VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Software Engineering

Report

Assignment Report

Group 6

1952069	Ho Trí Kháng
1952059	Phạm Nhựt Huy
1911667	Nguyễn Thanh Ngân
1952533	Ngô Thị Tú Vy
1911419	Trần Nguyễn Anh Khoa

HO CHI MINH CITY, AUGUST 2021

Contents

1 Task 1

1.1 Project's context

1.1.1 Project's stakeholder

1. **Customers:** The customers are the stakeholders because the company's economic success depends on them. They give feedback about the app during development. In return, the company receives money from the customers.

2. **Clerks:** The clerks are the stakeholders because like the customers, they contribute a lot of information for the apps since they have to interact with the system as well.

3. **System administrators:** The system administrators are the stakeholders because they ensure the work is done as prepared in both time and quality perspectives by reviewing all the detailed plans (tasks, assignments, ...).

4. **Restaurant owners, investors, shareholders:** The restaurant owners, investors and shareholders are also stakeholders because they give concrete guidance which affects directly the project.

1.1.2 Project's objective

The POS system is expected to reduce the peril caused by COVID-19 by increasing business intelligence, reduce wasted effort, and create an opportunity to scale to a large business. To do that, the POS should provide mechanisms to prepare an invoice for the customer as well as indicate the options for the customer to make payment with time-efficiency and high precision. **In conclusion, the main objectives for this project is** to design implement and test the intelligent web-based POS system that allows customers to order food/drink and use several different services via their devices without needing to do those tasks manually, aiming to solve the problem of limited manpower and improve QoS (quality-of-service)

1.1.3 Project's scope

By analyzing all the relevant stakeholders, constraints and objectives of this project as well as what the project is about, what is included, and what isn't. We have defined our scope of this project

- **A faster order processing time that improves productivity and user experiences:** The majority of the system depends on machines, which allows us to decrease food ordering time and payment time, thus increasing user experience.

- **Reducing operational cost and utilize manpower efficiently:** The cost of maintaining systems is lesser compare to the cost of hiring waiters, allowing money to be invested in other things.

- **A better user interface design to give high customer QoE:** A well-executed user interface facilitates effective interaction between the user and the program. It is important for the customer to order food without any struggle to order it. So

when designing a UI for your site, it's important to consider the user's expectations in terms of accessibility, visual aesthetics and ease of use.

- **Keeping the updates and management of the menu by the admin:** We want to limit the right to access some important functions of the system, as well as keep the data of the customers and food hidden from everyone but the manager. That's why we want to keep the menu updated if and only if the restaurant management intends to change it through the admin console.
- **Restaurant billing system:** A separate billing system also needs to be handled that can generate the customer bills at the time of payment. It will get the data of the food that the customers have ordered and generate the bill based on that data.

