

University of Science and Technology of Hanoi



A REPORT ON

Point of Sale system with integrated payment gateway and face recognition

Group Project

BY

Le Minh Hoang	<i>BI12-167</i>
Nguyen The Hoang	<i>BI12-171</i>
Nguyen Minh Hoang	<i>BI12-172</i>
Nguyen Vu Viet Hoang	<i>BI12-173</i>
Le Tuan Huy	<i>BI12-195</i>
Tran Ngoc Anh	<i>BI12-040</i>

Under the guidance of

MSc.Huynh Vinh Nam

*

December 2023

Contents

1	Introduction	3
1.1	History of Transaction	3
1.2	Why is POS machine ?	4
1.3	Statistics	5
2	Business Analysis	7
2.1	The current state of other applications	7
2.2	Project Objective	8
2.3	Project Scope	9
2.4	Tasks Assign	9
3	Methodology	10
3.1	Functional requirement	10
3.1.1	List of Actors	10
3.1.2	Featuring functions	11
3.1.3	Functionality relationship matrix	12
3.1.4	Facial recognition	13
3.2	Use-case	14
3.2.1	Use-case diagram	14
3.2.2	Use-case description	14
3.3	Diagrams	23
3.3.1	Sequence Diagrams	23
3.3.2	Database diagrams	27
3.4	System Architecture	28
3.4.1	Design	28

3.4.2	The connection and data	28
3.4.3	Client	30
3.4.4	Application	30
3.4.5	Other workers and third-party	31
3.5	Tools and Technical choices	32
3.5.1	Client-Server Model	32
3.5.2	Github	33
3.5.3	MongoDB NoSQL	33
3.5.4	Coding languages	33
3.5.5	REST API technology	35
3.5.6	S3	35
3.5.7	Redis	36
3.5.8	Socketti	36
3.5.9	face-api.js	36
4	Implementation	37
4.1	Back-end	37
4.1.1	Model	37
4.1.2	Resources	38
4.1.3	Controller	38
4.1.4	Face Recognition back-end	40
4.1.5	Handy POS Scanning application	42
4.1.6	Freshly generated QR	43
4.1.7	Events	43
4.2	Front-end	44
4.2.1	POS machine screen	44
4.2.2	POS customer screen	46
4.2.3	POS manager screen:	47
4.3	Side services	50
4.3.1	Banking Integration	50
4.4	Web Application	51
4.4.1	Managing the data	51
4.4.2	Creating the interfaces	51

5	Result	52
5.1	Applied functions	52
6	Conclusions	53
6.1	Conclusions	53
6.2	Future works	54
7	References	56
8	Appendix	57
8.1	CRUD diagrams	57
8.2	Metadata tables	61

Auxiliary verb

- POS: Point-of-sale
- CRUD: Create, Read, Update and Delete
- ATM : Automated teller machine
- EFTPOS : Electronic Funds Transfer at Point of Sale
- EDC : Electronic capture Device

Chapter 1

Introduction

1.1 History of Transaction

The history of transactions spans a long period of human civilization, evolving alongside the development of various forms of currency, trade, and economic systems. Here's a brief overview of the history of transactions:

1. **Barter System (Prehistoric Times):** In ancient societies, people engaged in barter, exchanging goods and services directly without a standardized medium of exchange. This system had limitations, as it required a double coincidence of wants.
2. **Introduction of Money (Ancient Civilizations):** Coins are widely used among various civilizations, including Greeks, Romans, Chinese as a medium of exchange.
3. **Medieval Trade and Banking (Middle Ages):** Use of promissory notes and bills of exchange. Merchants and traders used promissory notes and bills of exchange to facilitate transactions over long distances.
4. **Banking and Bills of Exchange (Renaissance):** Bills of exchange became widely used, allowing merchants to make payments without the need for physical currency.

5. **Introduction of Paper Money (17th Century):** Governments and banks started issuing paper money backed by a commodity like gold or silver.
6. **Industrial Revolution (18th-19th Centuries):** With increased production and urbanization, the volume and complexity of transactions grew. Financial institutions and stock exchanges were established.
7. **Advent of Electronic Transactions (20th Century):** The mid-20th century saw the introduction of electronic transactions, initially through the use of credit cards.
8. **Development of Digital Transactions (Late 20th Century to Present):** Online banking, e-commerce, and electronic funds transfer systems became commonplace. Debit and credit cards, as well as digital wallets, became popular for both in-person and online transactions.

1.2 Why is POS machine ?

The evolution of transactions, from bartering to digital currencies, highlights the need for efficient and secure payment methods. Point of Sale (POS) machines address several modern requirements in the following ways:

- **Electronic Payments:** In a digital age, POS machines enable businesses to accept electronic payments, including credit and debit cards. This accommodates the preference of many customers who no longer rely solely on cash.
- **Convenience:** POS machines offer a convenient and quick way for customers to make purchases. Electronic transactions streamline the checkout process, enhancing the overall shopping experience.
- **Record-Keeping:** POS systems automatically record transactions, providing businesses with accurate and real-time data. This helps in inventory management, sales tracking, and financial reporting.
- **Integration with Technology:** POS machines can integrate with other technologies, such as inventory management systems and customer relationship management (CRM) software. This integration improves overall business efficiency.

- **Security:** POS machines incorporate encryption and security features to protect sensitive financial information. This is crucial for safeguarding customer data and maintaining trust in electronic transactions.
- **Efficiency in Business Operations:** POS systems help businesses streamline operations by automating tasks such as inventory tracking, sales reporting, and employee management. This leads to improved efficiency and reduced manual errors.

In summary, POS machines play a crucial role in the modern business landscape by providing a secure, efficient, and convenient means of processing transactions. They bridge the historical evolution of transactions with contemporary needs and technological advancements, contributing to the smooth functioning of businesses and meeting the expectations of both businesses and consumers.

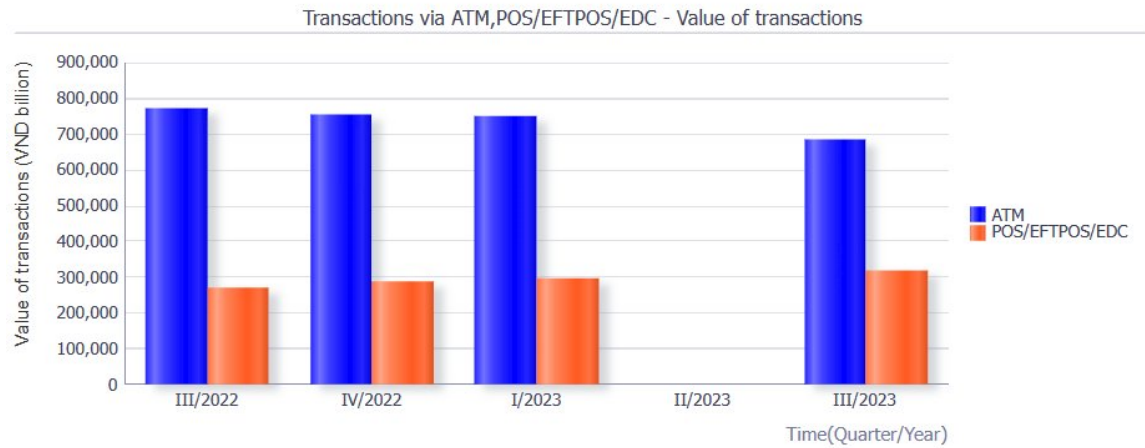
1.3 Statistics

Terminal	Number of Terminal	Number of Transaction(items)	Value of Transactions (billion Vnd)
ATM	21,079	241,906,017	683,079
POS/EFTPOS/EDC	471,973	189,672,620	317,738

Table 1.1: Q3/2023 Transaction Table

Statistics on the number and value of transactions via ATM, POS/EFTPOS/EDC of the reporting credit institutions during the reporting period, including:

- Cash withdrawals
- Account transfers, including: fund transfer, bill payment, payment for the purchase of goods and services via ATM, POS/EFTPOS/EDC;
- Other transactions, such as: term deposit; loans repayment, interest payment and fees,... between credit institutions and customers.



The chart above displays briefly the relative comparison between the total value of transaction via ATM(automated teller machine) and POS(point of sales). Throughout the time between Q3/2022 and Q3/2023, the total transaction value remains kind of unfluctuating. However, the proportion of POS is gradually going up , in exchange for the downward trend of ATM's transaction value. This shows the undeniable shift in social recognition and preference.

Chapter 2

Business Analysis

2.1 The current state of other applications

The point of sale machine in a regular pharmacy like "An Khang" or "Pharmacity" is mainly in charge of some main functions which will be listed below:

1. Inventory Management:

- Track the quantity of drugs in stock.
- Alert when it is time to reorder.
- Monitor expiration dates and batch information.

2. Sales and Payment:

- Record information about products sold
- Manage selling prices and discounts
- Process various payment methods such as cash, credit cards or electronic payments

3. Invoice and Receipt Printing:

- Print purchase invoices providing details transaction
- Automatically update data in the accounting system.

4. Customer Management:

- Record customer information, especially details related to purchased medications.
- Intergrate loyalty programs or special offers for regular customers.

5. Reporting and Analytics:

- Generate reports on sales, profits and inventory levels.
- Analyze data to support business decisions.

6. Discount and Promotion Management:

- Apply discounts and promotions according to specific rules.
- Monitor the effectiveness of discount and promotion strategies.

7. Security Management:

- Ensure the security of transaction information and customer data.
- Authenticate employees and monitor their activities on the system.

8. Integration with Other Systems:

- Connect with accounting systems or overall management systems of the store.

2.2 Project Objective

The primary objectives of the Point of Sale (POS) application encompass multifaceted functionalities, ensuring a seamless and efficient retail experience. Firstly, the application aims to perform as a standard machine, facilitating transaction processing, inventory management, and sales tracking. The key goal is to build a functional cloud-based product. To accommodate diverse retail environments, scalability is a minor goal, enabling the application to seamlessly integrate with multiple POS systems across different stores, thus fostering centralized management and reporting. Further, the POS system is intricately implemented and integrated with a banking gateway, mitigating concerns related to fake transaction proof. This feature ensures

a reliable and fraud-resistant payment process, relieving cashiers of the burden associated with identifying fake transfers. Additionally, the application incorporates cutting-edge technology by featuring face recognition capabilities. This innovative feature automatically identifies and associates customers while maintaining data privacy and security.

2.3 Project Scope

The project scope defines what needs to be achieved, the resources required, and the constraints involved. In the context of a Point of Sale (POS) application, the project scope would delineate the features and functionalities to be included, such as transaction processing, scalability across multiple stores, integration with a secure banking gateway, and implementation of face recognition technology. Additionally, the scope would specify any limitations or exclusions, providing a clear roadmap for project execution.

2.4 Tasks Assign

To better boost the overall quality, thus to make the best products in given time, we divided the project objectives into parts and assign to members. By making sure each member has the chance to express their capabilities and challenge new tasks and fields, while obeying strictly to the project scope, we have been successfully enhanced each individual's knowledge both in depth and width.

- **Table of Assign**

- **BI12-040(Coder):** Front-end, Back-end.
- **BI12-167(Leader):** Front-end, Back-end, Report, Presentation.
- **BI12-171(Coder):** Front-end, Back-end.
- **BI12-172(Designer):** Front-end.
- **BI12-173(Coder):** Front-end, Back-end, Report, Presentation.
- **BI12-195(Back-end Developer):** Back-end.

Chapter 3

Methodology

3.1 Functional requirement

3.1.1 List of Actors

1. Cashier.
2. Admin.

In our POS system project there are two "main" actor which are Cashier and Admin. The Cashier mainly focus on serving the Customer, control the cart content receive cash payment or provide QR for bank transfer. Admin is basically a Cashier with more capability over the system. Admin can access each and every information of the POS system, while making changes here and there. However, there is a role for Customer who actively interact with the system through cashier.

3.1.2 Featuring functions

No	Functionality	Description
1	Create Account	This is one of the admin privileges that can create, or delete user while applying role(cashier or admin) to them.
2	Sign in	By logging in the POS system with created account from previous step, cashier can immediately access the POS screen and can directly serving customers. Any orders placed from that moment is recorded in the name of that cashier, until he/she log out. The admin, by the other hand can view the whole system inventory while changing them.
3	Lock POS screen	There will be time that the cashier can not continuously operate the POS system. This function allows cashiers to temporarily lock the POS screen, avoiding fraud from accessing the system. Password is required to unlock the POS screen again.
4	Setup bank account	This is one of the Admin's privileges where bank account information is used for generating QR code, bank transfer transaction status is marked "Ok" when these accounts receive exact amount of value with correct message.
5	View Bank transaction	This is one of the Admin's privileges .Every bank transaction information is saved within a dataset for future review.
6	Manage coupon	Coupons are used to reduced the total cost of cart based on coupon value. Different users are granted difference coupon based on their account (loyal customer will be provided with larger value and number of Coupon).
7	Manage Customer	This function saves all the information provided by Customers like name, address, phonenumber and customfield like bloodtype, allergy..... Cashier can access View-o
8	Manage Product	Actors can view Products information within the Inventory. These information included quantity, price, input date

9	Manage Order	Actors can access Order details to retrieve information of previous orders. This empowers Actors with the ability to acknowledge the Order information.
10	Control POS screen	One of the most important group of functions in the whole System. This is the part where Cashier add, remove, change quantity or change cart number. Cart subtotal with tax is also provided after each change made to the Cart for both Cashier and Customer for view.
11	Check Out	Cashier will ask for the customer login method(facial or phone number). In this step customer can choose to login or pay as walk-in customer, and also with the payment method(cash or bank transfer). Employee will generate QR code based on the content of the current cart and coupon value. The items sold will be deduced from the inventory if the transaction status is "Ok"
12	Print Receipt	Either when the cashier receives the correct amount of cash or when the bank transfer with correct value reached the destination bank account, the transaction status is "Ok". After that the Customer screen will display an thank you message with the receipt.

