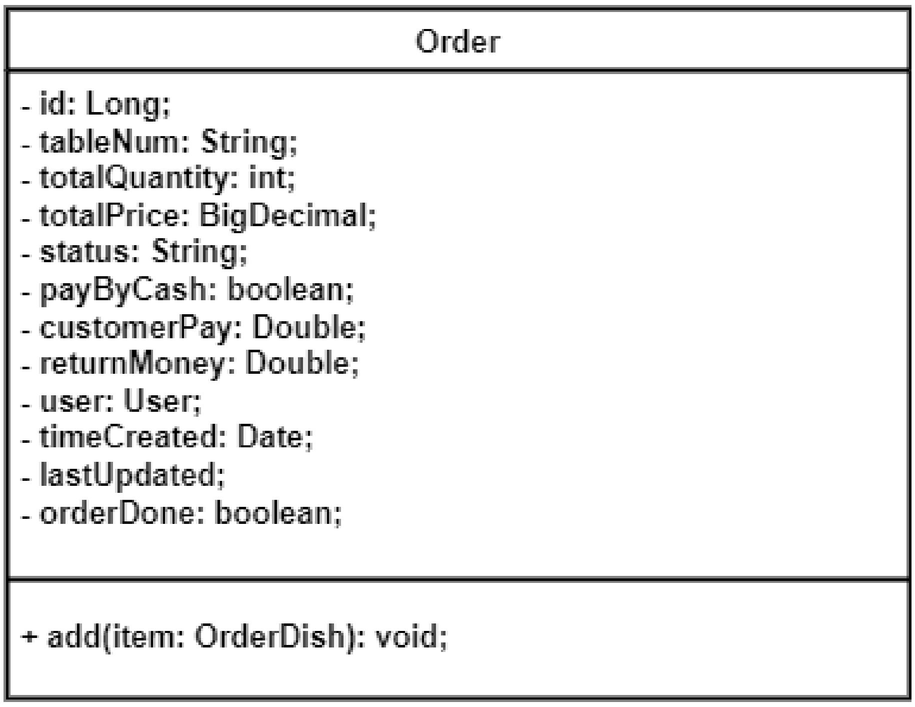


Figure 4.10: Detailed "Order" class

Figure 4.11 shows the sequence diagram of getting all OrderDish

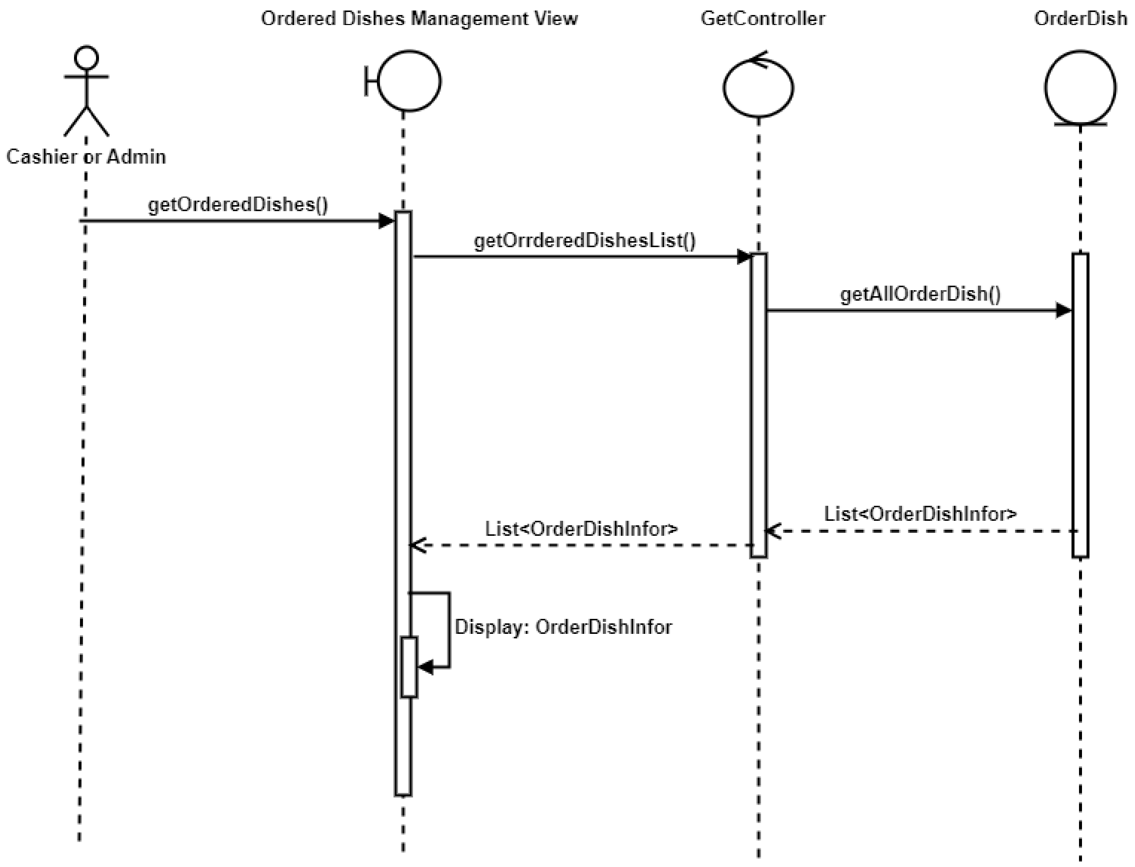


Figure 4.11: Sequence diagram of get all OrderDish

Figure 4.12 shows the sequence diagram of payment proccess

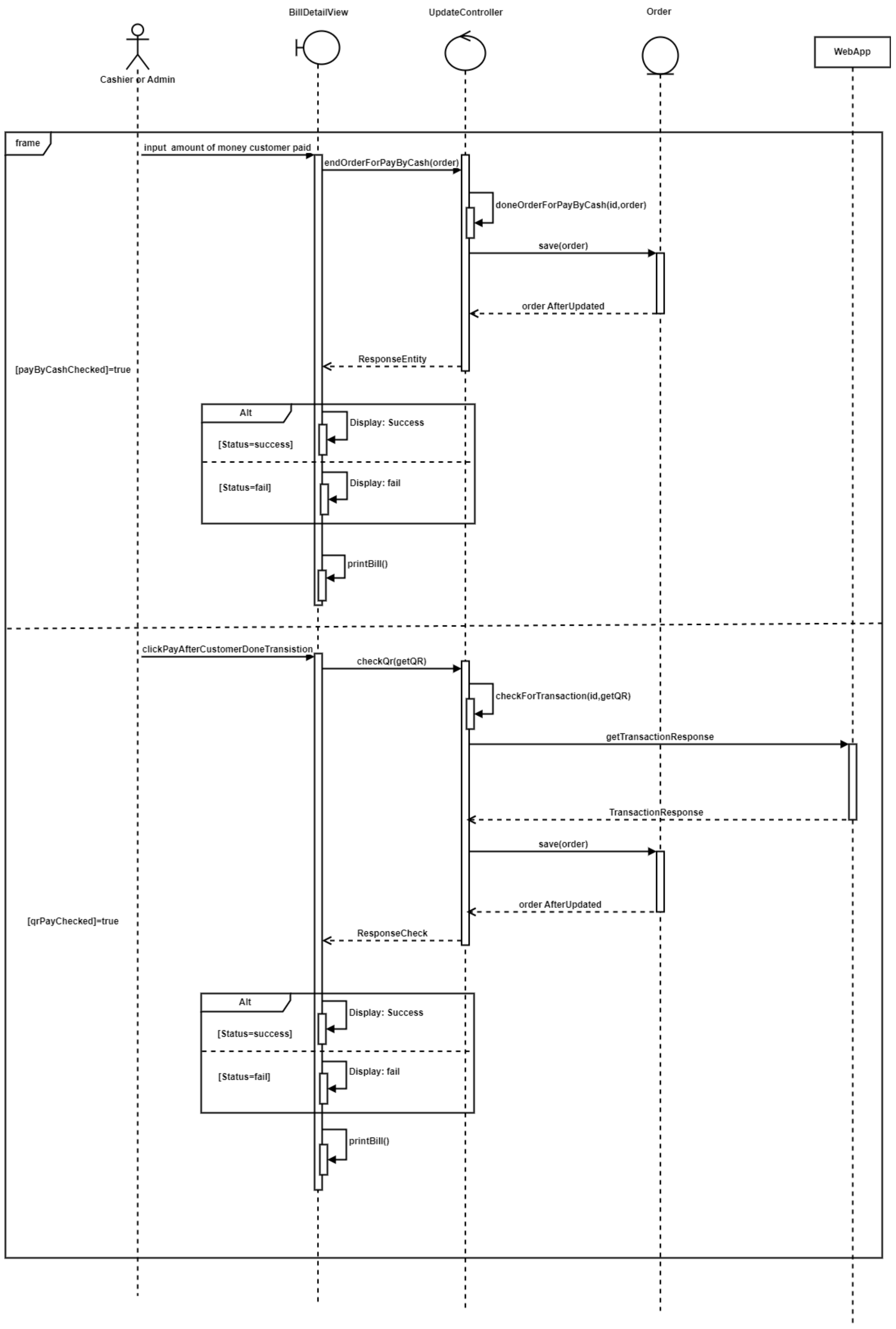


Figure 4.12: Sequence diagram of payment process

4.2.3 Database design