1.2 Functional and non-functional requirements

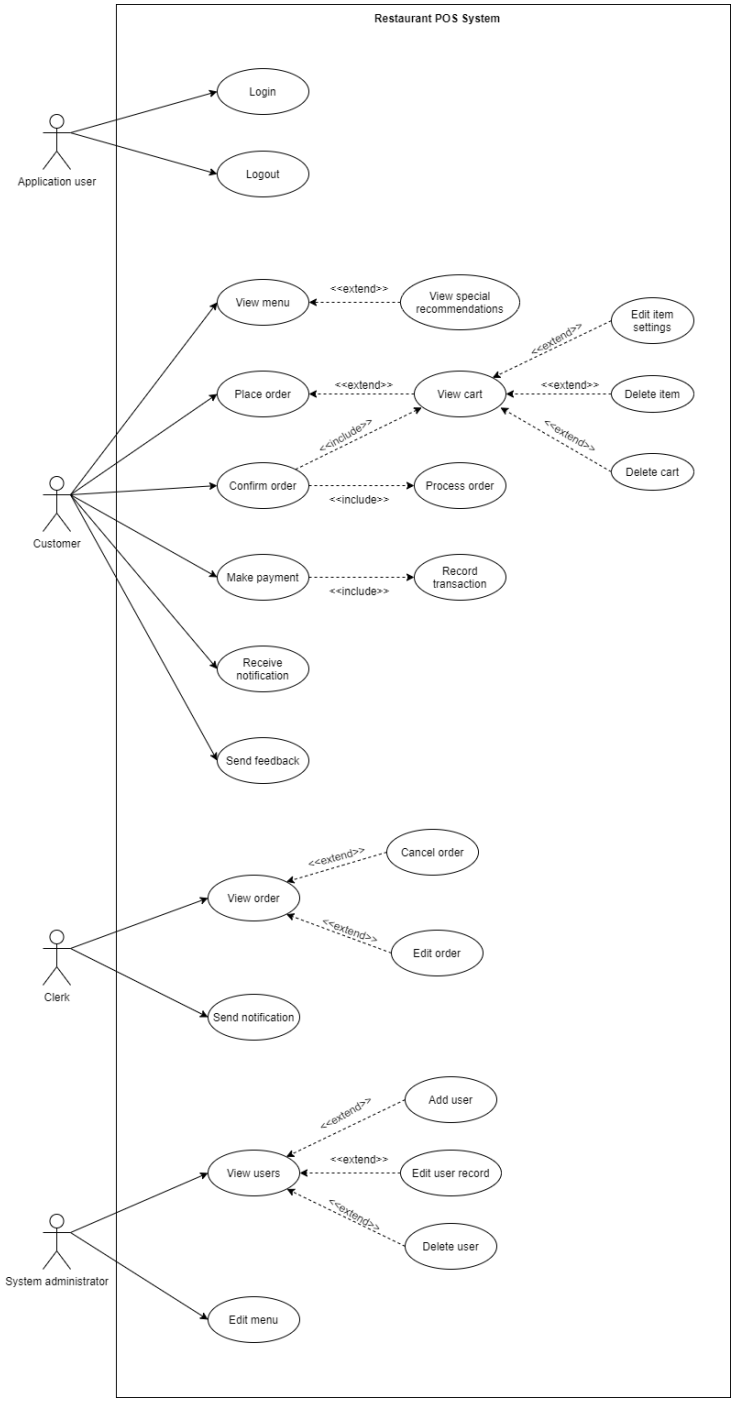
1.2.1 Functional requirements

No.	Requirement
R0	Customer can scan QR code or type the URL to access the POS web page .
R1	The system should display the food/drink menu on any devices with different screen sizes. E.g. tablets, laptops, smart phones.
R2	The menu can include image/video and description for each food/drink item. Some special recommendations should also be shown on the menu.
R3	Customer can choose takeaway or dine-in options, the services may act differently according to each option.
R4	Customer can select one or more food/drink items by clicking on their images shown on the menu screen.
R5	An item detailed screen which allow customer to choose item option such as quantity, portion size, toppings, spicy level. Customer can write notes if their preferences are not present on the screen.
R6	Customer can edit the order before submitting the order to the queue. In case the order has been submitted, customer can cancel the order within 3 minutes.
R7	In case a food/drink item is out of stock, its status should be shown on the menu screen. This action can be done manually by clerk or system admin.
R8	Customer, clerk and system administrator have to log in to use the system with their given username and password.
R9	Clerk and administrator after logging in can track the orders queue on screen.
R10	The system should provide a screen for clerk to manage orders. Clerk or system admin can edit or cancel an order manually.
R11	Clerk can send notifications to customer regarding the order status. E.g order has been accepted, order is being prepared, order is cancelled.
R12	There should be an admin panel where admin can perform create, delete and update operations on menu. System admin can change menu layout, or add some new features to the system.
R13	System administrator is able to create/delete users and assign roles to users.
R14	A payment page where customer can make payment. The system should support payment by credit or debit card.
R15	After the payment is made, there is a feedback page that allows customer to give rating and comment for the meal as well as restaurant services.

1.2.2 Non-functional requirements

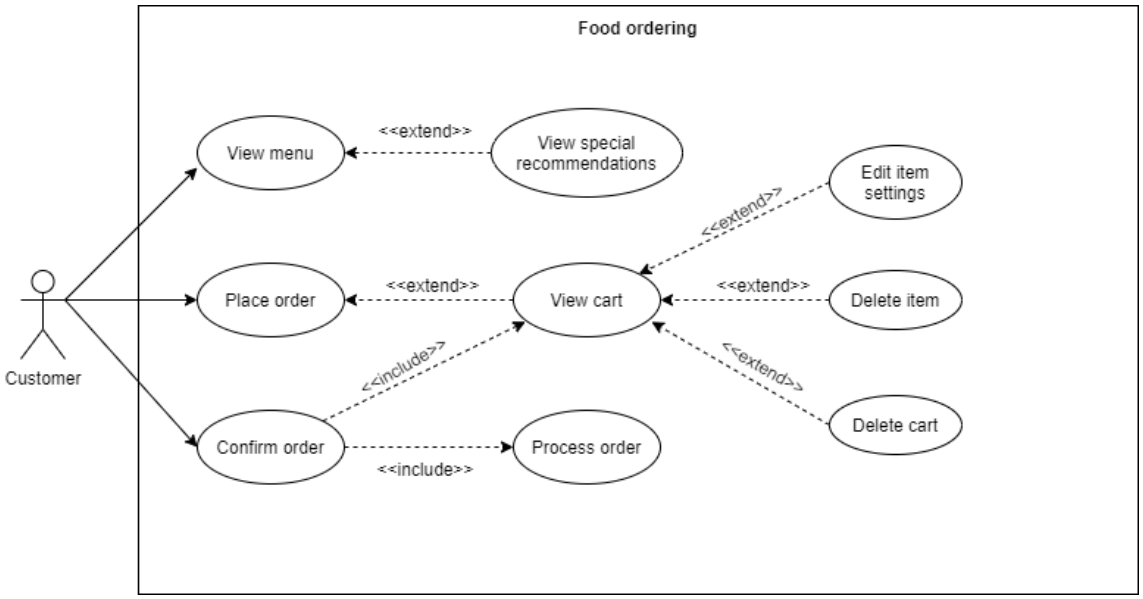
No.	Requirement
R0	All the services including ordering and payment should be performed on web, allowing non direct contact between customers and clerks.
R1	The system can manage up to 30 orders at the same time.
R2	System uptime higher than 90% of the working day.
R3	System's response time less than 1 second
R4	The system is developed using web technology and should be usable from many devices (smartphone, PC) and support almost all platforms (iOS, Android, Windows, ...).
R5	The customer's view needs to be simple and self-explanatory that every customer can easily use.
R6	System allows at least 300 transactions per day (by credit or debit card).