Table 3.1: Featuring function table

3.1.3 Functionality relationship matrix

No	Functionality	Priority	Cashier	Admin
1	Create Account	High		X
2	Sign in	High	X	X
3	Lock POS screen	Low		X
4	Setup Bank account	High		X
5	View Bank transaction	Low		X
6	Manage coupon	Normal	X	X
7	Manage Customer	Normal	X	X
8	Manage Product	High	X	X
9	View Order	Normal	X	X
10	Control POS screen	High	X	X
11	Check Out	High	X	X
12	Print Receipt	Normal	X	X

Table 3.2: Functionality relationship table

3.1.4 Facial recognition

No	Functionality	Description
1	Register	In the process of registering a new customer, a facial descriptor will be computed and subsequently stored in a distinct collection, utilizing the user's ID as the labeling identifier.
2	Searching & Matching	Employing a MongoDB query in conjunction with the FaceMatcher functionality from face-api.js, the computed facial descriptor obtained from the client's point-of-sale (POS) system will be utilized to identify the user ID associated with the matched facial data label
3	Updating	In the event of updating face data, the identical process of searching and matching, culminating in the final step, will lead to the overwrite of the existing facial data with the corresponding matched label (userID).

3.2 Use-case

3.2.1 Use-case diagram

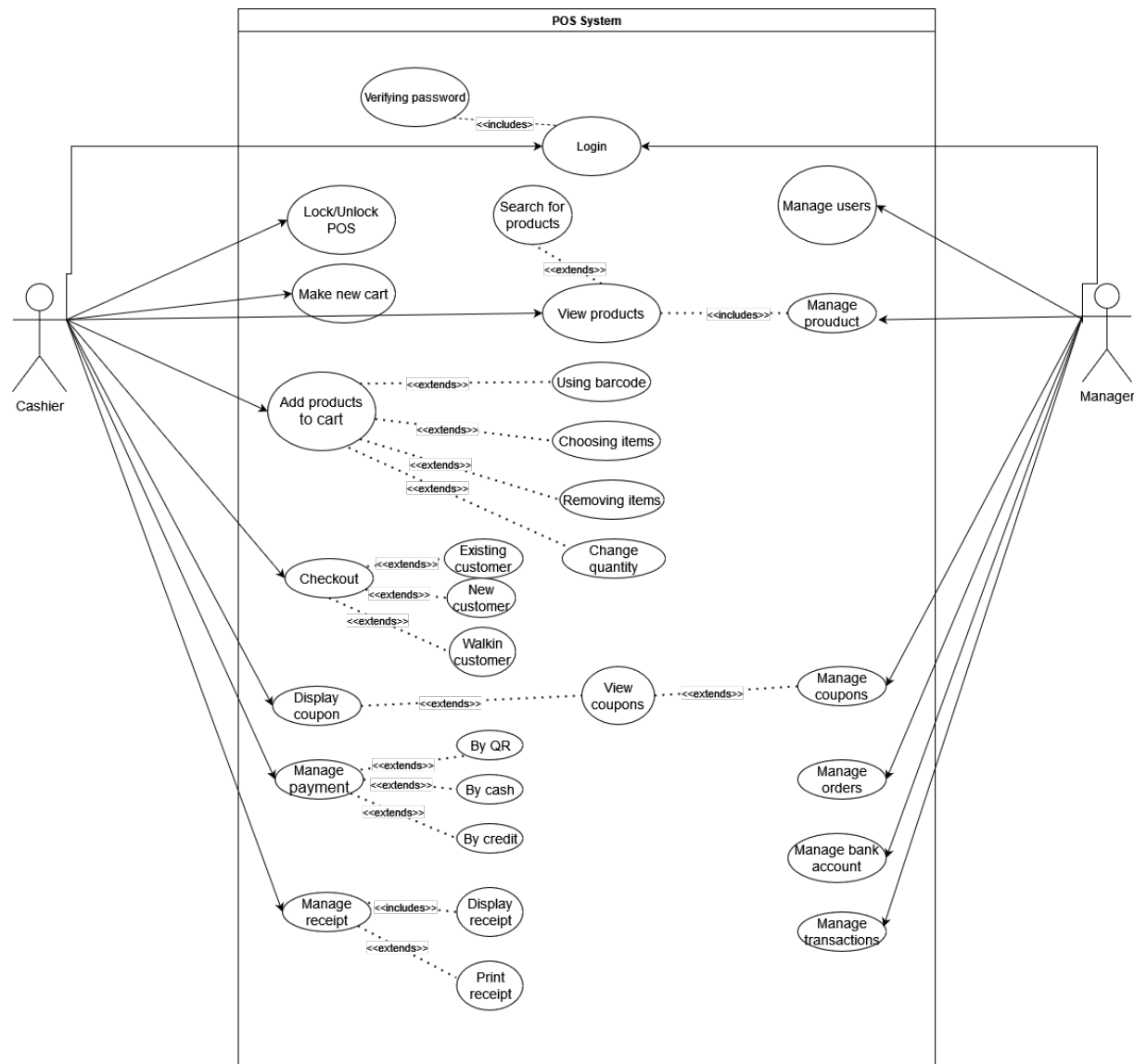


Figure 3.1: General use-case diagram

3.2.2 Use-case description

1. Create account

Actor action	System action	Input data
1. Admin login to system with admin account.	2. System requires account information	- email - password
3. Admin navigate to Management screen and click on User tab.	4. System shows current User list and "New User" button.	
5. Admin clicks on "New User" button	6. System shows create user form and requests Admin for information.	- name* - email* - password* - national ID* - role* - employee ID
	7. System validate data and create new User with input data. The account then can be used to log in.	

Table 3.3: Create account basic flow

- Alternative flows:
 - **Invalid Login credential:** During the login phase, if the Actor has not provided the correct login information, the system come up with two options: forgot-password or retype the login information.
 - **Invalid format or Insufficient Information:** If the actor fails to provide all the required information, the system will make actor to either fill the missing value or conclude the Use Case.
- Special requirement:
 - Only Admin can add new account into the system.
- Pre-conditions:
 - Admin must be logged into the system before.
- Post-conditions:

- New user is created if use case is successful, else the state remains unchange.

2. Sign in

Actor action	System action	Input data
1. User access any of POS system functions.	2. System shows the login page.	
3. User input login information.	4. System validates the input data and log in the system.	- email* - password*

Table 3.4: Sign-in Basic flow

- Alternative flows:
 - **Invalid email/password:** During the step 3, if User failed to provide correct login information (email or password), the system will make User restart the Sign-in flow or conclude the Use Case.
- Special requirement:
 - None.
- Pre-conditions:
 - Existing user account.
- Post-conditions:
 - If the Use case is successful, the User will be granted access to the system according to the account role.

3. Lock POS screen

Actor action	System action	Input data
1. User navigate to Settings in Management screen.	2. System shows the Settings page.	
3. User click on "New Settings" button and type in new password.	4. System validate the input and record new password.	- password*

5. User click on button "Loc-k" in the POS screen.	6. The screen is now locked and require password to unlock.	- password*
7. User type in password	8. System validate password and re-open the POS screen	

Table 3.5: Lock POS screen basic flow

- Alternative flows:
 - **Invalid format or Insufficient Information:** If the actor(admin) failed to provide the valid password, the system will require he/she to choose another password or conclude the UseCase.
 - **Invalid password:** If User failed to fill in the correct password, the POS screen remains locked and ask for a valid one.
- Special requirement:
 - Only Admin can add new password.
 - Both Admin and Cashier can lock and unlock the POS screen.
- Pre-conditions:
 - The Actor must have logined to the system.
- Post-conditions:
 - If the UseCase is successful, new password is added and can be used to unlock POS screen.

4. Management screen

- The flow of execution table below represent a relatively similar flow from No.4 to No.9 in the featuring functions table.

Actor action	System action	Input data
1. Actor navigate to the management page and clicks on the wanted category.	2. System shows the chosen page with current information from the dataset.	
3. Actors can either view or manage information based on their account role.		
4. Actor clicks on creates new data object.	5. System requires Actor to fill the information	- data*
	6. System validates the information and create new object.	

Table 3.6: Manage Information basic flow

- Alternative flows:
 - **Invalid Format or Insufficient Information:** During step5, if Actor failed to provide correct information, the System will require Actor to either fill up the missing value or conclude the UseCase.
- Special requirement:
 - Cashier has limited authority over some category while Admin can freely access to all existing resources.
- Pre-conditions:
 - Actor must have logged in.
- Post-conditions:
 - If the UseCase is successful, new Object will be added to the dataset and can be further used in the POS system.

5. Control POS screen

Actor action	System action	Input data
1. Actor navigate to the POS screen	2. System fetches the Products information and displays in the POS screen	
3. Actor(role: Cashier) can either add, remove and change quantity of products in cart.	4. System calculates and displays the current cart value with tax value.	

Table 3.7: Control POS screen basic flow

- Alternative flows:
 - None.
- Special requirements:
 - The product inventory must not be empty.
- Pre-conditions:
 - The Actors must have logged in the system.
- Post-conditions:
 - If the UseCase is successful, the cart value and cart content will be ready to be used in the next process - Check out.

6. Check out and Print receipt

Actor action	System action	Input data
1. Actor clicks on the "Payments" button.	2. System pop up a box with options for payment method and customerType	- payment method* - customer type*

	3. If client wants to register, system shows the registration form and requires input.	<ul style="list-style-type: none"> - name* - phoneNumber - email - face - address - credit - customField
	4. System scan the customer face to recognize client, fetches information and print on the screen.	
	5. If client wants to pay by bank transfer, system will generate QR code on the customer screen.	
6. If the user pay by cash, Actor checks the cash amount and click "Submit",	6. System validates the bank transfer message and value to mark the transaction status as "OK"	
	7. Items sold will be deduced from the inventory and receipt is printed on the customer screen. Order then is recorded.	

Table 3.8: Check out and Print receipt basic flow

- Alternative flows:
 - **Invalid Cart value:** If the cart is currently empty, checkout cannot be done.
 - **Invalid Format or Insufficient Information:** If during step3, valid information is failed to be provided, the system will request Actor to fill in the missing value.
- Special requirement:

- The Cart should be ready for check-out from the last process.
- Pre-condition:
 - None.
- Post-condition:
 - If the UseCase is successful, new Order row of data is created in the Order dataset, bank transaction is recorded and receipt is printed on the customer screen.

7. Create Customer

Actor action	System action	Input data
1. Cashier is entering the checkout process	2. System show a list of existing customers, and a create new customer button.	
3. Cashier clicks on Create New customer button.	4. Camera shows up and takes photos of Customer and requires other details filled.	- Face - name* -phone - Id number
	5. New customer is created and the system moves on with the checkout process.	

Table 3.9: Create customer basic flow

- Alternative flows:
 - **Invalid image value:** If the system failed to take picture, or customer just dont want to give out personal data, phone number can be used to create new customer and log in.
- Special requirements:
 - None.
- Pre-conditions:

- The Actors must have logged in the system.
- Post-conditions:
 - If the UseCase is successful, the new Customer is created and will be ready to be used in the next process - Check out.