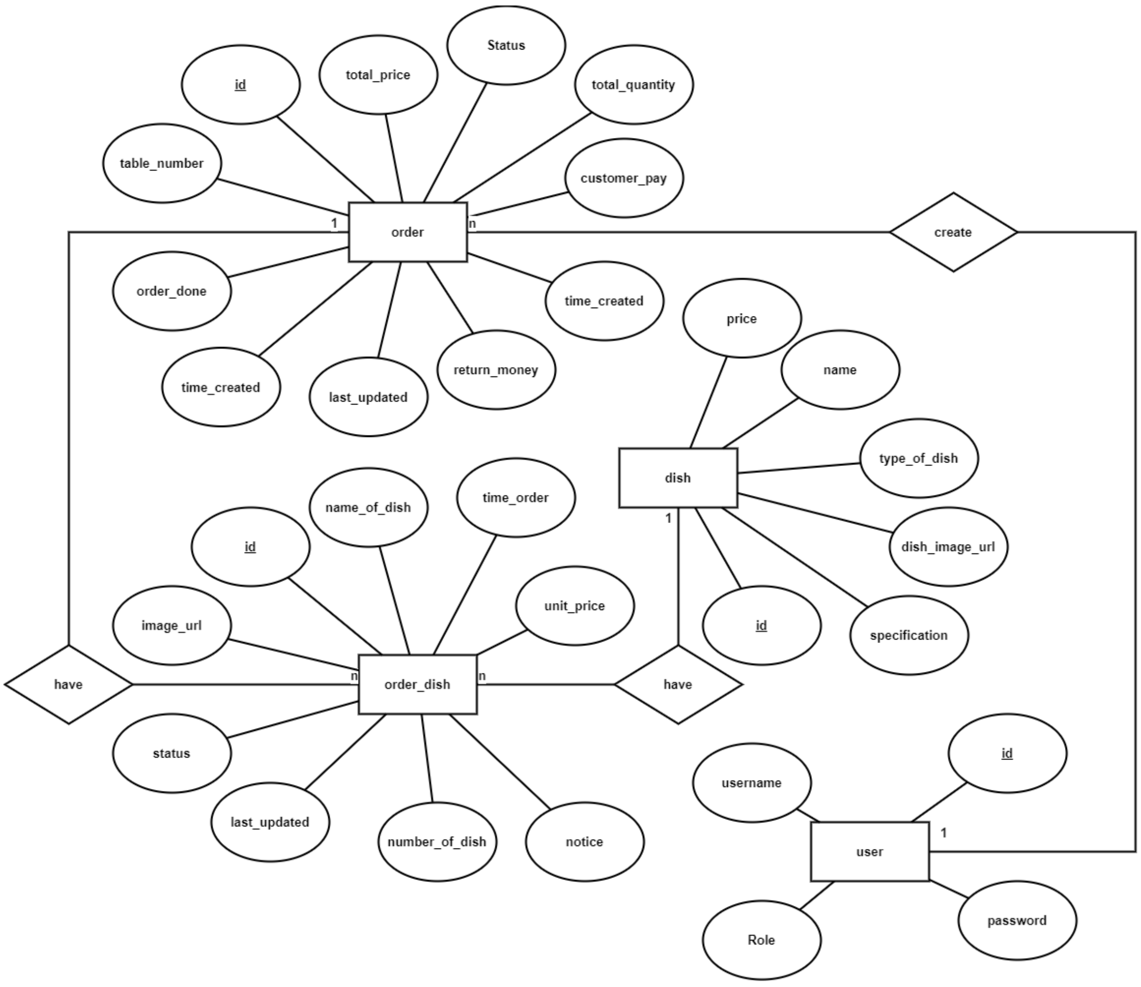


Figure 4.13: Entity Diagram

Figure 4.13 is the entity relationship diagram, there are 4 entities: (i) user, (ii) order, (iii) order_dish, and (iv) dish. Their relationship is as follows: a user can create one or more orders and inside there will be many order_dish and each order_dish will correspond to a dish.

Figure 4.14 is the database details diagram.