1.2.3 Use-case diagram



1.3 Specific feature: Food ordering

1.3.1 Use case diagram



1.3.2 Table format

In the food ordering feature, we split it into 3 main use cases: view menu, place order, confirm order View menu

Use case name	View menu
Actors	Customer
Description	This use case allows customer to see menu on the screen.
Preconditions	Customer must log in successfully.
Normal Flow	1. System displays the menu with all available categories, e.g., sea food, juice, etc. 2. Customer selects the category he/she needs. 3. System displays the menu items in the selected category.
Alternative Flows	Alternative 1: at step 2 Customer clicks on the "special recommendations" displayed at the top of the menu page. System displays the recommended items, combos, ... Use case ends.
Exceptions	None

Place order

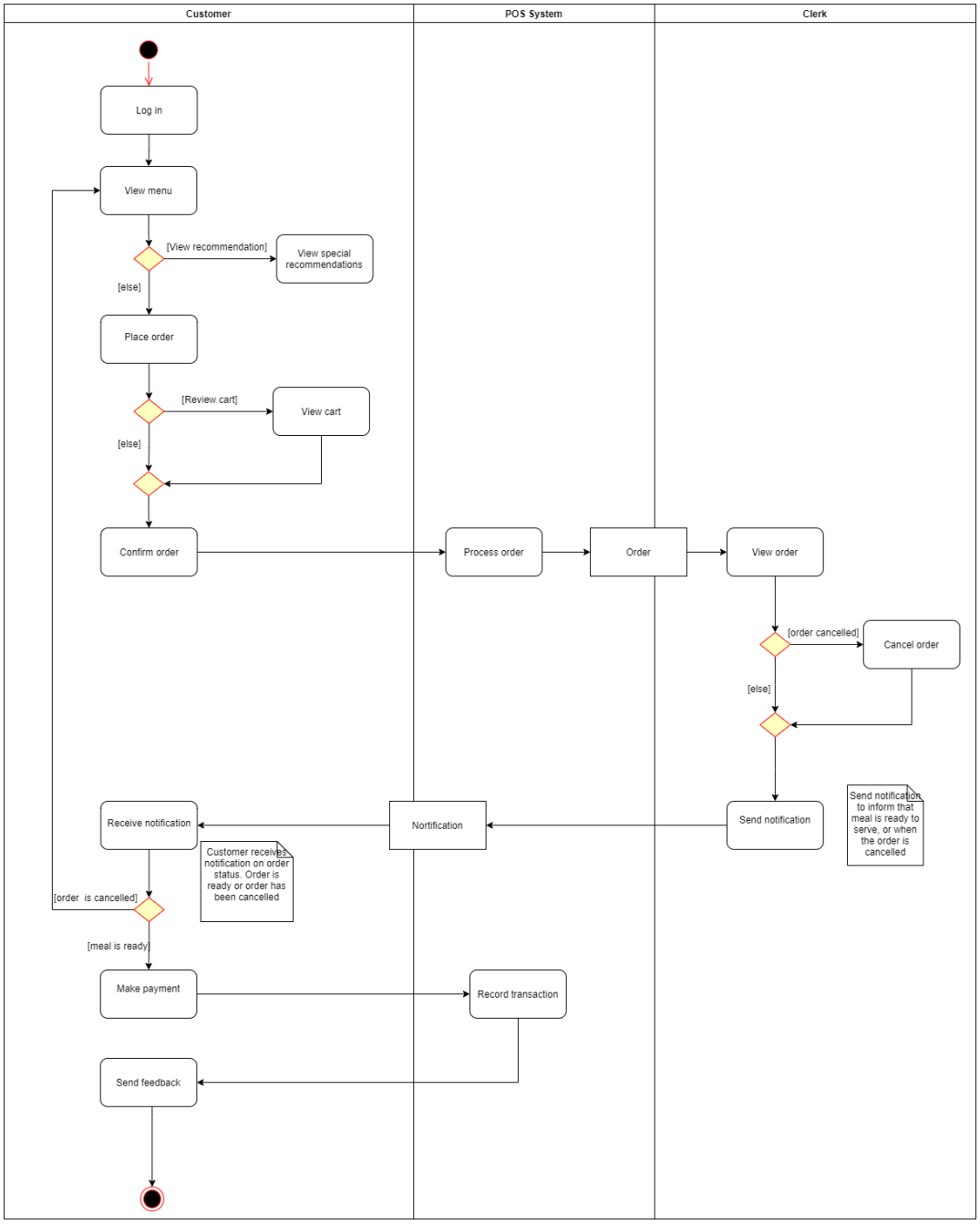
Use case name	Place order
Actors	Customer
Description	This use case allows customer to order more than one food/drink items displayed on the screen.
Preconditions	Customer must login successfully and see the menu.
Normal Flow	1. Customer selects the "+" button on an item widget of the menu to add that item to the cart. 2. System displays a detailed item screen. 3. Customer selects item settings (quantities, ingredients, etc.) or writes notes. 4. Customer selects "Add to cart". 5. System adds the item to the current cart and displays the updated cart.
Alternative Flows	Alternative 1: at step 3 Customer accepts the default settings. Go to step 4. Exception 2: at step 3 Customer selects the "x" button to discard the item. Use case ends. Exception 3: at step 4 Customer selects the "x" button to discard the item. Use case ends.