3.3 Diagrams

3.3.1 Sequence Diagrams

General Sequence diagram

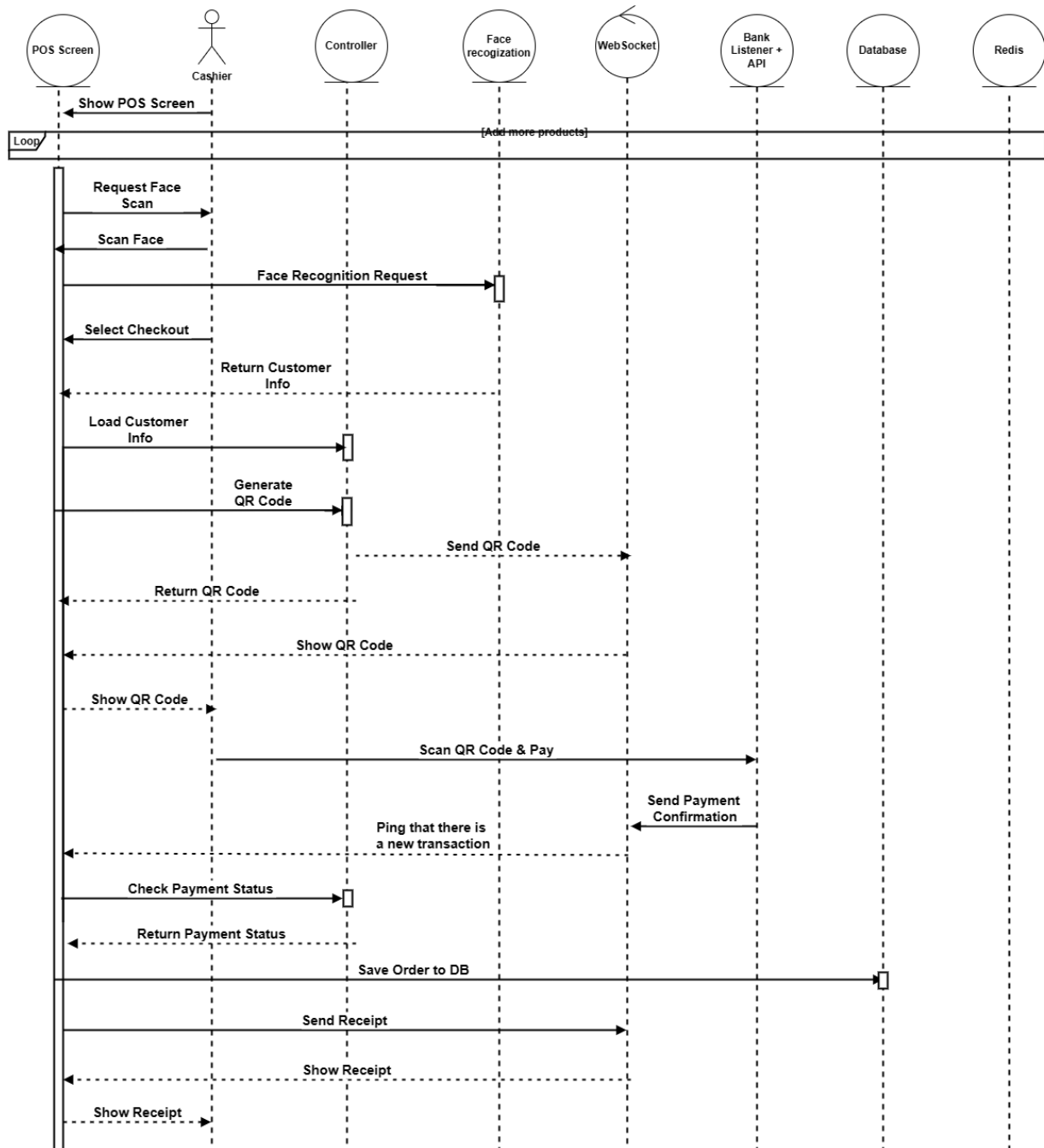


Figure 3.2: General Sequence Diagram

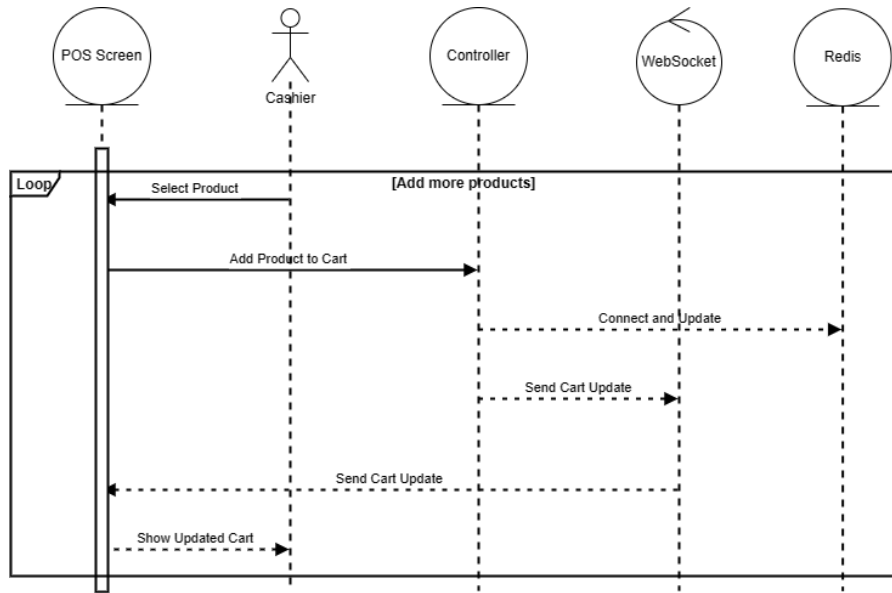


Figure 3.3: Cart Loop Diagram

Although this is only a simplified version, let us break down this complicated sequence diagram into text for better understanding, note that Controller in the diagram represents 3 different Controller (POS, Cart, Transaction):

1. The POSController is in charge of retrieving product / customer data and bring it onto the POS screen, where the Actor Cashier mainly interacts with the system.
2. Hereby products are added to cart, removed or changed in quantity repeatedly thanks to the CartController. Everytime changes happen to the cart content, the controller:
 - Connect to **Redis** to make changes.
 - Send changes to customer screen via **Websocket**.
 - New cart and cart value are then returned onto POS screen.
3. Let's just put in consideration that client chooses to pay via bank transfer:
 - The face scanner on customer screen will capture picture of client.
 - POSController sends the request to the **face recognition server** and get customer's information in return.
 - POS screen loads all the data from controller.

4. Now the system has already known which client is using the service, it sends a QR request to the **TransactionController** and receives a QR in return. The code is passed to the customer screen via Websocket server.
5. The **PaymentListener** will automatically call the **WebSocket** to send new event(payment received) to POS screen, which will call the **TransactionController** to validate the latest information from the bank with the specified QR data.
6. The notification of transaction completion is sent to the **WebSocket** to display onto both the customer screen and POS screen.
7. Cart content included value and products inside is saved to the **MongoDB Order table**, thanks to the CartController.

Customer CRUD diagram

This diagram illustrates the process of adding, removing, or editing customer details within the system. The manager has the option to initiate the addition of a new customer to the system. Upon successful completion of this process, the system generates a confirmation message. Furthermore, the manager can conveniently access the main view to review a comprehensive list of customers. In this main view, the manager has the capability to seamlessly edit or remove customers from the system as needed.

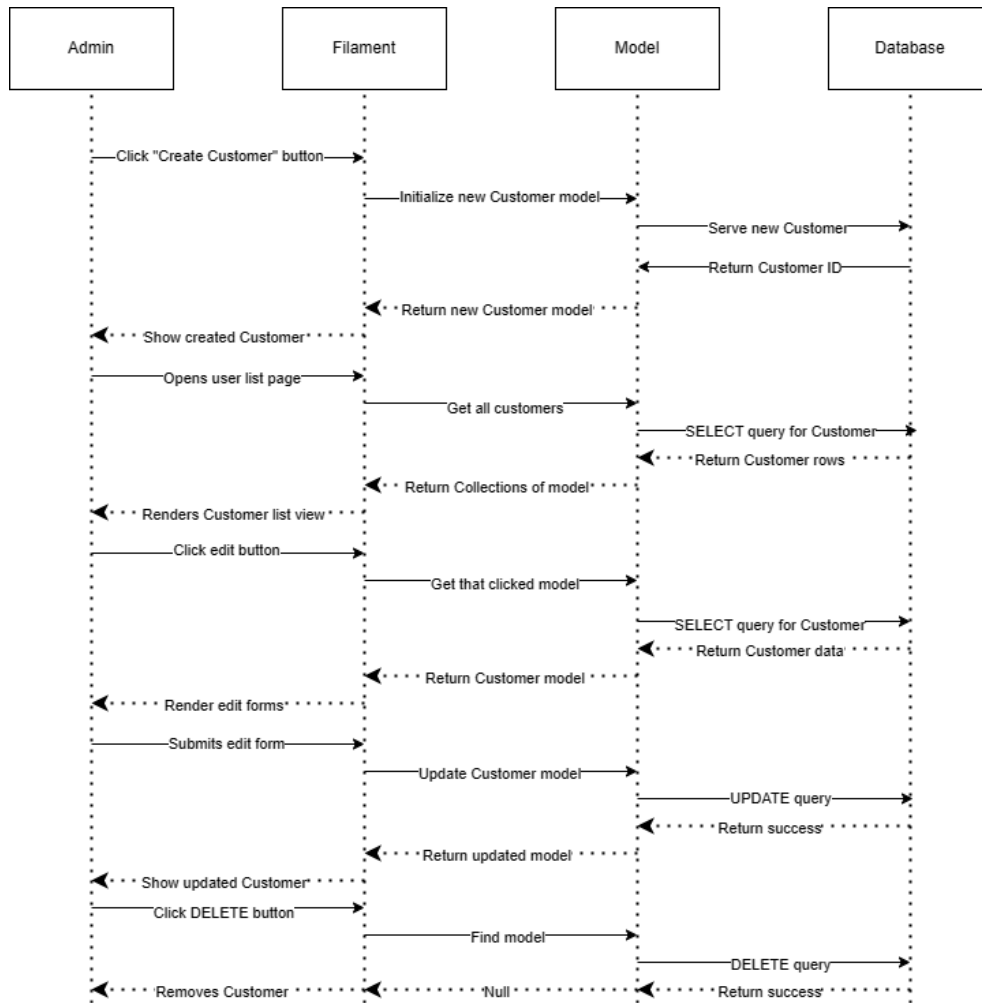


Figure 3.4: Customer CRUD diagram

3.3.2 Database diagrams

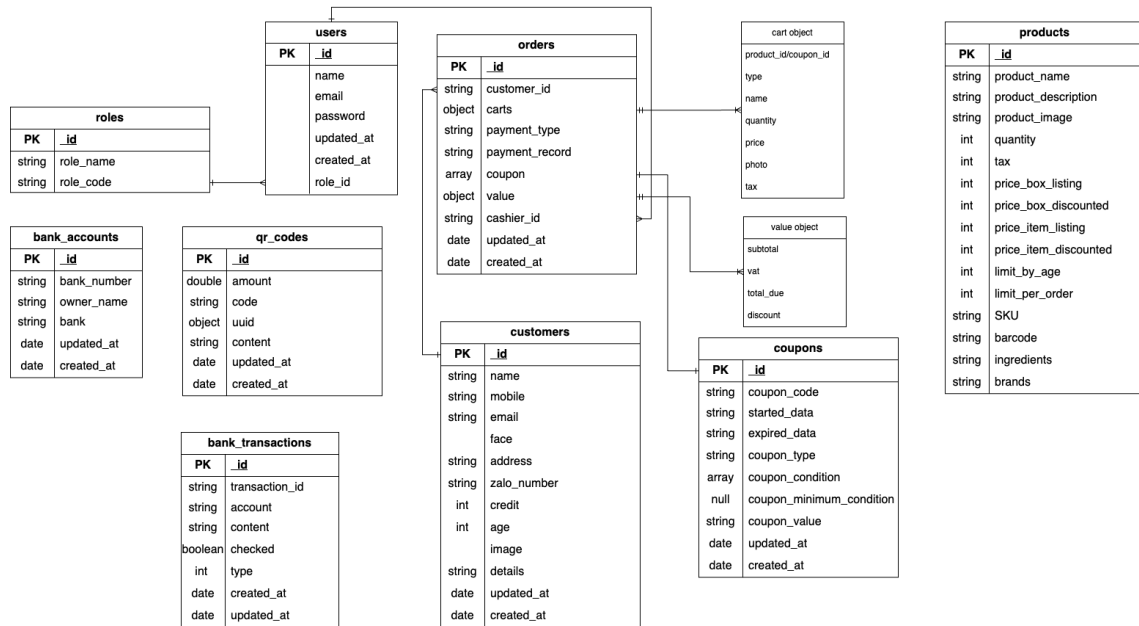


Figure 3.5: The design of the System Database

Database break down

The database consists of 9 main tables each with different size and usage, and 2 objects on the side:

1. **Bank-account:** Consists of bank transfer accounts information(bank-number, owner-name and bank).
2. **Bank-transactions:** The contents of every payment made by customer via bank transfer.
3. **Coupons:** This dataset reserves coupons information including expired-date, coupon-type and coupon-value with requirements and many more.
4. **Customers:** Every customer who signed in has their information saved here. This include name, email, phone, age...etc....
5. **Orders:** Cart content is saved in Object format, with customer and cashier-id, time and date of the creation of order.
6. **Products:** Products' name, image, prices other necessary information is recorded here.

7. **QR-codes:** Just QR code.

8. **Roles:** Account role - which grants specific accounts with appropriate privileges.

9. **Users:** Including name, email, password and role-id of every Actor account.