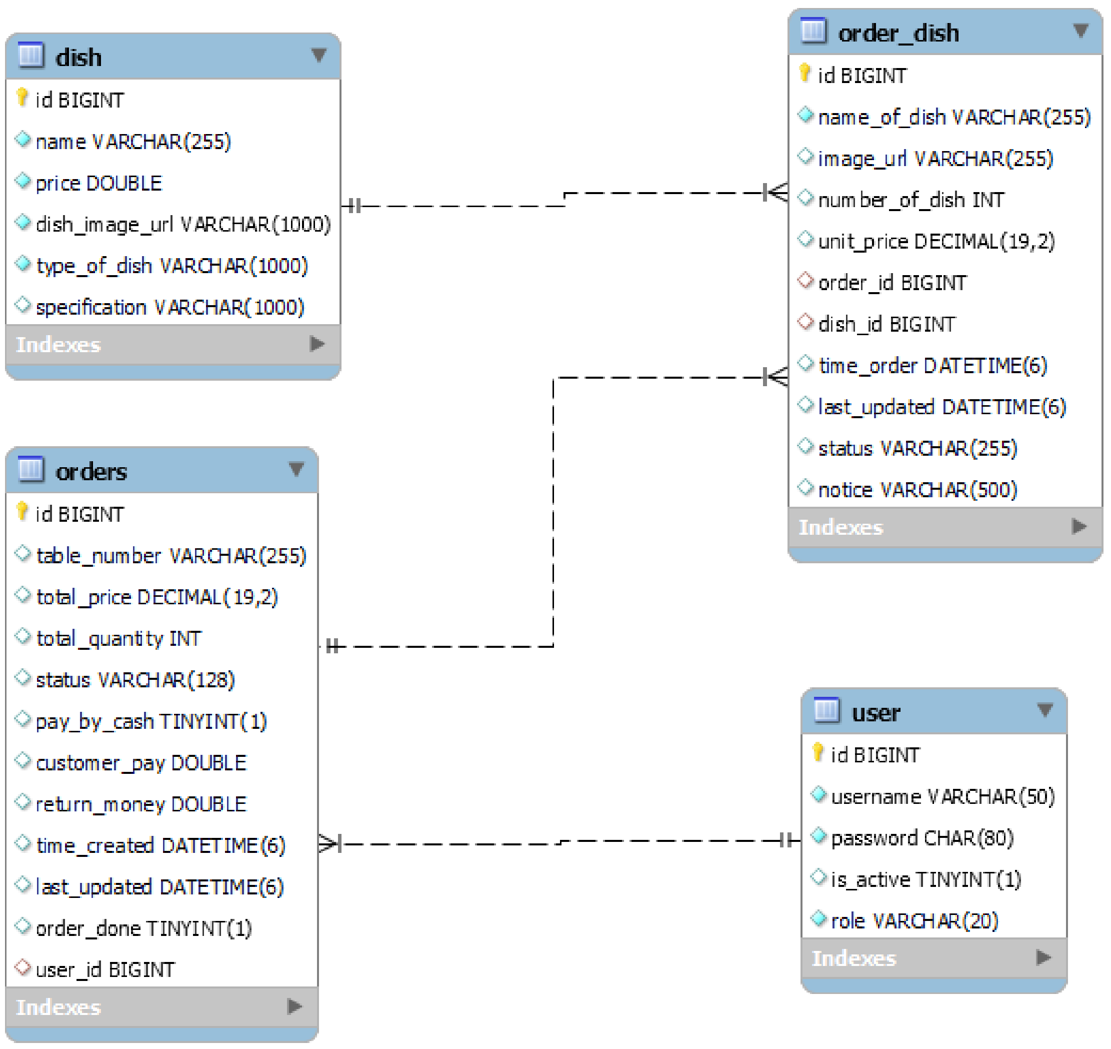


Figure 4.14: Database Details Diagram

Table 4.1 is details describe the properties of the table user.

Attribute Name	Types	Sizes	Constraint	Description
id	BIGINT	-	Primary key	Id of the user
username	VARCHAR	50	-	Username
password	CHAR	80	-	Password
is_active	BOOLEAN	-	-	is_active is true if user working at the store and false if not.
role	VARCHAR	20	-	Role of user in the restaurant

Table 4.1: "user" table detail

Table 4.2 is details describe the properties of the table dish.

Attribute Name	**Types**	**Sizes**	**Constraint**	**Description**
id	BIGINT	-	Primary key	Id of dish
name	VARCHAR	255	-	Name of dish
price	DOUBLE	-	-	Price of dish
dish_image_url	VARCHAR	1000	-	Url to image of dish
type_of_dish	VARCHAR	1000	-	Type of dish
specification	VARCHAR	1000	-	Information about the dish

Table 4.2: "dish" table detail

Table 4.3 is details describe the properties of the tables orders

Attribute Name	**Types**	**Sizes**	**Constraint**	**Description**
id	BIGINT	-	Primary key	Id of order
table_number	VARCHAR	255	-	Table number of order
total_price	DECIMAL	-	-	Total price of order
total_quantity	INT	-	-	Total number of dishes in order
status	VARCHAR	128	-	Status of order
pay_by_cash	BOOLEAN	-	-	If pay_by_cash is true, order is pay by cash and false if pay by QR Pay
customer_pay	DOUBLE	-	-	Amount of money that customer pay if they pay by cash
return_money	DOUBLE	-	-	Amount of money return to customer if they pay over the total price
time_created	DATETIME	-	-	Time that order created
last_updated	DATETIME	-	-	Time that order is updated
order_done	BOOLEAN	-	-	True if customer have done eating, and false if they are still eating
user_id	BIGINT	-	Foreign key	Id of the user who create this order.

Table 4.3: "orders" table detail

Table 4.4 is details describe the properties of the tables order_dish

Attribute Name	**Types**	**Sizes**	**Constraint**	**Description**
id	BIGINT	-	Primary key	Id of order_dish
name_of_dish	VARCHAR	255	-	Name of dish of the order_dish
image_url	VARCHAR	1000	-	Image of dish in the order_dish
number_of_dish	INT	-	-	Number of this dish is ordered
unit_price	DECIMAL	-	-	Price of this dish
order_id	BIGINT	-	Reference key	Id of order of this order_dish
dish_id	BIGINT	-	Reference key	Id of dish of this order_dish
time_order	DATETIME	-	-	Time this dish is order
last_updated	DATETIME	-	-	Time that status of order_dish is updated
status	VARCHAR	255	-	Status of this order_dish such as preparing,
notice	VARCHAR	500	-	Notice for chef when cooking this dish

Table 4.4: "order_dish" table detail

4.3 Application Building

4.3.1 Libraries and Tools

Table 4.5 is the table showed the list of libraries and dependencies that are used in this system.

Purpose	**Tool & Libaries**	**Url**
IDE for develop Back-end	IntelliJ IDEA	jetbrains.com/idea/
IDE for develop Font-end and Mobile	Visual Studio Code	code.visualstudio.com/
Build mobile app	Flutter	flutter.dev
Build API	Springboot Framework	spring.io/projects/spring-boot
Build web interface	Angular	angular.dev/
Feature to build Front-end	Boostrapt	getbootstrap.com/
Feature to build Front-end	Angular Material	material.angular.io/
Database	MySQL	mysql.com/
Store shared values	shared_preferences	https://pub.dev/packages/shared_preferences
Android virtual device	Android Studio	developer.android.com/studio
Font in mobille	google_fonts	pub.dev/packages/google_fonts
Request and recive response to server in mobile	http	pub.dev/packages/http
Develop a method to use between screens	riverpod	riverpod.dev/
Java annotation library which helps to reduce boilerplate code.	Lombok	projectlombok.org/
Build Authentication	Json Web token	jwt.io/
Mapping data in the database in Back-end	Hibernate	hibernate.org/
Call api to get Data in Backend	Apache HttpClient	hc.apache.org

Table 4.5: Libraries and tools in thesis

4.3.2 Achievement

Through research and analysis, I have built a selling management system for restaurant with functions The main functions are (i) User Management, (ii) Menu Management, (iii) Ordering Management, (iv) Cashier Management, (v) Ordered Dish Management.