Confirm order

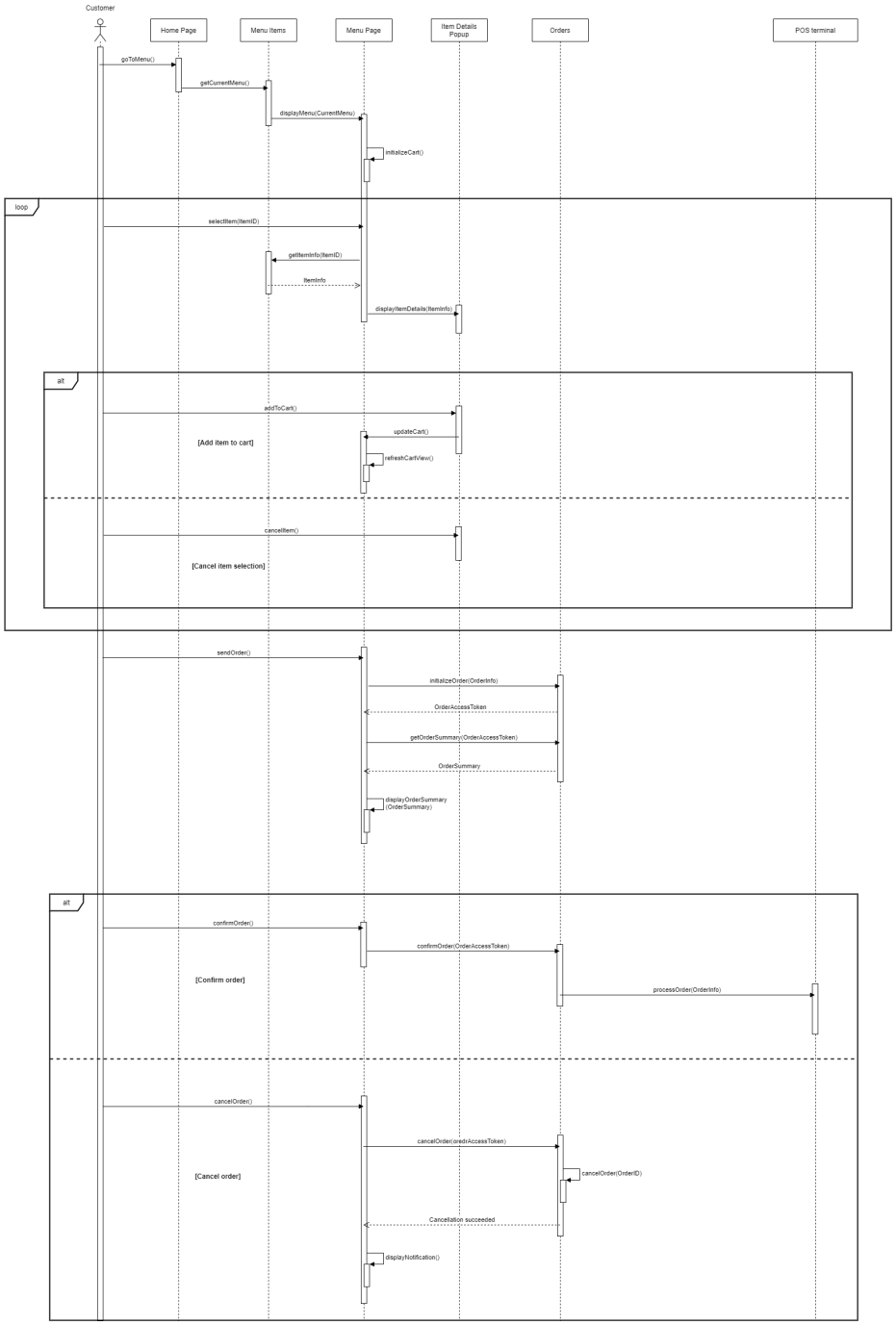
Use case name	Confirm order
Actors	Customer
Description	This use case allows the customer to submit his/her order.
Preconditions	The customer must log in successfully.
Normal Flow	1. Customer clicks the cart icon to open the cart. 2. System displays the cart with all the items that he/she has selected. 3. Customer selects the “Make payment” button to go to payment page. 4. System displays the payment page. 5. Customer fills in payment information (card-number, CVV) and press "Payment". 6. System displays a message showing that the order has been submitted.
Alternative Flows	None
Exceptions	Exception 1: at step 3 3a. Customer selects the “x” button. Use case ends. Exception 2: at step 5 Customer selects the “Cancel” button. Use case ends.