3.4 System Architecture

3.4.1 Design

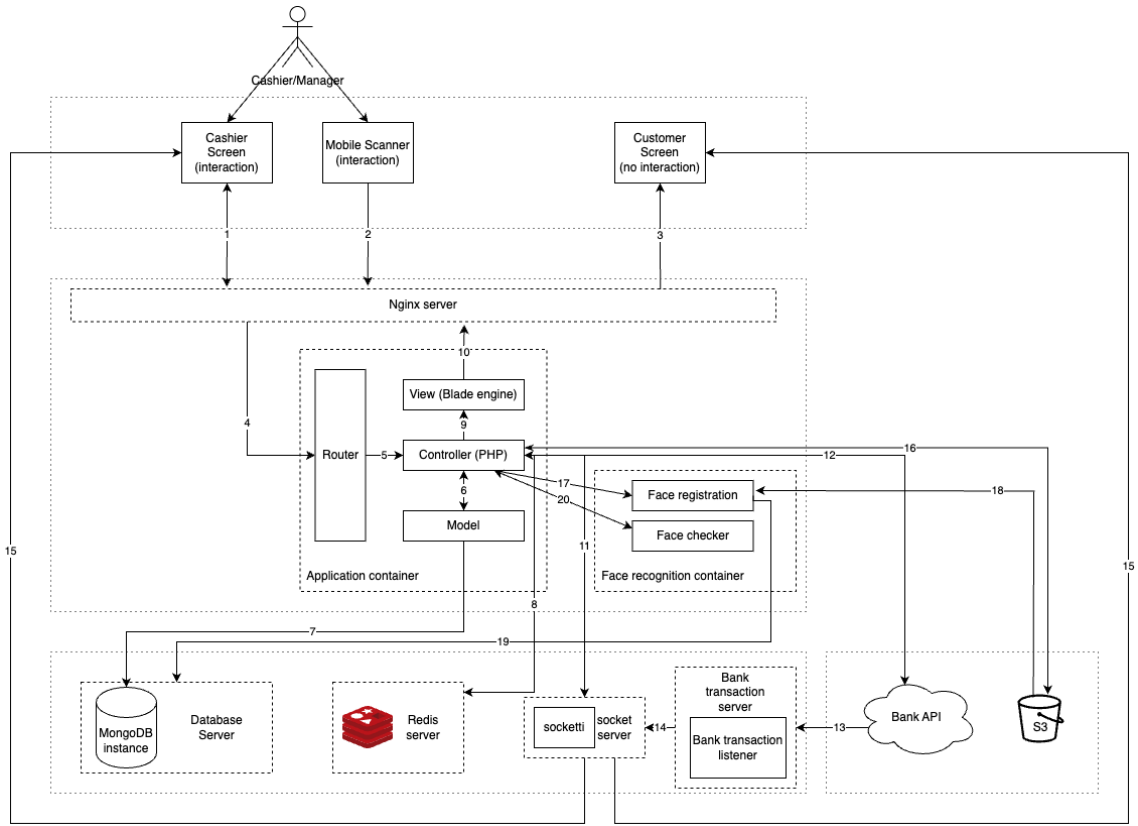


Figure 3.6: The design of the System architecture

The system architecture has been designed with 2 main zone and several listeners and workers that do other task that cannot be integrated within the main system.

3.4.2 The connection and data

1. Send and retrieve the data of POS screen to the application
2. Send the request containing specific bar code to the application

3. Retrieve the data from application
4. The HTTP request from NGINX server will be transferred to the PHP router or the
5. The router will read and perform the routing corresponding controller
6. If the method of the controller require the read and write operation to the persistent database, the controller will work with the data based on the model
7. When there is confirmed about the data, the model will perform the query via the Eloquent to the Database
8. If the method of the controller involve in the task of reading or writing to temporary cart data, the controller will directly read or write to Redis via cache interface
9. If the method of the controller involve in the task of providing user interface or responding the API data, it will send to the view engine.
10. After the view engine generate the content of the view, it will send the response data to the NGINX server and the NGINX server will serve it as HTTP data
11. If the method of the controller need to send the information directly to the POS screen or customer screen, it will send directly to selected channel
12. If the method of the controller need to query for the transaction from the bank, it will call to the Bank API via HTTP with provided bank token
13. When the bank receive a new notification, it will send a message via Firebase to registered device
14. The registered device will listen to the change of the notification logs and send messages to all registered socket channel in socket server.
15. The socket server send the messages to selected channel for the screens whenever the request is sent (via controller or via other servers)

16. If the method of the controller involve in the task of read or write static contents to the S3, it will directly connect to S3 compatible storage via the AWS S3 package
17. After the user is created, it will send the newly created user id and link of the image to face server
18. The face registration load the S3 image based on the link
19. After performing face features extraction and descriptors generation from image it will be saved to the MongoDB with label and objects
20. When the face is captured and send to the face checker, it will send to the face checker and the face checker will return to the user details

3.4.3 Client

The client zone is demonstrated by one single cashier and customer. This is consisted by 3 different devices:

- **POS Screen:** This is the device that the casihier do the interactions, including but not limited to: Manager Cart, Manager or Read Customer information, Perform Checkout for customer.
- **Scanner Mobile:** This is the device that connected to the POS server and it will perform the scanning of BarCode of each items.
- **Customer Screen:** This is the device that the customer will read what they have bought, the subtotal of the cart and the QR code for banking transactions.

3.4.4 Application

- **Nginx Server:** The nginx http server prodvides an protocol for the deviecs to communicate with the Application and Data.
- **App Module:** The app module is specified in to 4 different blocks
 1. Router system: Routing the client to specific controllers

2. Controllers systems: This will handle the logic of the Point of Sale system, the management of different data sections
3. Controllers systems: This will handle the logic of the Point of Sale system, the management of different data sections
4. Views: Render the view and screens for the users of the app.

3.4.5 Other workers and third-party

MongoDB instance

The instance is deployed to store the persistent data

Socket server

The socket server act as the server that handle realtime messages for the web application, including but not limited to: sending transaction notification, sending ping for new product add to cart, sending cart information,...

Redis server

Redis is an open-source, in-memory data structure store that can be used as a database, cache, and message broker. It is known for its high performance and flexibility.

Bank transaction server

Since the VPBank provide an API for querying latest transactions but not providing the WebHook. That's why we have to setup a server that listens to the Firebase notification and then send websocket message (which doesn't contain information) ping to the screen for the server to query the transactions and check for needle provided.

3.5 Tools and Technical choices

3.5.1 Client-Server Model

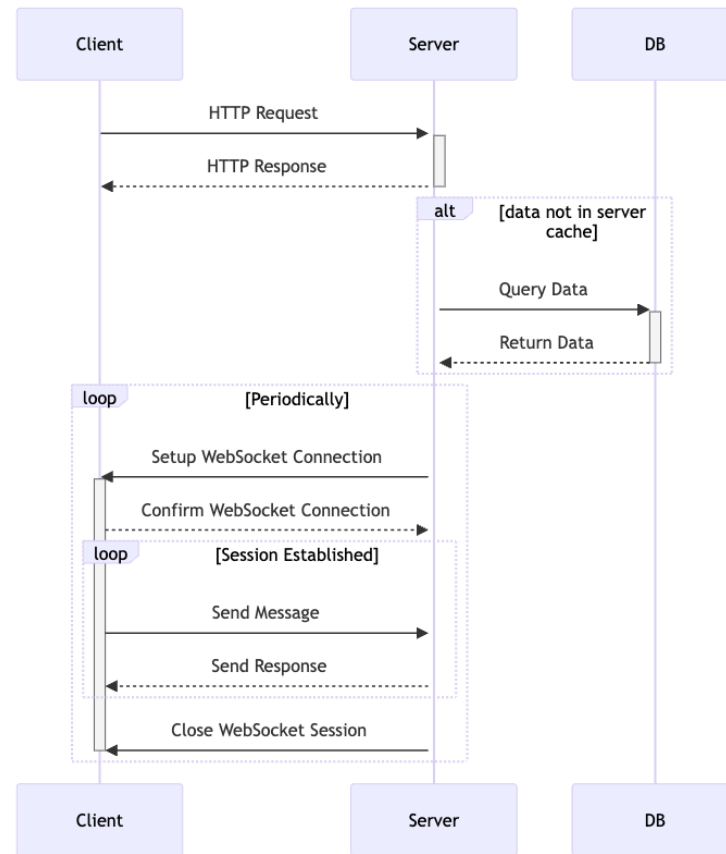


Figure 3.7: The client-server and websocket

This web application is based on the basic client-server, which is clients send an request to server and the server return a response to that specific response. The reason why we choose this model is that the application is based on the database, so the data can share between multiple machines and adapt with high availability and changing devices. Moreover, with the support of modern web, it can establish a socket that server can push back data for the client without waiting for the client performing request.

3.5.2 Github

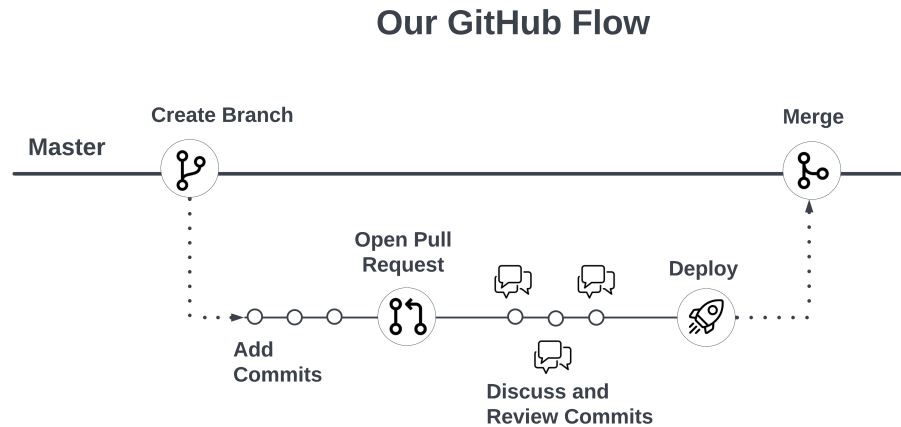


Figure 3.8: Github workflow

GitHub is a web-based platform and a version control system that facilitates collaborative software development. GitHub is based on Git, a distributed version control system. Git helps track changes in source code during software development, making it easier for multiple developers to collaborate without conflicts.

3.5.3 MongoDB NoSQL

When it comes to choosing database, the biggest decision is picking a database management system (DBMS). Among those widely used DBMS, we decided to choose NoSQL, an upcoming category of DBMS, with no common structured scheme for all records. Unlike conventional databases that adhere to a rigid structured scheme for all records, NoSQL databases offer unparalleled flexibility, allowing us to store and retrieve data without the constraints of a predefined schema. With the flexibility of the MongoDB, we can store the snapshot of carts, value in child documentation instead of having to create different tables for each child object.

3.5.4 Coding languages

PHP

PHP: Hypertext Preprocessor is the most common language for implementing an web application with the ease of deploying and simplicity.

Flutter and Dart

In the realm of mobile application development, our choice lies with Flutter¹ and Dart². Flutter provides a comprehensive framework that empowers us to initiate projects seamlessly. It encompasses key elements and the cross support for two major mobile operating systems: Android and iOS.

Laravel and Filament

We choose the Laravel³ Framework for the application since it provide more than enough for us to start a project including the authentication, sessions, MVC model, connection to the database and also the needed security measures for protecting the application. For basic management and CRUD, we use another extension called FilamentPHP⁴ for the framework to supercharge and speed up the development process. It provides the forms, table and components with the stacks of AlpineJS⁵, Tailwind-CSS⁶ and LiveWire⁷

HTML.

HTML, or HyperText Markup Language, is the standard markup language used to create and design documents on the World Wide Web. It is the cornerstone of web development and provides a structure for web content by using a system of tags and attributes.

CSS.

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation and visual formatting of a document written in HTML or XML. In web development, CSS is essential for controlling the layout, appearance, and design of web pages.

¹<https://flutter.dev/>

²<https://dart.dev/>

³<https://laravel.com/>

⁴<https://filamentphp.com/>

⁵<https://alpinejs.dev/>

⁶<https://tailwindcss.com/>

⁷<https://laravel-livewire.com/>

JavaScript.

JavaScript is a versatile and widely used programming language primarily known for its role in web development. It enables developers to add interactivity, manipulate the Document Object Model (DOM), and build dynamic content within web browsers and also the support of JQuery, which handle tasks like DOM manipulation, events, animations, and Ajax.

3.5.5 REST API technology

In our POS system, several endpoints that is called by Ajax needs an API with the system.

REST (Representational State Transfer)

REST is an architectural style for designing networked applications. REST allows communication between software written in multiple platforms can communicate and exchange data easier by a common datatype called JSON(JavaScript Object Notation). The REST common uses 4 method of HTTP to express request purpose:

- GET: retrieve data from server
- POST: submit data to specific resource.
- DELETE: remove resource
- PUT: update resource

API (Application Programming Interface)

API facilitate communication and data exchange between different software systems. If the user sends request following some specific formats then the server will process that request as a specific command for running some special services provided by server-side and responding the output results of that service back to user.

3.5.6 S3

S3 stands for Simple Storage Service, which is a product initially provided by Amazon to provide the storage for objects like pictures, audio, movies, static contents, etc for

cheap price. The reason why we decided to go with S3 is because it can make the deployment and movement easier when we do not have to worry about moving images from one to another server.

3.5.7 Redis

Redis provides a in-memory database which can be accessed really fast and can be accessed directly via Unix-socket or TCP socket, this will speed up the storing process and provide an ability to scale up the application when there is an array of application nodes.

3.5.8 Socketti

The ability of web real-time communication is really important in this POS system because we wanted to push and retrieve data in real-time without. When we search to a pusher compatible self-hosted app, we see this server that can create a websocket server with high concurrency user.