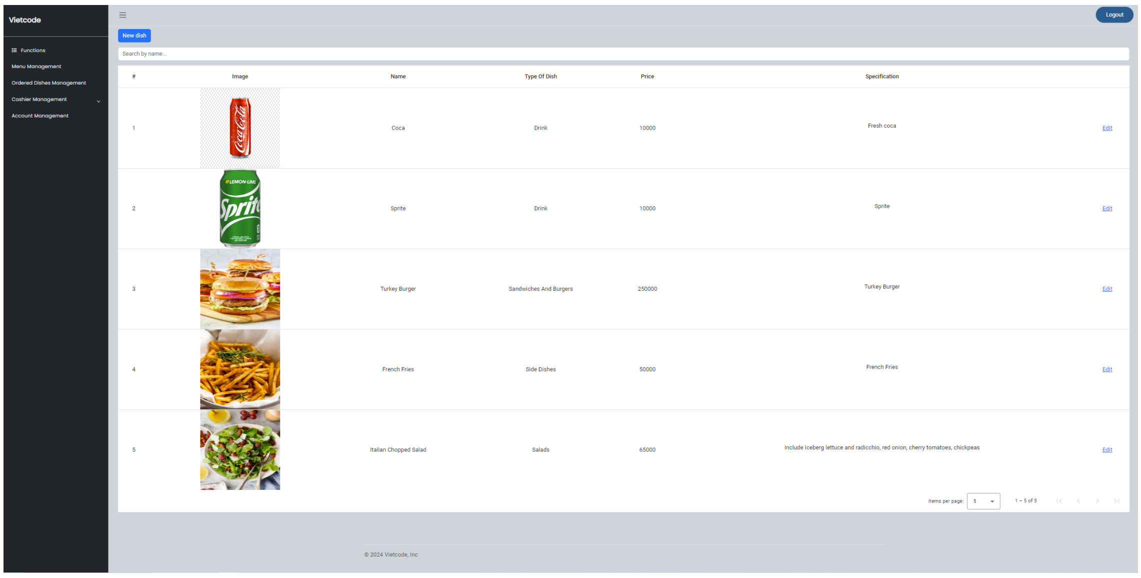
Table 4.6 show the application specification of the system.

Infomation	**Specification**
Front-end	Source lines of code: 4081 Number of files: 108 Total file size: 222 KB
Back-end	Source lines of code: 1631 Number of files: 42 Total file size: 99.9 KB
Mobile	Source lines of code: 2845 Number of files: 23 Total file sizes: 136 KB

Table 4.6: Application specification

4.3.3 Illustration of main functions

Figure 4.15 is the screen for menu mangement web screen. In this screen, user can click new dish to go to new dish button form screen. Moreover, user can click edit to go to edit form to edit and in that edit form user can update dish information or delete dish. Not only that user can also search for dish by its dish name.



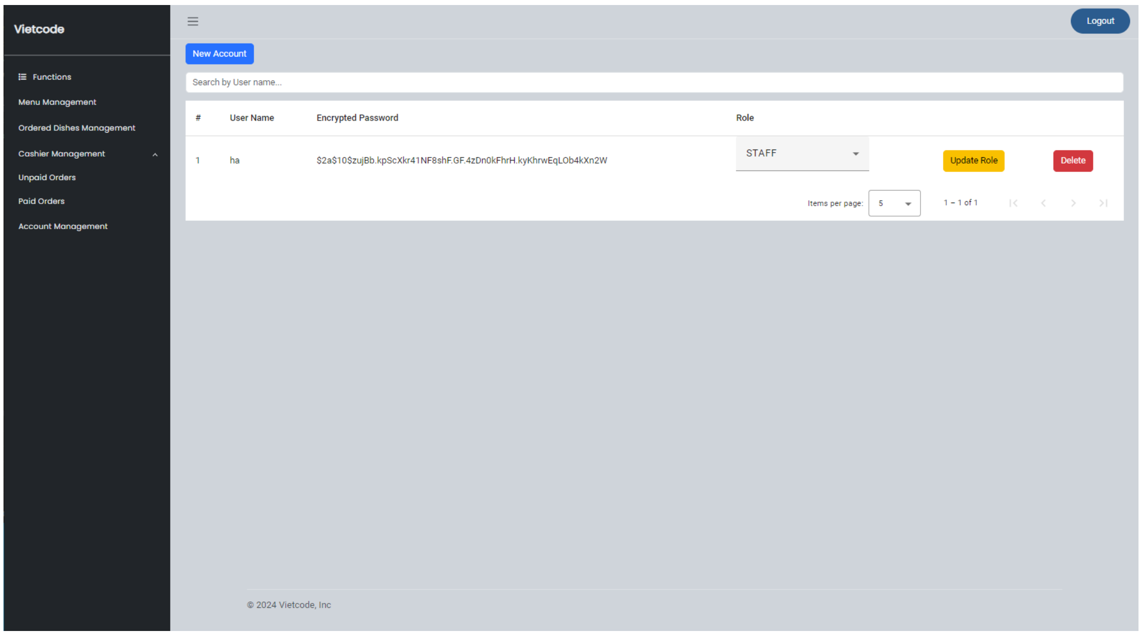
#	Image	Name	Type Of Dish	Price	Specification	
1		Coca	Drink	10000	Fresh coca	Edit
2		Sprite	Drink	10000	Sprite	Edit
3		Turkey Burger	Sandwiches And Burgers	250000	Turkey Burger	Edit
4		French Fries	Side Dishes	50000	French Fries	Edit
5		Italian Chopped Salad	Salads	65000	Include Iceberg lettuce and radicchio, red onion, cherry tomatoes, chickpeas	Edit

Items per page: 5 1 – 5 of 5

© 2024 Vietcode, Inc

Figure 4.15: Menu Management web screen

Figure 4.16 is the screen for account mangement web screen. In this screen, user can click new account button to go to new account form screen. Moreover, user can change the role and click update button to change the role of user. Not only that, user can also delete user by click delete button



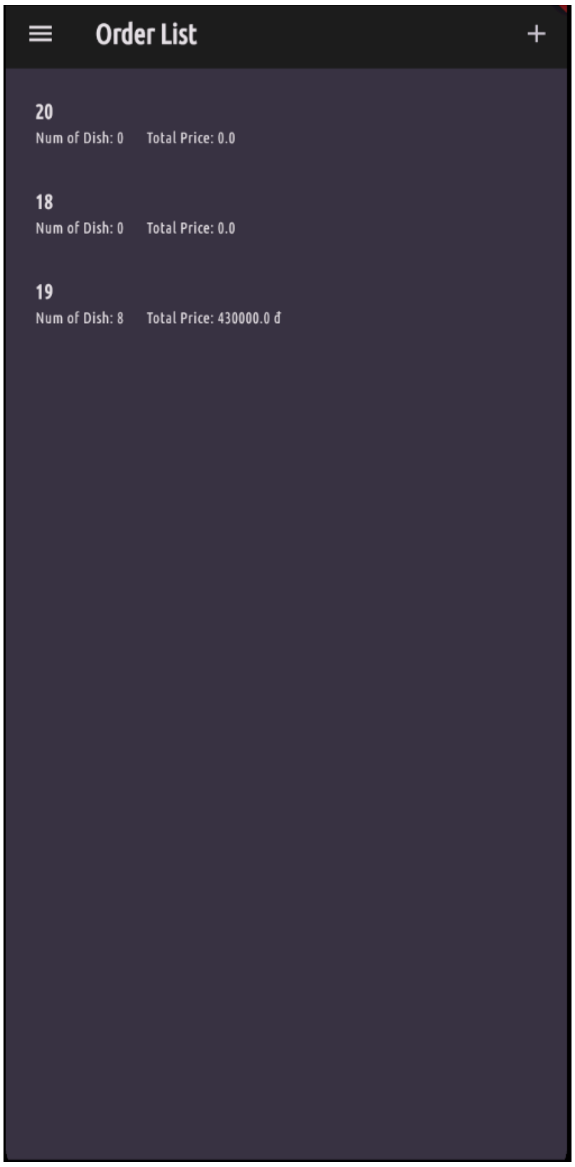
#	User Name	Encrypted Password	Role		
1	ha	$2a$10$zujBb.kpScXkr41NF8shF.GF.4zDn0kFhrH.kyKhrwEqLOb4kXn2W	STAFF	Update Role	Delete

Items per page: 5 1 – 1 of 1

© 2024 Vietcode, Inc

Figure 4.16: Account Management web screen

Figure 4.17 is the mobile Order List screen. In this screen, user can see a list of orders from tables with customers who have not finished eating and user can also add new order by click plus button to go to add new order form. Moreover, user can delete order if order don't have any dish by swiping order to from right to left.