2 Task 2

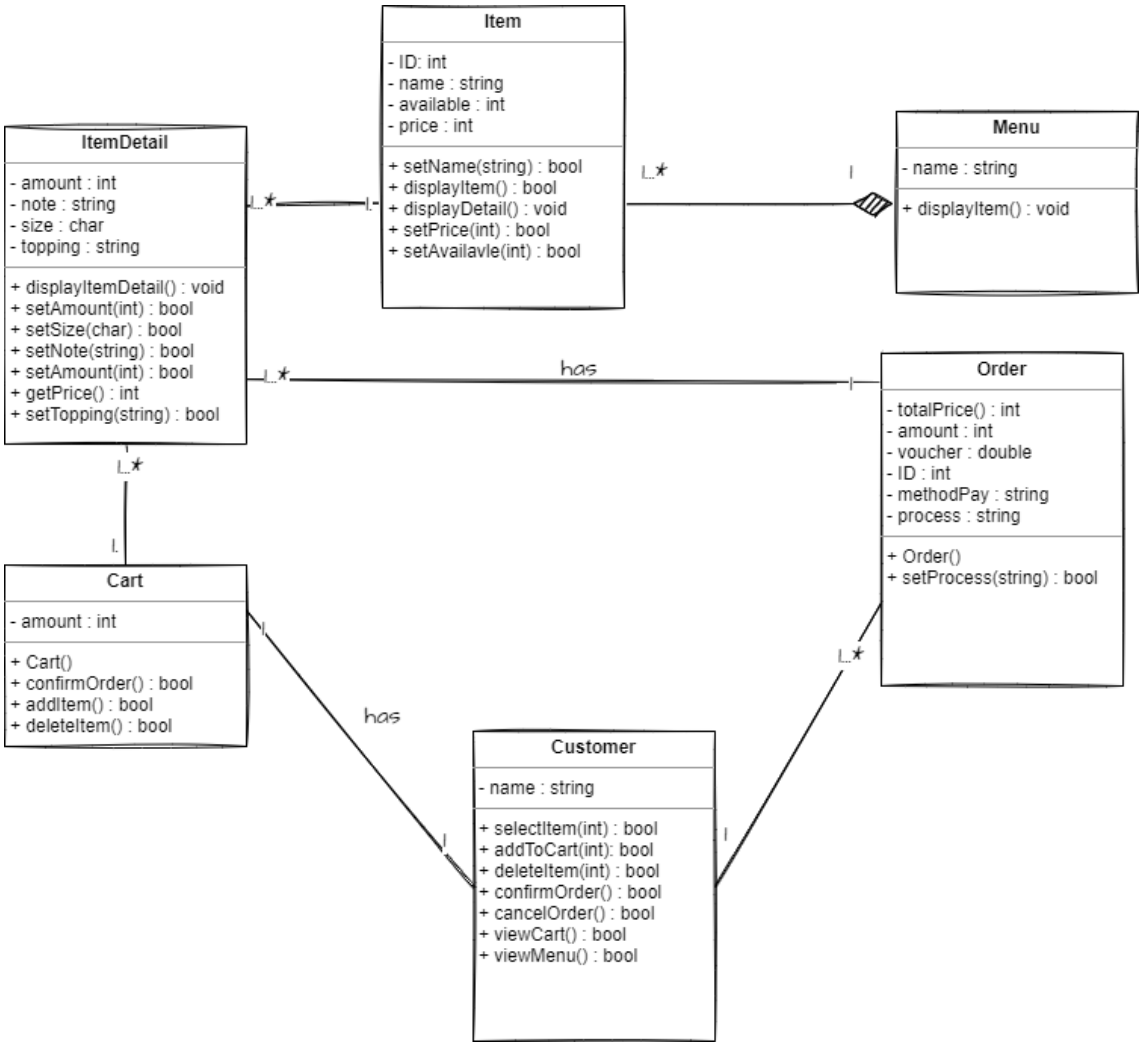
2.1 Activity diagram for major functional requirements



2.2 Sequence diagram for food ordering



2.3 Class diagram for food ordering



3 Task 3

3.1 Describe architectural approach

3.1.1 Model-View-Controller

What is MVC?