3.5.9 face-api.js

Vincent Muhler's face-api.js is a JavaScript API developed for face detection and recognition in both browsers and Node.js, built on the foundation of tensorflow.js. Specifically designed for mobile devices and browsers, this API ensures that face recognition tasks are accomplished in just a matter of seconds.

Chapter 4

Implementation

4.1 Back-end

The back-end of the application will handle these following aspects:

- Provide managers and cashiers interface for manage customers, inventory, coupon, etc.
- Provide the cashier interfaces to perform cart, checkout for customers.
- Provide the customer an interface to see what they are buying in realtime.
- Provide the face recognition back-end to handle computing tasks.

4.1.1 Model

It is dangerous to let cashier directly access to the database, so Laravel already has the protocol called `ConnectionFactory`¹ for the application can securely access to the Database. In order to use the Connection, we need to implements Models and relations of the Models. We defined the Model based on the database design and relationships. In final, we have made total of 9 models: `BankTransaction`, `Coupon`, `Customer`, `Order`, `Product`, `QrCode`, `Role`, `Supplier`, `User`

¹<https://laravel.com/api/10.x/Illuminate/Database/Connectors/ConnectionFactory.html>

4.1.2 Resources

Usually, the process of managing the data in the database contains data posting, data querying, data validation, etc. For easily handling the process of managing the data in the database, we decided to use Filament supported us to provide the components for such things. For each model, we added a resources to manage the. The resources will contains the defined datatype of attributes to be stored, which attributes to be displayed and also the data validation for the input data. It will also handle the process of Creating, Reading, Updating and Deleting the data of each model.

4.1.3 Controller

With the POS machine, the flow and the interactions are ways more complicated than just normal CRUD, so we have to create Controllers to handle the Logic. This application contains these following controllers:

AuthController

The **AuthController** provides 2 methods: Login and Logout, which are `login` and `logout` that called the `AuthProvider` from Laravel Authentication.

CartController

The **CartController** provides the API for the POS machine and the customer screen. The functions are implemented to perform the following load cart, calculate the cart, load coupons, apply coupons, modify the current cart. The cart is stored in Redis Cache to provide the most performance and scalability in the future. It also call the events when the cart changes, so that the pos and the screen can acknowledge when the cart changed.

PosController

The **PosController** is quite simple, it just used to load the view of the POS and customer screen, also provide and API for the cashier can get the details of the customer.

TransactionController

This controller used to synchronize the list of transactions when there is request, and perform the search with the method finding needle in a haystack to extract the transaction message and compare it with the generated message to match them together. Moreover, it also handle the task of generating QR code and push the QR code to screens, so that client can scan them to make bank transfer without typing the details. The checksum is generated provided on Cyclic Redundancy Check on VietQR documentation

Function: generateChecksum(*text*)

Variables:

$crc = 0xFFFF$ (initial value)

$polynomial = 0x1021$ (0001 0000 0010 0001, 0, 5, 12)

$bytes = str_split(text)$

Operations:

For each byte b in bytes :

$b = ord(b)$ (Get ASCII value of the character)

For $i = 0$ to 7 :

$bit = \left(\frac{b \gg (7 - i)}{1} \right) \& 1$ (Get the i -th bit of b)

$c15 = \left(\frac{crc \gg 15}{1} \right) \& 1$ (Get the 15th bit of crc)

$crc \ll= 1$ (Left shift crc by 1)

If $(c15 \oplus bit)$:

$crc \oplus = polynomial$

Return $dechex(crc \& 0xFFFF)$

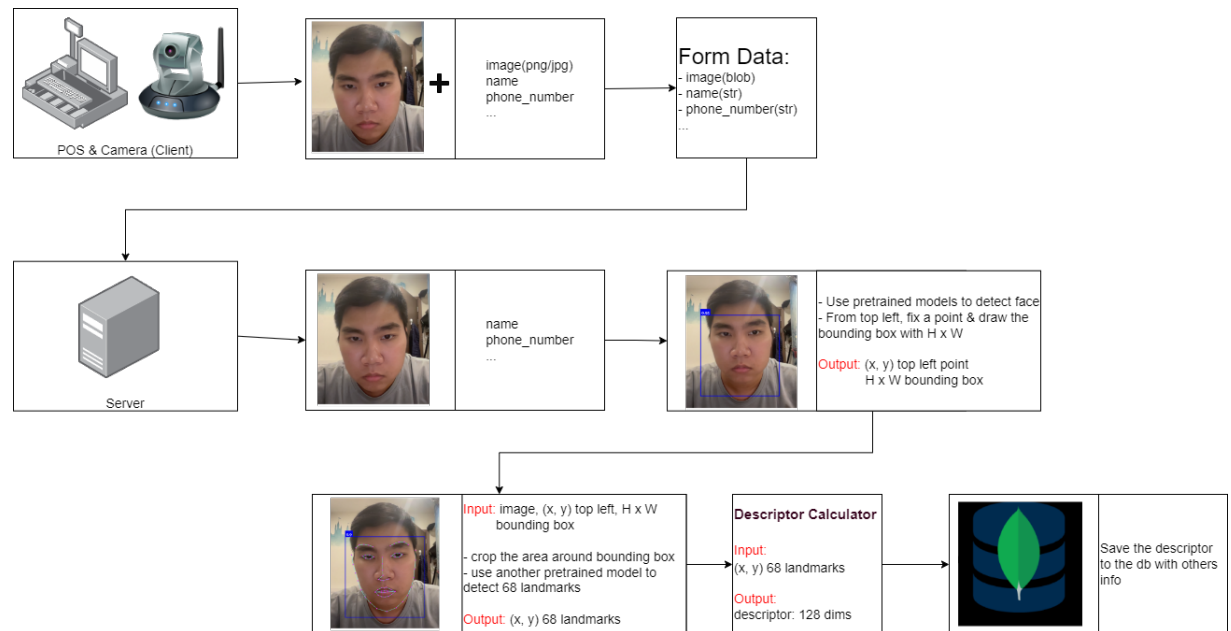
VPBankController

This is the bank controller which provide the ability to query transaction notifications from the bank with provided token and also normalize data to store them into database.

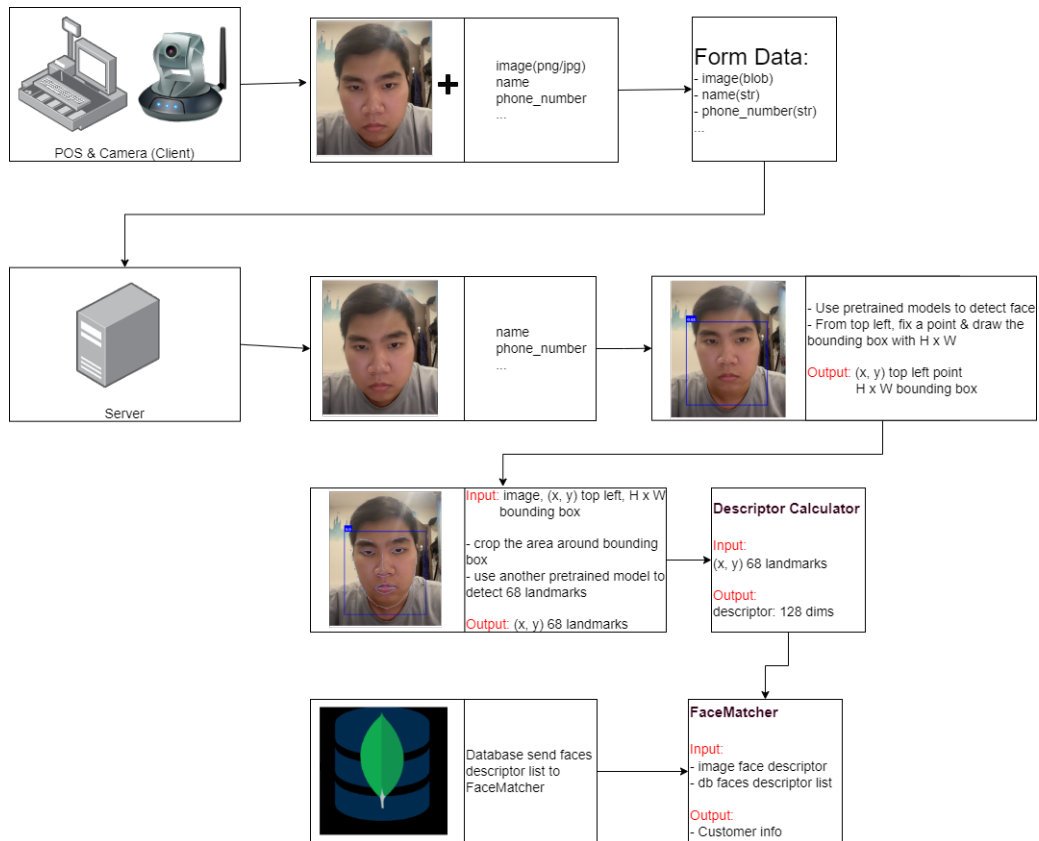
4.1.4 Face Recognition back-end

Handling the computation and feature extraction from customer images on the client side is not advisable. Therefore, all tasks related to face recognition will be conducted on the server side, utilizing images captured by the client side.

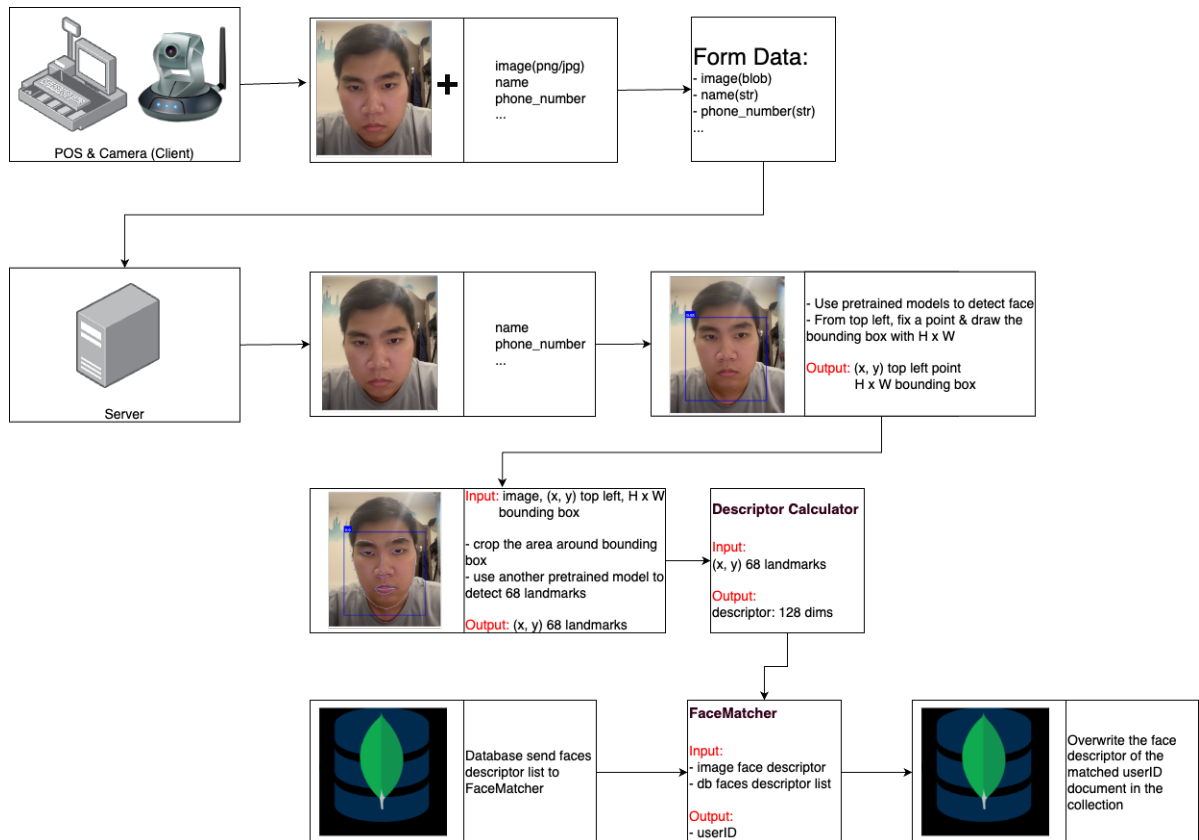
- Register new face data:



- Searching & Matching face data:



• Updating face data:



4.1.5 Handy POS Scanning application

The screenshot shows the 'Cielo POS Barcode Scanner' app interface on a mobile device. The status bar at the top displays the time 21:01, signal strength, and battery level at 30%. The app's title bar is blue with the text 'Cielo POS Barcode Scanner'. Below the title bar, there are three input fields: 'URL' containing 'https://apppos.vslim.io', 'ScreenID' containing '0', and 'CartID' containing '0'. At the bottom of the form, there are two blue buttons: 'Scanner' and 'Check'.