20
Num of Dish: 0 Total Price: 0.0

18
Num of Dish: 0 Total Price: 0.0

19
Num of Dish: 8 Total Price: 430000.0 đ

Figure 4.17: Order list screen

Figure 4.18 is the mobile Order Deatails screen. In this screen, user can see information about order and user can also add new dish to order by click Add Dishes to Order button to go to Ordering Screen. Moreover, user can delete order of dish if dish hasn't been prepare by swiping order of dish to from right to left. Not only that, user can change the table number if customer want to change table to sit by clicking Change Order Information button.

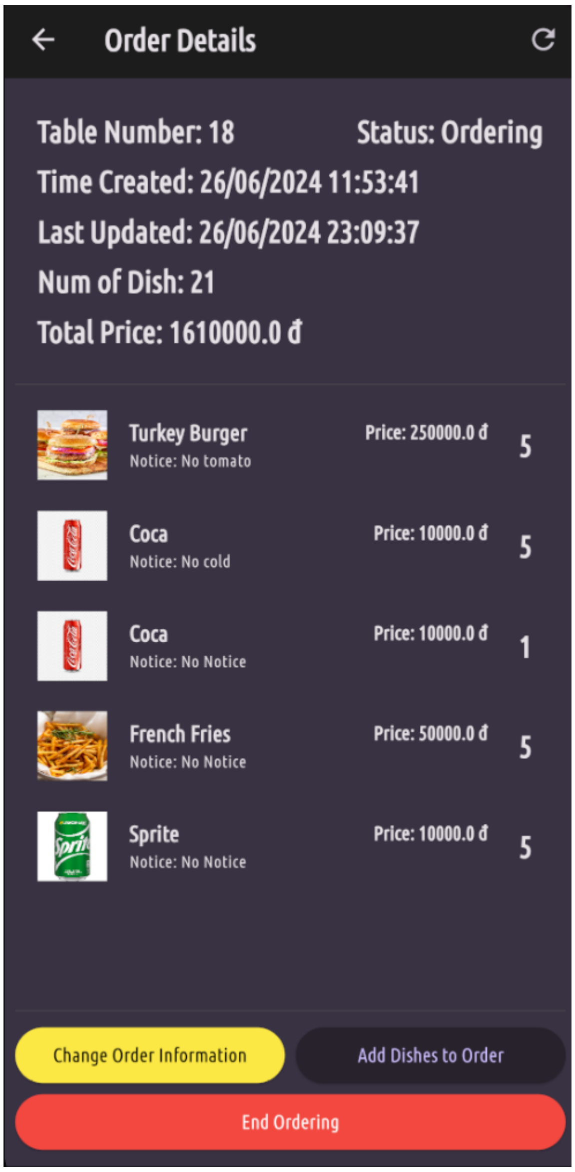


Table Number: 18 Status: Ordering
Time Created: 26/06/2024 11:53:41
Last Updated: 26/06/2024 23:09:37
Num of Dish: 21
Total Price: 1610000.0 đ

Turkey Burger
Notice: No tomato
Price: 250000.0 đ
5

Coca
Notice: No cold
Price: 10000.0 đ
5

Coca
Notice: No Notice
Price: 10000.0 đ
1

French Fries
Notice: No Notice
Price: 50000.0 đ
5

Sprite
Notice: No Notice
Price: 10000.0 đ
5

Figure 4.18: Order detail screen

4.4 Testing

This system using black box testing and the below is the table of test for some main functions:

Table 4.7 is the test table for the use case 'Ordering Management' with the pre-condition that the user is logged in with the role of admin or staff.

Function	**Input**	**Output**	**Result**
Create order	Order table number	New order show	Pass
Delete order	Swipe order from right to left	Order is deleted in the list of orders	Pass
Update order	Order table number	Table number of order updated	Pass
Add new order_dish to order	A list of order_dish	new order_dish show	Pass
Delete order_dish of order	Swipe order_dish from right to left	order_dish is deleted in the list of order_dish in order detail screen	Pass

Table 4.7: Test for "Ordering managment"

Table 4.8 is the test table for use case 'Menu Management' with the pre-condition that the user is logged in with the role of admin.

Function	**Input**	**Output**	**Result**
Add new dish to menu	Dish information	New dish show in the list of dishes	Pass
Update dish information	Updated information of the dish	Dish information is updated and show in the list of dishes	Pass
Delete dish	Click delete in the edit dish form	Dish is deleted from the list of dishes	Pass

Table 4.8: Test for "Menu managment"

Table 4.9 is the test table for use case 'User Management' with the pre-condition that the user is logged in with the role of admin.

Function	**Input**	**Output**	**Result**
Add new user to system	user information	New user show in the list of users	Pass
Update user role	Updated role of user that want to updated in the list of user and click update	Role is updated and show in the list of users	Pass
Delete user	Click delete in the row of user information	User is deleted from the list of users	Pass

Table 4.9: Test for "User Management"

Table 4.10 is the test table for use case 'Cashier Management' with the pre-condition that the user is logged in with the role of admin or cashier.

Function	**Input**	**Output**	**Result**
See list of unpaid order	Click on unpaid order in the cashier management dropdown list in the side bar	Show list of unpaid order	Pass
See detail of unpaid order	Click on the unpaid order	Show detail information of order	Pass
Payment for pay by cash	Click the radio button of pay by cash in the detail of unpaid order page and fill the total number and click pay	Show success message and navigate to list of unpaid orders and print biill	Pass
Payment for pay by QR Code	Click the radio button of pay by QR code in the detail of unpaid order page and wait for customer scan the QR code then click pay	Show success message and print bill	Pass
See list of paid order	Click on paid order in the cashier management dropdown list in the side bar	Show list of paid order	Pass
See detail of paid order	Click on the paid order	Show detail information of paid order	Pass

Table 4.10: Test for "Cashier Management"

4.5 Deployment

The application is run directly on the local host and Android virtual machine. To run the application locally, you need to perform the following steps:

(I) Install the environment and configure application components such as Maven, NPM, MySQL, Angular CLI, Flutter SDK, and Git.

(ii) Install Android Studio to create a virtual device for running the mobile app.

(iii) Clone the application's source code.

(iv) Create a database in MySQL.

(v) Configure the server environment to connect to the database.

(vi) Install libraries in the mobile code.

(vii) Run the application.

CHAPTER 5. SOLUTION AND CONTRIBUTION

5.1 Optimizing restaurant ordering and service processes at a low cost

5.1.1 Problem

In section 1.2, the thesis has pointed out significant limitations of existing restaurant management software on the market such as CukCuk, PosApp, SapoFnb, and similar solutions. Although these software provides powerful and diverse features, they are not suitable for the needs and financial capabilities of small and medium-sized restaurants.