Model-View-Controller (MVC) is one of the most well-known architecture patterns in many web-based systems and MVC is supported by many programming languages. It was used for desktop graphical user interfaces but nowadays is used in designing mobile apps and web apps. MVC architecture pattern separates presentation and interaction from the system data. MVC is structured into three logical components: Model, View and Controller:

- **Model:** Manage the system data, associated operations on data. MVC represents the data to the user and defines the storage of all the application's data objects.
- **View:** Defines and manages how the data is presented to the user which is also associated with the User Interface (UI). View also handle interaction between user and Controller
- **Controller:** Manages user interaction and passes these interactions to the View and Model. The controller completes the cycle of taking the user output, converting it into desired messages, and passing them on to the views(UI).

Advantages:

- MVC architecture will separate the user interface and business logic
- Components are reusable.
- Easy to maintain.
- Different components of the application in MVC can be independently deployed and maintained.
- This architecture helpt to test components independently.
- There are multiple ways to view and interact with data

Why do we use MVC in POS?

Because we build a large system with many components, MVC helps us to control. We can improve the components which might not affect the others. Furthermore, our system might have many components in the future and there will be many developments in the functions, so the way we control the code into 3 parts also helps us to deploy and maintain more easily. The way we split into Model, View, Controller also helps us to reuse the modules more efficiently because each block will have the specific task, not be mixed with the others. In addition, POS requires a lot of business logic so we can control it separately with the user interface. Also, POS will have many ways to interact with data so MVC is a suitable candidate.

MVC in POS

Model: Build up the database with: food, transactions, users or payment. Also, Model in charge of operating data on the database. View: The user interface of the application. View is what we see in the application, what is the user interface of customers, clerk or the administrator. View also get the user input and pass it to the controller and get the output from the Controller and update on the screen Controller: Controller is the bridge between Model and View. Controllers take care of managing the input and output and converting it to the desired block.

3.1.2 Client-Server

What is client-server architecture pattern?

Client-server architecture pattern is a computing model in which the functionality of the system is organized into services, and each service is managed and delivered from a separate server. The clients can consume and make use of these services by accessing the servers. Client-server architecture pattern is also known as a networking computing model or client-server network because all the requests and services are delivered over a network.

Terminology

- **Client:** A software or application which takes the input and sends the request to the server.
- **Server:** A software that receives and processes requests from clients.
- **Service:** Functionality provided by the server to the client.

Advantages of client-server architecture pattern

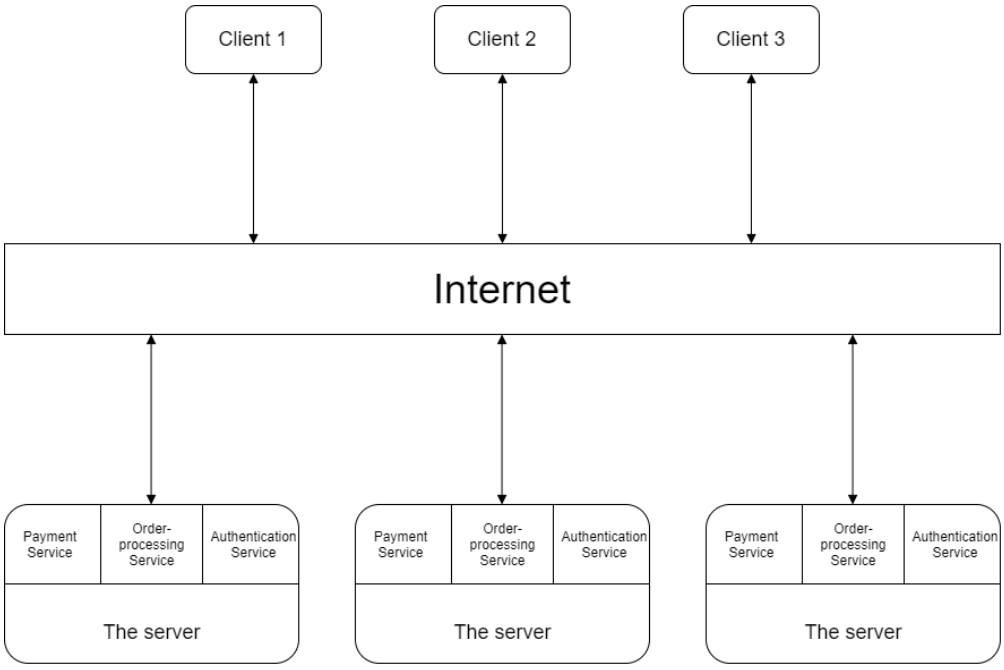
- Remote access: The client can access the server from a range of location through a network.
- Reusable: General functionality can be available to all clients, and it does not need to be implemented again in each service.
- Security: Clients and servers can complete tasks independently, and the data is centralized in the server, so the processing of sensitive data is carried out on server side.

Why we choose the client-server architecture pattern?