21:01

Cielo POS Barcode Scanner

URL
https://apppos.vslim.io

ScreenID
0

CartID
0

Scanner

Check

Figure 4.1: Cielo scanner After-scan

Barcode Scanner Integration: The barcode scanning functionality was implemented using Flutter packages like `mobile_scanner` to capture barcode data. The application was configured to make HTTP requests using the `http` package in Dart. The requests include barcode data and target cart/screen information which is preconfigured to be scalability.

4.1.6 Freshly generated QR

There are multiple cases reported being fraud or scam, using exactly the QR code each shop provided. Scammers take advantages of shop owner ignorance, to overpass their own QR code ontop of shop's QR code. For that reason, we implement models to freshly generate QR code and display it on-screen, avoiding any type of fraud and mismatch in bank transactions details.

4.1.7 Events

We implemented events to send real-time notifications to the screen based on the Pusher compatibility server socketti, so that the screen of POS and Customer can receive the update in real-time. There are two different event channels

- PushPOSDData
- PushScreenData

4.2 Front-end

4.2.1 POS machine screen

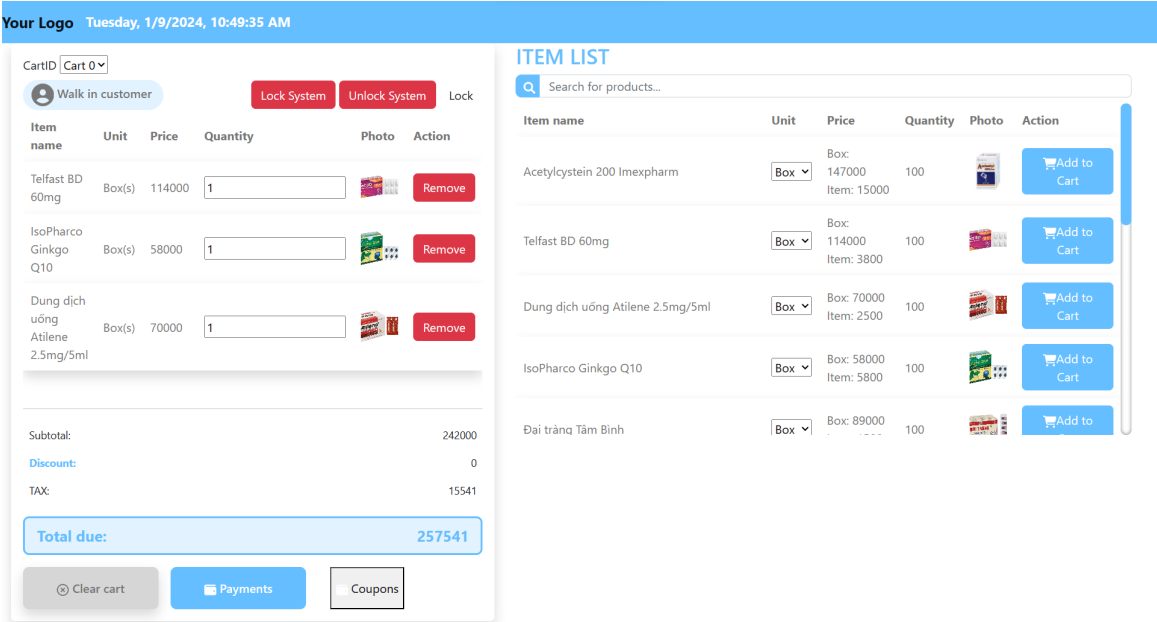


Figure 4.2: POS machine main screen

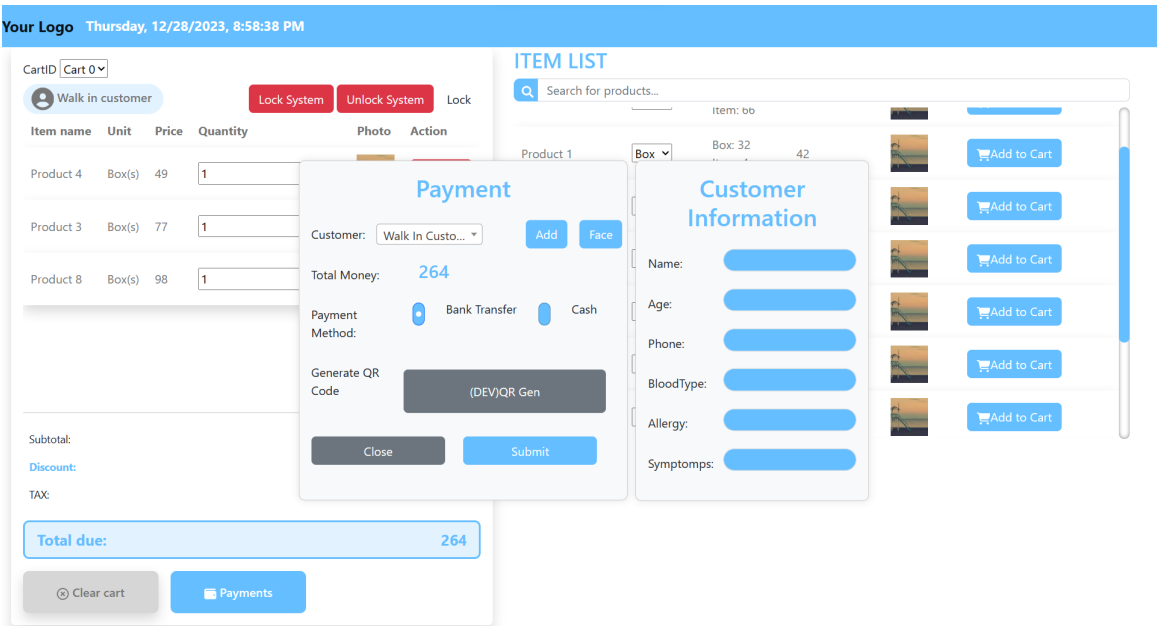


Figure 4.3: POS machine checkout screen

Overall, this front-end page represents a POS system interface with features for managing shopping carts, adding products, handling customer information, and processing payments. Our project relies on popular front-end libraries and frameworks for styling and enhanced functionality.

- **Structure:** The screen contains a navigation bar, two main container sections ("screen-left" and "screen-right"), and additional div elements for modals and overlays.

- **Left Screen Section ("screen-left"):**

- * This section represents the left side of a POS (Point of Sale) system.
- * It includes a drop-down for selecting a cart, a datetime indicator, and various UI elements for managing the shopping cart.
- * The shopping cart is divided into top, middle, and bottom sections.
- * The middle section contains a table for displaying items in the cart, and the bottom section shows the subtotal, discount, and tax information.
- * Buttons for locking/unlocking the system, clearing the cart, and initiating payments are provided.

- **Right Screen Section ("screen-right"):**

- * This section displays a list of products available for purchase.
- * It includes a search bar, a table showing the available products, and buttons for adding products to the shopping cart.
- * The right section also contains forms for customer information and payment details.

- **JavaScript Interactions:**

- The JavaScript code in the script tags handles dynamic interactions and functionality.
- It includes functions for adding items to the cart, searching for products, locking/unlocking the system, loading customer information, generating QR codes, and performing checkout/payment.

- Select2 is initialized for the customer type drop-down, and radio button change events are used to show/hide elements based on the selected payment method.
- **Styling:**
 - The styling is done using a combination of custom styles (defined in an external CSS file) and Bootstrap classes.
 - The design follows a clean and organized layout, with attention to responsive and user-friendly elements.

4.2.2 POS customer screen


Product	Unit	Price	Qty	This is some ads
Telfast BD 60mg	Box(s)	114000	1	
Dung dịch uống Atilene 2.5mg/5ml	Box(s)	70000	1	
<div></div>				
Subtotal:		184000		
Discount:		0		
Tax:		11480		
Total due:		195480		

Figure 4.4: POS customer screen

Customer screen implements code which facilitates real-time updates to a shopping cart screen in a POS system using Pusher for WebSocket communication. It dynamically updates the cart, total values, and displays messages and QR codes based on received events.

- **Structure:** The HTML includes two main sections: "cont" (for the shopping cart) and "thank-screen" (for displaying a thank-you message and QR code).

- The cart section (cont) contains a table displaying product information, and a div for showing subtotal, discount, tax, and total due.
- The "thank-screen" section includes an ad area, a thank-you message, and a div for displaying a QR code.

- **JavaScript Functionality:**

- jQuery is used to handle document loading and AJAX requests.
- The code establishes a connection to a Pusher channel using the provided Pusher key.
- The Pusher channel is subscribed to events related to cart updates and screen data changes.
- Functions are defined for updating the cart, displaying a QR code, and handling various screen data updates.
- The "update-total", "show-qr", "update-cart", "update-total-value", and "clear-screen" functions handle different aspects of updating the screen.

- **CSS Styles:**

- The CSS styles define the appearance of the cart and ad area, including border, padding, background color, and table styles.

- **Pusher Events:**

- The script listens for events related to new products added to the cart ('NewProductToCart') and general screen data updates ('PushScreenData').
- Depending on the received event, the script updates the cart, total values, shows a QR code, displays a thank-you message, or clears the screen.

4.2.3 POS manager screen:

This screen represents the inventory management tool of our Cielo POS system. We implement Laravel Filament3 to better visualize the inventory in divided category. Below is the current state of Dashboard. Here is where Admin witnesses the business model, the working efficiency of the POS system via charts and statistics number.

These provided the deeper knowledge and better view of how the POS system should be developed.



Figure 4.5: POS manager screen

- **Structure:** There are 3 main components onscreen, which are side-bar on the left, main-screen and the navigation bar on top.
 - Navigation Bar: Contains 2 buttons, language options and screen mode option (light-dark mode).
 - Side bar: Side-bar is divided into 3 categories: System, Business and Order management, each contain a specific role in the POS system.
 - Main screen: This is where the main contents of the management screen settled. This is often a table with saved data fetched from the MongoDB, with a create new button on the top left. This button links to the form of creating new object.