The main reason is the high operating and maintenance costs of these systems. Many solutions require investment in specialized equipment, which is not only initially expensive but also demands regular maintenance to ensure stable operation. This creates a significant financial barrier for restaurants with limited resources.

As a result, many small and medium-sized restaurants are forced to maintain rudimentary management systems, mainly based on paper. Especially in the process of taking orders and serving customers, the use of paper order slips is not only inefficient but also prone to errors. For example, staff may lose order slips, leading to incomplete or inaccurate service according to customer requests.

Recognizing this challenge, the thesis aims to develop a solution to optimize the restaurant ordering and service process at a low cost. The focus of the research is to create a system that small and medium-sized restaurants can easily apply to improve their ordering and service processes while minimizing investment and operating costs.

By focusing on developing a solution that is both effective and economical, the thesis aims to bridge the technology gap between large and small restaurants, creating opportunities for small businesses to improve service quality and compete more effectively in the food industry.

5.1.2 Solution

This graduation thesis has developed a comprehensive solution to optimize the restaurant ordering and service process while keeping costs low through the creation of a multifunctional web app. The core of the system is a mobile application specially designed for waitstaff, allowing them to quickly and accurately record orders right at the customer's table.

The online ordering system includes a detailed electronic menu integrated into the mobile app. This menu not only displays the list of dishes but can also include images, descriptions, and pricing information, helping waitstaff easily introduce and record customer choices. Using staff's personal mobile devices not only helps save on equipment investment costs but also increases flexibility in the service process.

When an order is recorded, the system automatically transmits information to the kitchen in real-time. This helps minimize waiting time and errors in the information transmission process. The kitchen staff can monitor and manage incoming orders through an intuitive interface, helping them arrange and prioritize dishes efficiently.

Another important feature of the system is notifying waitstaff when dishes are ready. This ensures that food is served promptly, maintaining optimal quality and temperature of the dishes, while improving the overall customer experience.

Regarding payment methods, the thesis has integrated a QR payment system (QRPay), an increasingly popular electronic payment solution. This not only brings convenience to customers but also helps restaurants minimize risks associated with cash handling and increase transaction speed.

The system allows staff and restaurant managers to have an overview of table status in real-time through the feature of reviewing paid orders. This helps optimize seating arrangements, reduce customer waiting times, and increase the efficiency of restaurant space utilization.

Technically, the thesis has chosen to use modern and powerful open-source technologies. Angular was chosen as the framework for web front-end development, providing a solid foundation for building dynamic and responsive user interfaces. For mobile applications, Flutter is used, allowing cross-platform app development with beautiful user interfaces and high performance. On the back-end, Java Spring Boot is applied, bringing stability, security, and high scalability. MySQL database was chosen for data storage and management, providing a reliable and cost-effective solution. This combination creates a comprehensive technology stack, meeting the thesis requirements while maintaining low development and operational costs.

The use of open-source technologies not only helps reduce initial development costs but also facilitates future system maintenance and upgrades. This is particularly important for small and medium-sized restaurants where IT budgets may be limited.

In summary, this graduation thesis has created a comprehensive, integrated, and cost-effective solution to optimize the restaurant ordering and service process. By combining modern technology with a deep understanding of the needs of the food service industry, this system promises to enhance operational efficiency, improve customer experience, and ultimately increase revenue for restaurants.

5.1.3 Result

The implementation of the solution developed in this graduation thesis has brought significant improvements to the restaurant ordering and service process. The multifunctional web app system, with its focus on a mobile application for waitstaff, has optimized workflows and reduced operational costs.

Waitstaff can quickly and accurately record orders right at the customer's table through a detailed electronic menu on the mobile app. This not only improves work efficiency but also increases flexibility in the service process, while saving on equipment investment costs by utilizing staff's personal mobile devices.

Figure 5.1 shows the order list screen where in there have electronic menu.

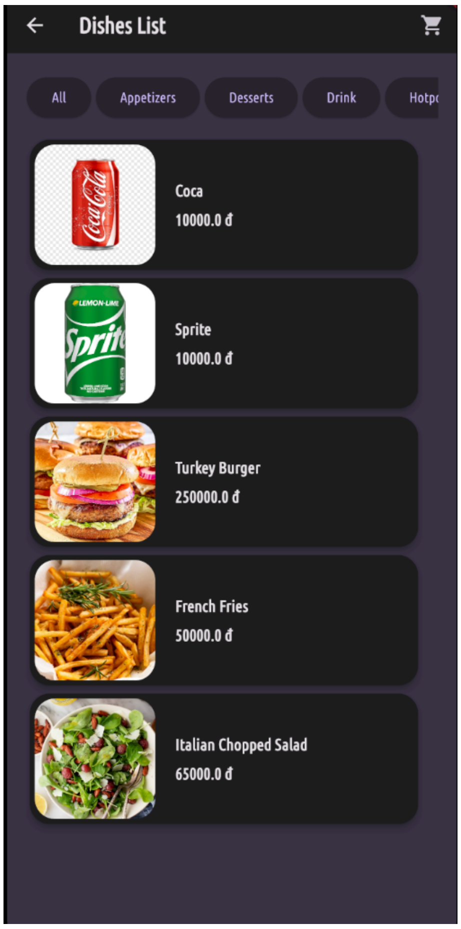
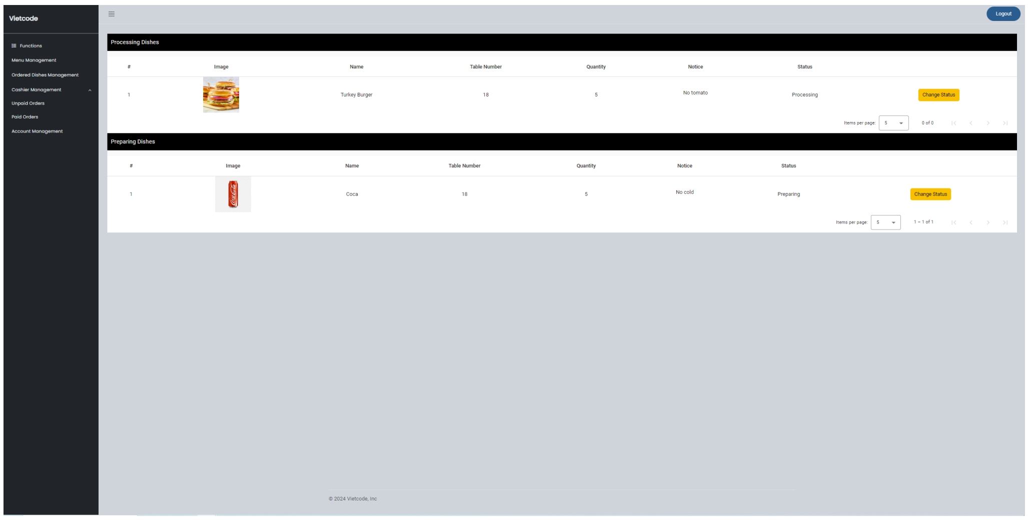


Figure 5.1: Ordering Screen

The kitchen process has been significantly improved with a system that automatically transfers order information in real-time. The kitchen staff can manage orders more effectively through an intuitive interface, helping to minimize waiting times and errors. The feature that notifies when dishes are ready ensures food is served promptly, maintaining optimal quality and temperature, enhancing the customer experience.