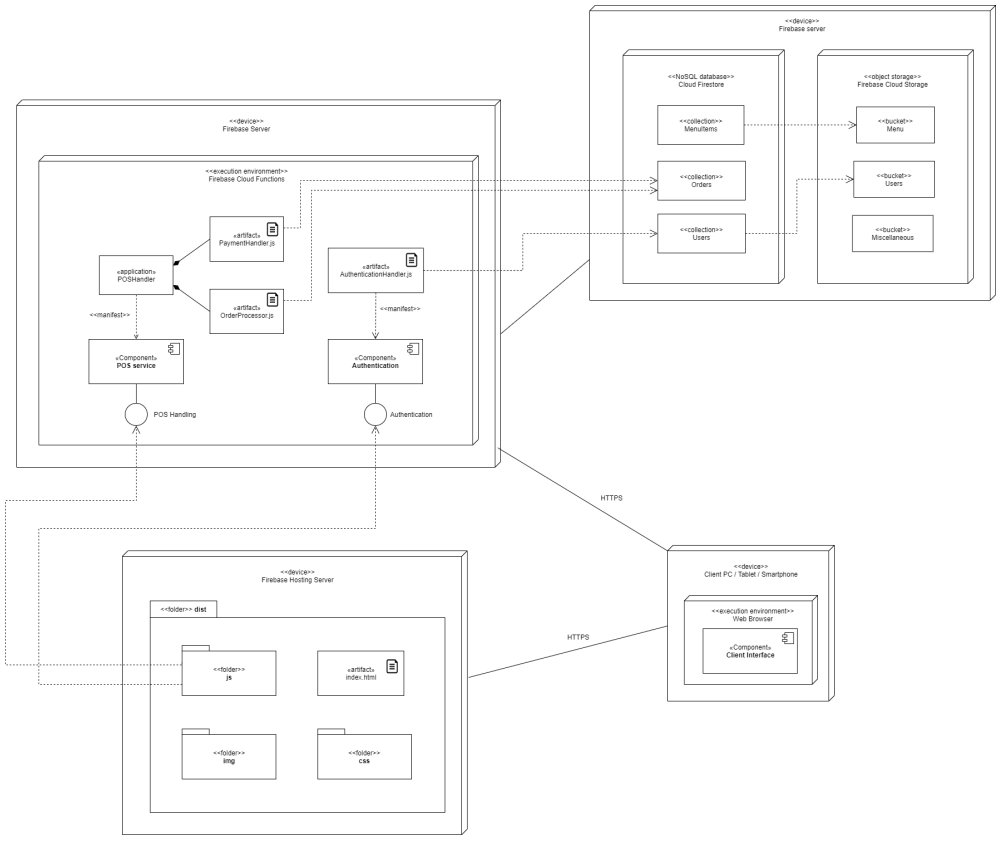
The POS system is a web-based system that provides methods for the customers to purchase their meals without having them make any direct contact with the clerk. The POS system also records data of the users after they make their payment, so the recorded data have to be highly secured remotely in the server. Based on those reasons, the client-server architecture pattern is a good candidate for the system.

Services needed

- **Payment service:** The payment service from the restaurant server takes the information from the customers' bank account and uses that information to make a money transfer request to the bank.
- **Order-processing service:** The process order service from the restaurant server helps the kitchen get the order from customers without having them make any contacts to the clerk directly. The information about the order is transferred through the service and ends up displayed in the clerk terminal.
- **Authentication service:** The authentication service from the restaurant server can check if the login information given by the customers matches the database. The service will authorize if the login information is correct. Otherwise, it will stop the user from accessing the system.



3.2 Implementation diagram



4 Implementation

4.1 Restaurant POS application

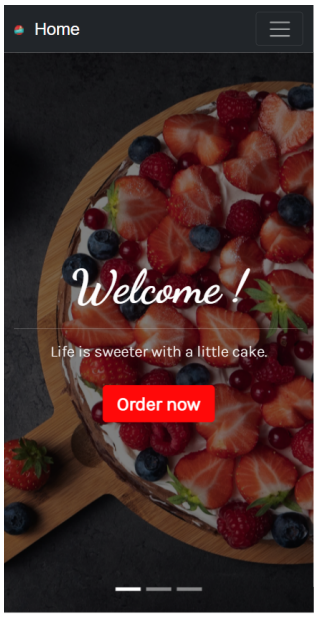
We have implemented a web application for our POS system. About the technical decisions, we choose Vue.js for front-end and Firebase for back-end.

The link to our website: https://restaurant-pos-group6.web.app/

Our Github repository for the project can be found here.

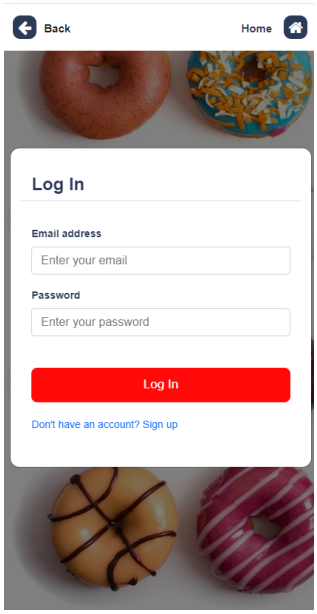
4.2 User manual

1. Access the website. You will see the home screen as below



The "Home" page

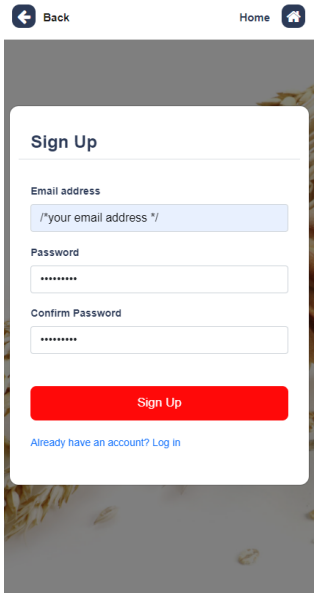
2. Selecting "Order now" will move you to the log in screen:



The "Log in" page

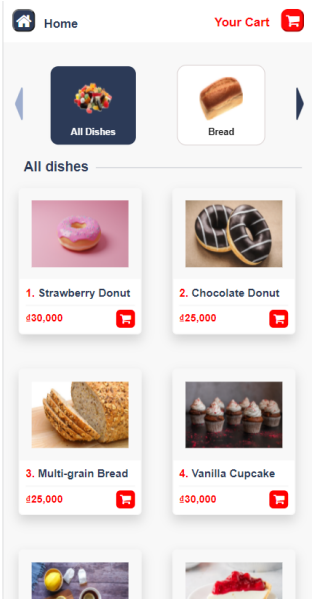
3. Enter username and password to log in to the system. If you haven't create an

account, select "Sign up" to create your account:



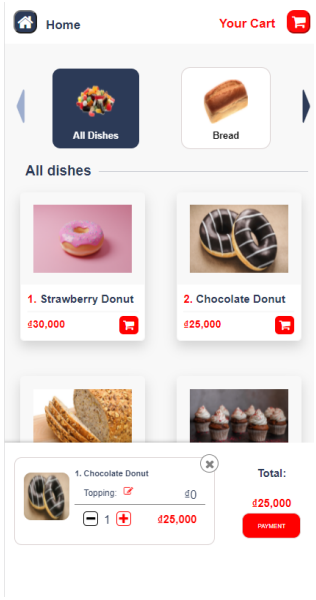
The "Sign up" page

4. Use the new account to sign up to the system. It will move you back to the home page. Now if you click "Order now", the menu page will be displayed:



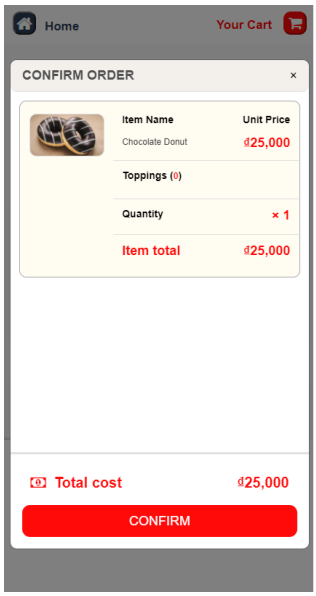
The "Menu" page

5. Click on a category icon from the filter menu displayed on the top of the page to filter the items according to that category. If "All dishes" is selected, it will display all the item. To order an item, click on the small cart button at the bottom right corner of each item widget. When you click "Your cart". The cart will appear showing all the items that have been selected:



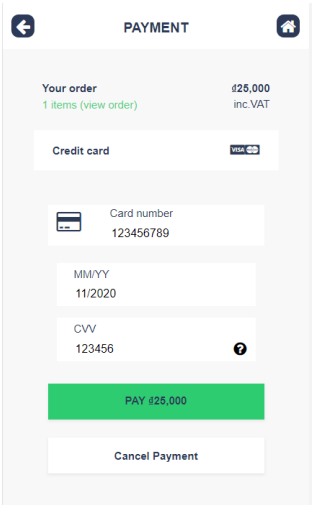
The "Cart" component of the "Menu" page

6. You can add or decrease quantity, or add other item to the cart. After that, click on "Payment" button, the detail item screen will be displayed:



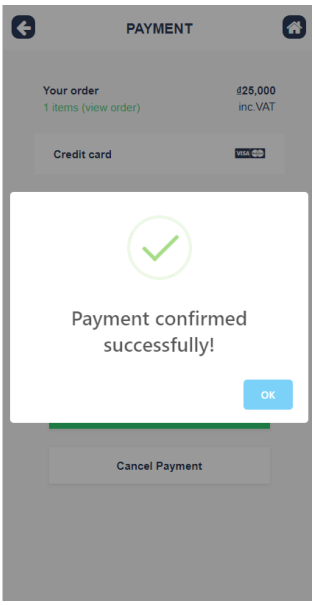
The confirm dialog displayed for user to review the order

7. Press "Confirm" to confirm the order, you will be moved to the payment page:



The "Payment" page

8. Enter all the necessary payment information. If you want to cancel the payment, click "Cancel payment" to go back to the menu page. Otherwise, click "Make payment", a small window will pop up showing that payment has been made successfully:



A dialog will pop up when the payment is made successfully