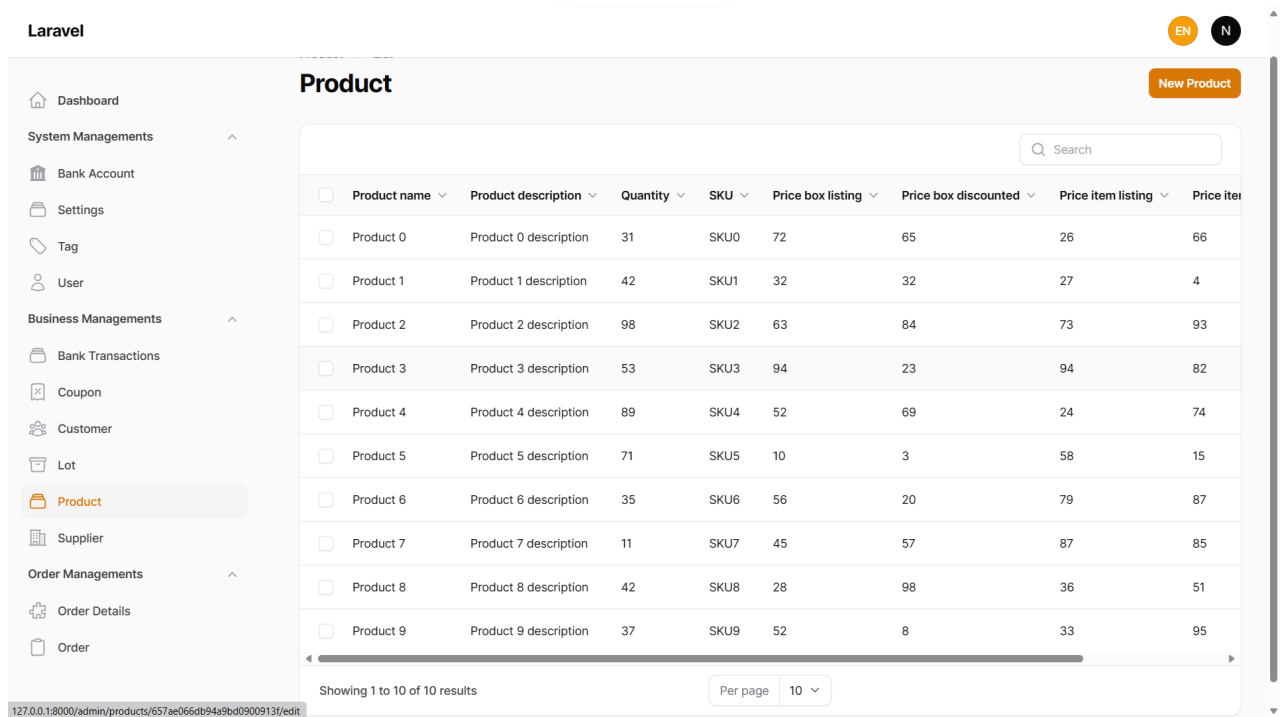


Figure 4.6: POS manager screen for Product

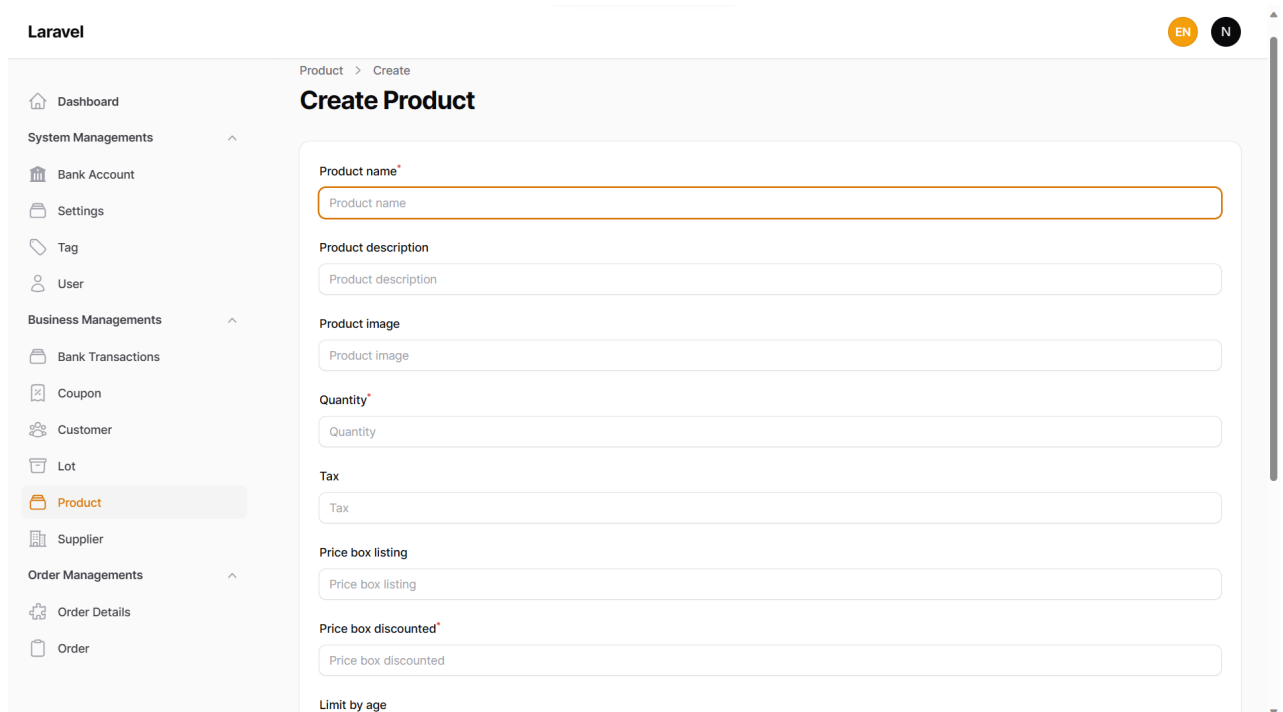


Figure 4.7: POS manager screen for create new Product

4.3 Side services

4.3.1 Banking Integration

The banking is one important part of our project, we have made the contact with MBBank and they do provide the API with fully functional webhook when an transaction is made. However the time for them to get ready and get started is so long, that we have to make a round turn.

Luckily, VPBank do provide a method², so that client can accesss and get the transactions history of a specific account. They also made a Firebase call into the registered device, but not a webhook. So, we have to made the following diagram to show how we made contact with the bank to get the transaction. The following sequence diagram indicates how we made the contact with bank.

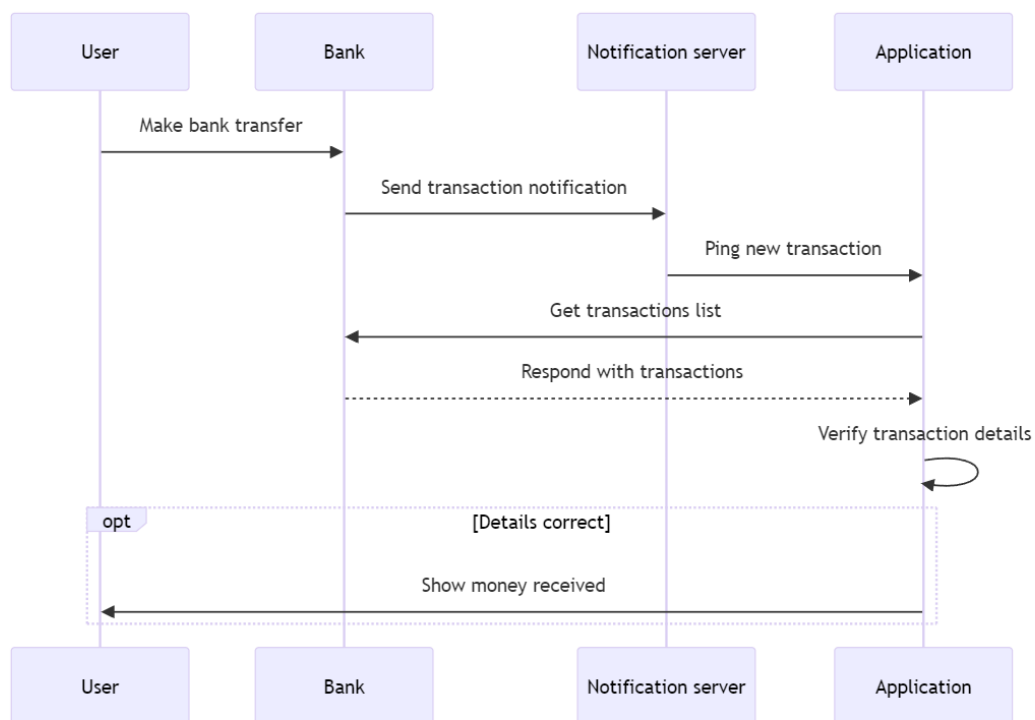


Figure 4.8: The sequence diagram of paying via bank

²<https://www.facebook.com/watch/?v=280789328218781>

4.4 Web Application

4.4.1 Managing the data

The process of managing the users, managing the products and seeing the report is important and to save time and to keep the management process simple, we decided to choose FilamentPHP Form Builder and Table Builder to create the stunning form and table for the clients. In this section, we divided the role by a middleware that check if user role is admin and provide different permission for each user.

4.4.2 Creating the interfaces

The POS interface

The POS interface is divided into different section with the left pane is for the list of items and the action to them, and the right pane is for item name with search box, so cashier can search with name or code of the product and select from there. Also, there should be option to select the applicable coupons and checkout for specified customer.

The Customer interface

The customer interface is quite simple while it will display the current cart, the customer will be able to know what is on the cart, which coupon has been applied and the total amount and the cart, the QR code for payment and also some advertising for products.

Chapter 5

Result

5.1 Applied functions

We have successfully implemented within our system these features below:

- **Checkout function:** We provide a new approach for customer registration on our system using facial recognition. It can identify unregistered customers and those who already have their information saved in our database.
- **Managing items function:** On the main screen of the POS, the cashier can manage the list of items available on our system. The cashier can also perform some fundamental tasks, such as adding items to the cart, changing the quantity, searching for items and applying an existing coupon for those products. And on the customer screen, they can view the total amount of items, their final prices.
- **Payment function:** We have included several ways for the customer to process their payment when buying items. Such standards include the ability to pay by card, by cash, and by using our generated QR that can automatically detect the total value of all items and display them on the customer's bank details, so they only need to proceed with the payment without the needs of filling the amount needed.
- **General manager functions:** We implemented on the admin screen various tabs for managing users, products, order, transaction... This helps to streamline the managing process.

Chapter 6

Conclusions

6.1 Conclusions

In conclusion, implementing a Point of Sale (POS) system tailored for pharmacists proves to be a transformative solution that significantly enhances efficiency, accuracy, and overall customer experience within the pharmacy setting. The integration of a specialized POS system not only streamlines the transaction process but also facilitates comprehensive inventory management, enabling pharmacists to monitor stock levels, expiration dates, and reorder supplies seamlessly.

Moreover, the implementation of a POS system offers robust reporting capabilities, providing valuable insights into sales trends, popular products, and customer preferences. This data-driven approach empowers pharmacists to make informed business decisions, optimize their product offerings, and enhance overall business performance.

Moreover, with the customer facial recognition technology represents a cutting-edge solution that enhances security, streamlines authentication processes, and offers a personalized and frictionless experience. Its ability to provide a seamless and secure means of identity verification marks a significant advancement in customer service, setting the stage for a future where convenience and security coexist harmoniously in various industries. As businesses continue to prioritize user experience and security, customer facial recognition stands out as a powerful tool to meet these evolving demands.

In essence, investing in a tailored POS system for pharmacists is a strategic decision that not only addresses the unique requirements of the pharmaceutical industry but also positions the pharmacy for long-term success by embracing innovation, optimizing operational processes, and ultimately delivering enhanced value to both patients and stakeholders.

6.2 Future works

Looking ahead, the future development of the facial recognition system for customers holds exciting possibilities, with a focus on enhancing user engagement, operational efficiency, and scalability. The following areas represent key directions for future work:

- **Mobile App Integration with Push Notifications:** Implementing a dedicated mobile application that seamlessly integrates with the facial recognition system can enhance user engagement. This app could provide customers with personalized notifications, offers, and updates based on their preferences and purchase history. Push notifications can serve as a direct communication channel, fostering stronger connections with customers and promoting loyalty.
- **Integration with Delivery Services:** Expanding the system to integrate with delivery services can further streamline the customer experience. By linking the facial recognition system with delivery companies, customers can enjoy a cohesive end-to-end service, from in-store interactions to doorstep deliveries. This integration could also facilitate real-time tracking and updates, ensuring a seamless and efficient delivery process.
- **Performance Optimization for Multiple POS Machines:** Improving the overall system performance to handle multiple Point of Sale (POS) machines concurrently is crucial for scalability. This includes optimizing the facial recognition algorithms, database management, and communication protocols to ensure a smooth and responsive experience, even during peak times. Robust performance across multiple POS terminals will be essential for accommodating the system's widespread adoption. Moreover, we achieved few tests on the potential

of the application to be deployed on Cloud and scaling with the implementations of Continuous Delivery that can be applied to build a functioning cluster in the future.

- **Scaling up process:** can be implemented via a cluster of Docker containers or Kubernetes cluster behind a Load balancer with decentralized database server and redis database. Also, the image of the customer is stored in S3 storage, so the container will only contains the application and the environment configuration.

By pursuing these future developments, the facial recognition POS system can evolve into a comprehensive solution that not only enhances security and efficiency but also enriches the overall customer experience and contributes to the long-term success of the business.

Chapter 7

References

- The state bank of VietNam website.
<https://sbv.gov.vn/webcenter/portal/en/home/sbv/statistic/settleactiv/gdqatmpos>
- face-api.js
<https://justadudewhohacks.github.io/face-api.js/docs/index.html>
- QR Code Application implementation
<https://medium.com/@flutterone/how-to-implement-a-qr-scanner-using-flutter-71>
- VietQR Banking Software requirements specification
https://developer.napas.com.vn/files/assets/files/Techspec_NapasCore_v1.1.pdf
-

Chapter 8

Appendix

8.1 CRUD diagrams

Product CRUD diagram

This diagram outlines the CRUD operations for managing products within the system. Users, particularly administrators, initiate the creation of a new product by adding its details to the system. Upon successful addition, the system confirms the product's inclusion. Additionally, users can navigate to the main product dashboard, offering a comprehensive list of available products. Within this dashboard, users possess the ability to effortlessly update or delete existing products as required, ensuring accurate and up-to-date product information within the system.

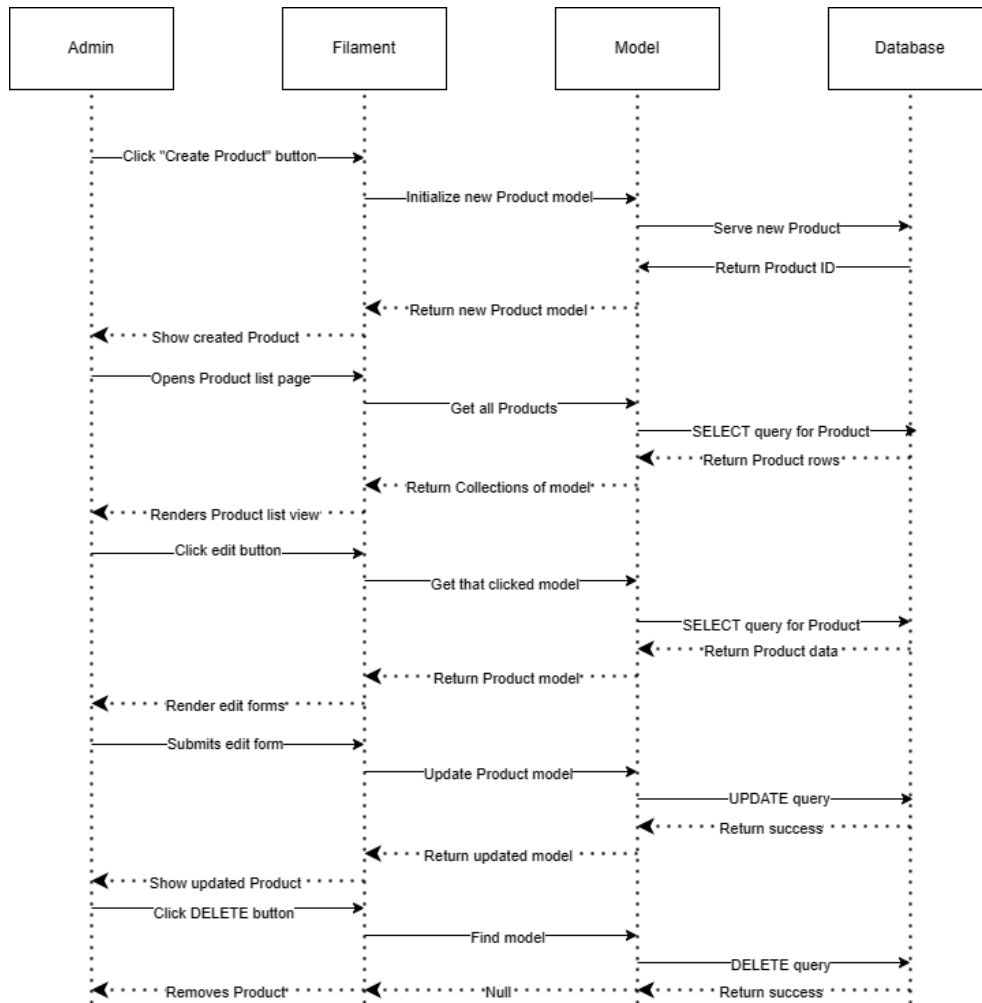


Figure 8.1: Product CRUD diagram

Order diagram

This order system diagram delineates the order lifecycle: Customers initiate orders, providing product and shipping details. Upon submission, the system generates order confirmations. Simultaneously, it updates the order database. Admin (or manager) can modify or delete orders, ensuring accuracy. The CRUD model allows seamless order management, facilitating creation, updating, and deletion of orders as necessary.