Figure 5.2 shows the Ordered Dish Management screen which is used by the Kitchen supervisor.



Processing Dishes

#	Image	Name	Table Number	Quantity	Notice	Status	
1		Turkey Burger	18	5	No tomato	Processing	Change Status

Items per page: 5 0 of 0

Preparing Dishes

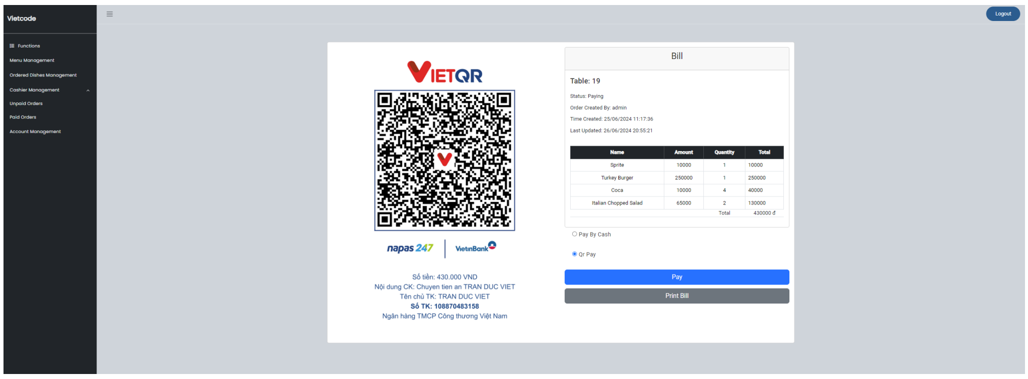
#	Image	Name	Table Number	Quantity	Notice	Status	
1		Coca	18	5	No cold	Preparing	Change Status

Items per page: 5 1 – 1 of 1

Figure 5.2: Ordered Dishes Management Screen

The integration of the QR payment system (QRPay) has brought convenience to customers while helping restaurants minimize risks and increase transaction speed. The system also allows managers and staff to have a real-time overview of table status, optimizing seating arrangements and increasing space utilization efficiency.

Figure 5.3 shows the order detail screen for unpaid order and payment



Số tiền: 430.000 VND
Nội dung CK: Chuyen tien an TRAN DUC VIET
Tên chủ TK: TRAN DUC VIET
Số TK: 108870483158
Ngân hàng TMCP Công thương Việt Nam

Bill

Table: 19

Status: Paying

Order Created By: admin

Time Created: 25/06/2024 11:17:36

Last Updated: 26/06/2024 20:55:21

Name	Amount	Quantity	Total
Sprite	10000	1	10000
Turkey Burger	250000	1	250000
Coca	10000	4	40000
Italian Chopped Salad	65000	2	130000
		Total	430000 đ

Pay By Cash [✓ Qr Pay]

Figure 5.3: Unpaid Order Detail Screen

Technically, the use of modern open-source technologies such as Angular, Flutter, Java Spring Boot, and MySQL has created a comprehensive technology stack that meets thesis requirements with low development and operational costs. This is particularly suitable for small and medium-sized restaurants where IT budgets may be limited.

In conclusion, this solution has succeeded in optimizing the ordering and service process, improving customer experience, and increasing restaurant operational efficiency. By combining modern technology with a deep understanding of the food service industry's needs, the system has created a solid foundation for enhancing performance and increasing revenue for restaurants.

5.2 Payment using QR Code

5.2.1 Problem

In Section 1.2, the graduation thesis emphasized that the system will provide a seamless payment function for restaurants. This raises the need to integrate an efficient, convenient, and low-cost payment method. In this context, QR code payment (QR Pay) emerges as an ideal solution, meeting the increasing trend of cashless payments in Vietnam and aligning with the national digital transformation strategy.

QR Pay brings significant benefits to both restaurants and customers. For restaurants, this method allows easy integration into existing management systems, minimizing investment costs for traditional POS terminals. It also helps optimize the payment process, reduces the burden of cash management, and limits risks associated with cash handling. For customers, QR Pay increases transaction speed, reduces waiting time, and improves the overall experience, while providing clear transaction history and better personal financial management capabilities.

Moreover, QR Pay creates opportunities for restaurants to integrate promotional and loyalty programs flexibly and effectively. Collecting data from QR Pay transactions also helps restaurants analyze customer behavior, thereby adjusting business strategies and improving services. With the increasing trend of smartphone usage in Vietnam, especially among the youth, integrating QR Pay helps restaurants demonstrate modernity and meet customer expectations for a dining experience combined with technology. Therefore, integrating QR Pay into the system to optimize the ordering and serving process is not only a necessary step but also a smart strategy to enhance operational efficiency and competitiveness in the food service industry.

5.2.2 Solution

The solution for implementing QR Pay into the restaurant management system is designed based on a sophisticated combination of VietQR technology and

Casso's service [8], aiming to create a seamless, efficient payment process that aligns with the digitalization trend in the food service industry. The system chose to use VietQR's QR Pay code after careful consideration among available options in the market. VietQR was preferred over other solutions like VNPay due to its flexibility, ease of deployment, and particularly its clear, detailed documentation on how to generate QR codes [9]. VietQR provides a simple URL that allows for quick QR code generation, only requiring adjustment of a few parameters to create QR codes suitable for each transaction. This enables the system to easily customize payment information such as amount and transfer content, meeting the diverse needs of restaurant transactions.

To address the crucial issue of confirming successful transactions, the system has integrated Casso's service - an advanced financial process automation tool. Choosing Casso resulted from the search for an effective intermediary solution capable of connecting with various banks without requiring complex legal and technical processes as would be needed when directly connecting with each bank. Casso plays a key role in linking the restaurant's bank account with the management system, allowing for automated and accurate real-time tracking of incoming transactions. This not only helps minimize errors in the payment confirmation process but also enhances transparency and efficiency in the restaurant's financial management.

The transaction information processing workflow is designed intelligently and efficiently. Casso exports transaction information to a Google Sheets spreadsheet, creating a flexible and easily manageable database. Next, the system leverages the power of Google Apps Script to create a compact yet powerful server capable of retrieving the two most recent transactions from the spreadsheet. This method not only simplifies the data processing but also optimizes performance by focusing on the latest transactions, helping to reduce response time and increase the speed of payment confirmation.

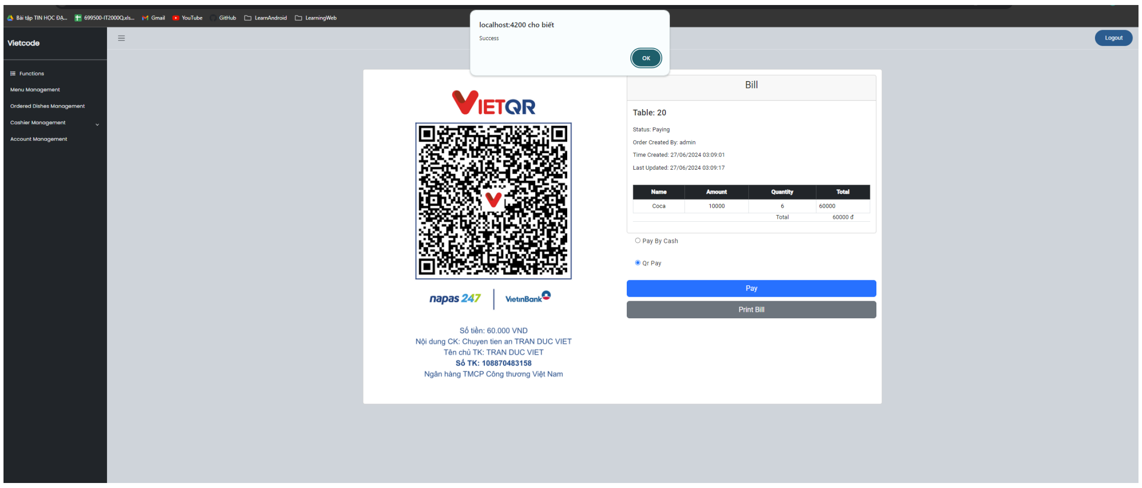
The final step in the process is payment verification, performed meticulously and accurately. The server returns transaction information to the restaurant management system, allowing the system to perform the reconciliation process. The system compares the total order amount with the transaction amount, while also checking the transfer content (pre-generated in the QR code) to confirm successful transactions. This process not only ensures transaction accuracy but also helps detect and promptly handle cases of unsuccessful payments or errors.

Overall, this solution allows restaurants to deploy QR Pay payments efficiently and cost-effectively, without requiring large investments in complex infrastructure or direct connections to banking systems. By leveraging existing tools like VietQR, Casso, and Google Sheets, the system creates an automated, secure, and easily manageable payment process. This not only significantly improves customer experience through quick and convenient payments but also helps restaurants optimize their financial management processes, enhancing operational efficiency and competitiveness in the increasingly digitalized foodservice market.

5.2.3 Result

After the system generates a QR code for a specific transaction, the payment process unfolds as follows: The customer uses their mobile banking application to scan the displayed QR code. This automatically fills in the necessary payment information such as the recipient's account number, amount, and transfer content into the customer's banking app. Once the customer confirms and completes the transaction on their application, the restaurant's cashier will proceed with the next step in the process. The cashier will click the "Pay" button on the restaurant's management system interface. This action triggers the automated payment verification process. The system will query information from the server created by Google Apps Script, retrieving data about the most recent transactions from the Google Sheets spreadsheet that has been updated by Casso. The system then compares the received transaction information with the order details, including the amount and transfer content. When the system confirms that a transaction matches the order information, it will display a successful payment notification on the cashier's screen. This is the signal for the staff member to know that the transaction has been verified and they can proceed with the final step of the process. After receiving the success notification, the cashier will print the invoice for the customer. This invoice will include details of the ordered items, the total amount, and confirmation that the payment has been successfully made via the QR Pay method. Printing the invoice not only provides the customer with proof of the transaction but also marks the completion of the payment process.

Figure 5.4 shows the success message when payment with the QRpay code is successful.



localhost:4200 cho biết

Success

Số tiền: 60.000 VND
Nội dung CK: Chuyen tien an TRAN DUC VIET
Tên chủ TK: TRAN DUC VIET
Số TK: 108870483158
Ngân hàng TMCP Công thương Việt Nam

Bill

Table: 20

Status: Paying

Order Created By: admin

Time Created: 27/06/2024 03:09:01

Last Updated: 27/06/2024 03:09:17

Name	Amount	Quantity	Total
Coca	10000	6	60000
		Total	60000 đ

Pay By Cash [✓ Qr Pay]

Figure 5.4: Success QR pay picture

This procedure ensures accuracy, safety, and efficiency in payment processing while providing a smooth and professional payment experience for the customer.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

My restaurant sales management system is the result of meticulous research and development. Although this system is still not on par with existing software due to its development by a non-professional team and its completion in a short period of time, it has successfully achieved the main functions I set out, including (i) User Management, (ii) Menu Management, (iii) Order Management, (iv) Cashier Management, and (v) Ordered Dish Management.

The standout feature of this system compared to other systems is its ability to optimize the ordering and service process of the restaurant at a low cost, making it especially suitable for small and medium-sized restaurants in Vietnam. If it continues to be developed in the future, this system could significantly improve the revenue of small-scale restaurants by enhancing efficiency and minimizing errors in the service process.

During the development process, I gained valuable experience in various areas. My skills in programming mobile and web applications were significantly enhanced as I learned to use Flutter and Dart for mobile development and Angular and Typescript for web development. This allowed me to better understand the integration between front-end and back-end systems.

Additionally, I developed strong project management and software development skills. I learned to effectively plan and manage time throughout the thesis, from defining initial requirements and system design to testing and deployment. Each phase required careful attention and effective management skills.

However, my graduation thesis still has some limitations and shortcomings such as: (i) the web and mobile interfaces are not fully polished and aesthetically pleasing, (ii) the system does not yet have real-time data update features to automatically refresh the interface when data changes, and (iii) currently, JWT is only created upon login but is not yet authorized at the backend, so the APIs are still publicly accessible.

6.2 Future work

Certainly. Here's the updated paragraph translated into English: In the future, the restaurant sales management system will continue to be developed to optimize the ordering and service process at a low cost. The main development directions include:

(i) Enhancing security by building middleware for authentication and authorization using JWT and implementing secure APIs.

(ii) Develop real-time features to instantly update the interface when data changes occur.

(iii) Optimizing for mobile devices by developing responsive interfaces and dedicated applications.

(iv) Improving system deployment and management processes.

(v) Integrating direct payment with banks for automatic transaction verification.

(vi) Optimizing the ordering and service process by developing an online ordering system, implementing smart notification systems, and integrating AI to predict ordering demands.

(vii) Developing data analysis and reporting tools to monitor performance and support better business decisions.

(viii) Adding a return function for unused beverages and items, allowing customers to easily return items they don't need and receive a refund or exchange for other items.

These improvements will enhance customer experience, increase revenue, and boost profits for restaurants through automation and improved operational efficiency. The system will focus on streamlining the ordering process, reducing waiting times, improving order accuracy, and optimizing kitchen and staff management. By leveraging AI and real-time data, the system will help restaurants predict customer needs, optimize ingredient preparation, and allocate staff resources more effectively. The integration of direct banking connections will eliminate reliance on third-party services, reducing costs and improving financial management. Overall, these future developments aim to create a comprehensive, efficient, and cost-effective solution for restaurant management.

The newly added return function will increase flexibility for customers, improve their satisfaction, and help restaurants manage inventory more efficiently. This can also help reduce waste and enhance the overall customer experience.

REFERENCE

[1] *Introduction to java*. [Online]. Available: https://www.oracle.com/java/ (visited on 06/28/2024).

[2] *Java spring boot-java web application framework*. [Online]. Available: https://spring.io/projects/spring-boot (visited on 06/28/2024).

[3] *Introduction to typescript*. [Online]. Available: https://www.typescriptlang.org/docs/ (visited on 06/28/2024).

[4] *Angular-web application framework for front-end*. [Online]. Available: https://angular.dev/overview (visited on 06/28/2024).

[5] *Introduction to dart*. [Online]. Available: https://dart.dev/guides (visited on 06/28/2024).

[6] *Introduction to flutter framework*. [Online]. Available: https://flutter.dev/learn (visited on 06/28/2024).

[7] *Mysql-an open-source relational database management system*. [Online]. Available: https://dev.mysql.com/doc/ (visited on 06/28/2024).

[8] *Introduction to casso*. [Online]. Available: https://docs.casso.vn/ (visited on 06/28/2024).

[9] *Vietqr-document*. [Online]. Available: https://www.vietqr.io/en/intro (visited on 06/28/2024).