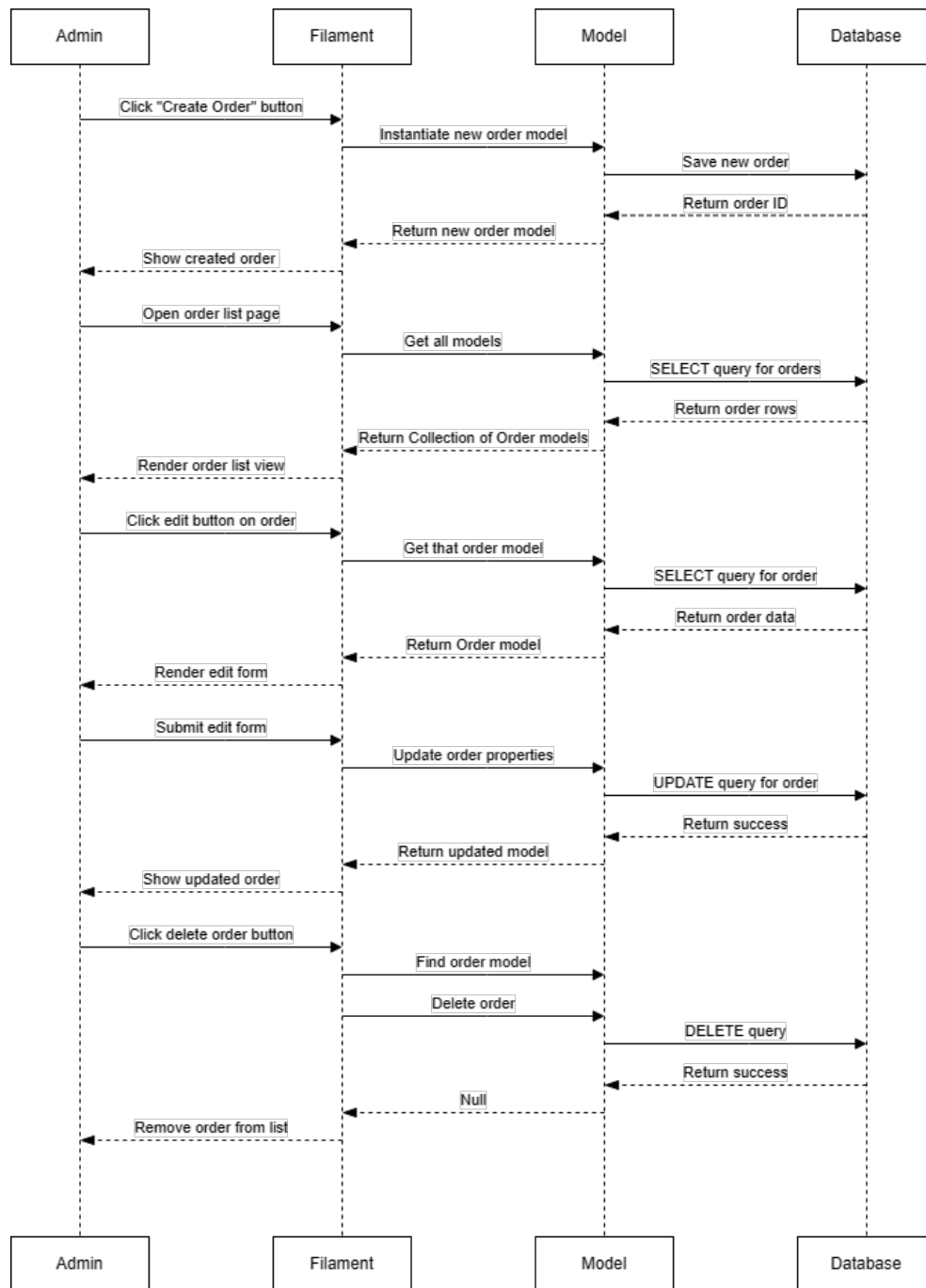


Figure 8.2: Order CRUD diagram

Coupon diagram

This coupon system diagram depicts the lifecycle of coupons: Admins create coupons, specifying discounts and expiration dates. Cashier can apply these during checkout on the customer behalf; the system verifies and adjusts the total accordingly. Admins access an interface to review, update, or delete coupons, ensuring accurate and timely

discounts. The system validates coupon eligibility, providing seamless customer experiences. This CRUD-based approach enables effective coupon management, allowing for the creation, modification, and removal of coupons as needed.

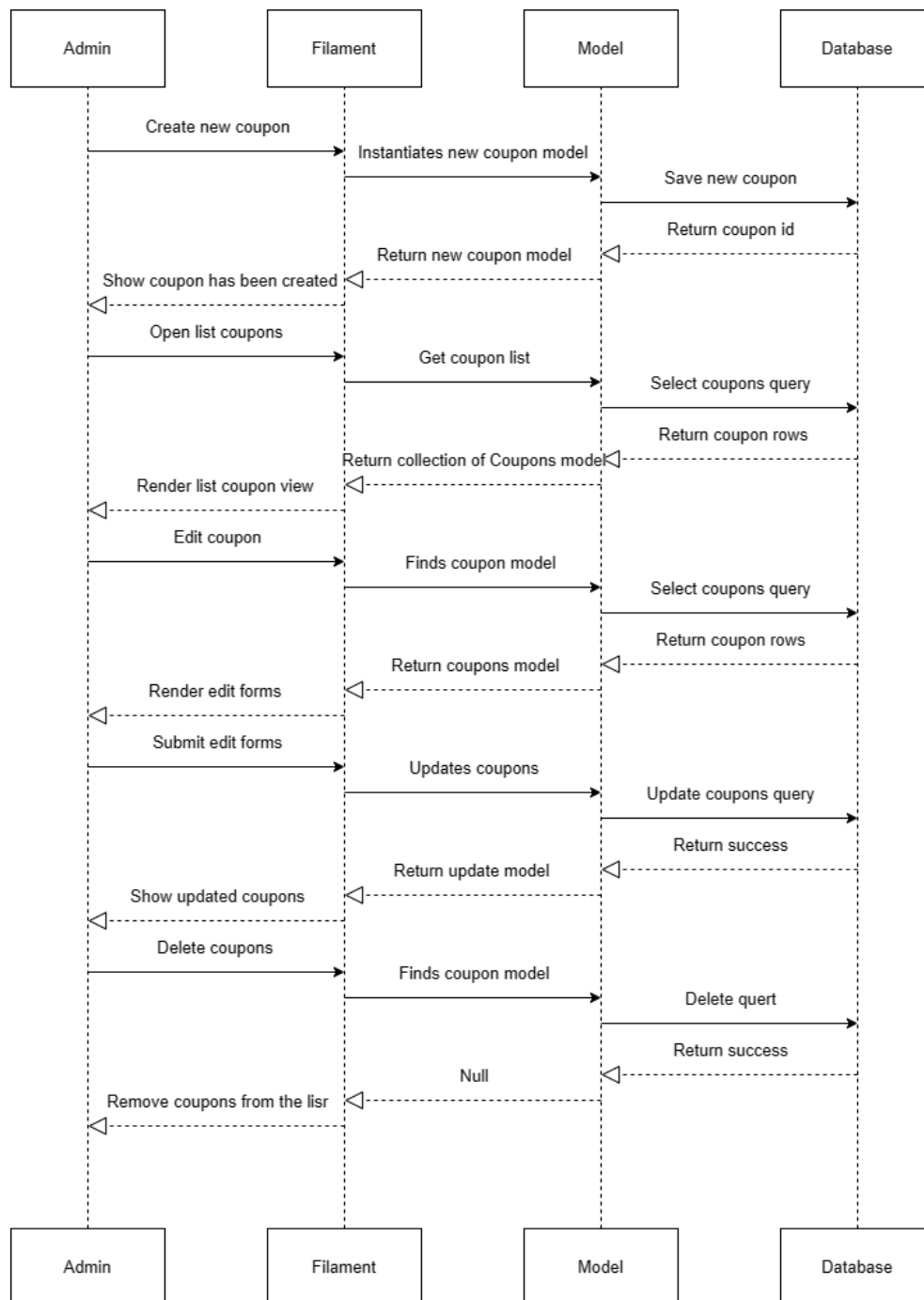


Figure 8.3: Coupon CRUD diagram

8.2 Metadata tables

users		
Attribute	Data type	Range of values
name*	String	min:4 max:191
email*	String	min: 6 max: 255
password*	String	min: 1 max: 255
role_id*	String	min: 1 max: 191

Table 8.1: User metadata table

bank_account		
Attribute	Data type	Range of values
bank_number*	String	min: 9 max: 15
owner_name*	String	min: 6 max: 191
bank	String	min: 1 max: 191

Table 8.2: Bank account metadata table

bank_transactions		
Attribute	Data type	Range of values
transaction_id*	String	min: 9 max: 15
account*	String	min: 6 max: 191
content	String	min: 1 max: 191
checked	boolean	True, False
type	int	

Table 8.3: Bank transaction metadata table

coupons		
Attribute	Data type	Range of values
coupon code*	String	min: 9 max: 15
started date*	String	min: 6 max: 191
expired date*	String	min: 1 max: 191
coupon type*	Boolean	True, False
coupon condition*	Array	
coupon minimum condition	Null	
coupon value*	String	min: 5 max: 191

Table 8.4: Coupon metadata table

customers

Attribute	Data type	Range of values
name*	String	min: 5 max: 191
mobile*	String	min: 5 max: 191
email*	String	min: 5 max: 191
face*		
address*	String	min: 5 max: 191
zalo_number*	String	min: 5 max: 191
credit*	int	min: 5 max: 191
age*	int	min: 5 max: 191
image*		min: 5 max: 191
details	String	min: 5 max: 191
updated_at	date	
created_at	date	

Table 8.5: Customer metadata table

products		
Attribute	Data type	Range of values
product_name*	String	min: 5 max: 191
product_description*	String	min: 5 max: 191
product_image*	String	min: 5 max: 191
quantity*	int	min: 5 max: 191
tax*	int	min: 5 max: 191
price_box_listing*	int	min: 5 max: 191
price_box_discounted*	int	min: 5 max: 191
price_item_listing*	int	min: 5 max: 191
price_item_discounted*	int	min: 5 max: 191
limit_by_aged*	int	min: 5 max: 191
limit_by_order*	int	min: 5 max: 191
SKU*	String	min: 5 max: 191
barcode*	String	min: 5 max: 191
ingredients*	String	min: 5 max: 191
brand*	String	min: 5 max: 191

Table 8.6: Product metadata table

qr_code		
Attribute	Data type	Range of values
amount*	double	min: 5 max: 191
code*	String	min: 5 max: 191
uuid*	object	min: 5 max: 191
content*	String	min: 5 max: 191
updated_at*	date	
created_at*	date	

Table 8.7: QR code metadata table

orders		
Attribute	Data type	Range of values
customer_id*	String	min: 9 max: 15
carts	Object	
payment_type	String	min: 1 max: 191
payment_record	String	True, False
coupon	Array	
value	Object	
cashier_id*	String	min: 5 max: 191

Table 8.8: Orders metadata table

car object		
Attribute	Data type	Range of values
product_id / coupon_id*	String	min: 9 max: 191
name	String	min: 9 max: 191
quantity	Integer	
price	Float	
photo	String	min: 9 max: 191
tax	Percentage	

Table 8.9: Car Object metadata table

value object		
Attribute	Data type	Range of values
subtotal	Float	
vat	Percentage	
quantity	Integer	
total_due	Float	
discount	Float	

Table 8.10: Value Object metadata table

Bibliography

[1]