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

SOFTWARE ENGINEERING (CO3001) - CC02

Assignment 5

Advisor: Quản Thành Thơ
Students: Nguyễn Đình Sáng - 1952955
Lỉ Quang Tiến - 1953019
Huỳnh Minh Trí - 1953041
Dương Xuân Anh Tú - 1852845
Nguyễn Quốc Việt - 1953096

HO CHI MINH CITY, NOV 2021

Contents

1 Workload

No.	Date	Task	Member
1	10/09/2021	Meeting, dicussion about assignment	All members
2	11-15/09/2021	Task 1.1, Task 1.2, Task 1.3	Trí, Tú, Việt
3	15/09/2021	Meeting review task answers	All members
4	16-20/09/2021	Complete Latex report for task 1	Tiến, Việt, Sáng, Tú
5	21-24/09/2021	Web prototype on Figma phase 1	Sáng, Trí, Tiến
6	24/09/2021	Meeting review Latex report	All members
7	29/09/2021	Meeting review requirements	All members
8	02/10/2021	Task3.1	Tiến, Tú
9	02/10/2021	Task3.2	Trí, Sáng, Việt
10	20/10/2021	Meeting review Latex report	All members
11	04/11/2021	Meeting plan presentation video, code web for task 4	All members
12	11/11/2021	Presentation our works for Mr.Tho	All members
13	03-20/11/2021	Improving website, making the final report	All members

2 Task 1: Requirement elicitation

2.1 Task 1.1

2.1.1 Requirement

Question: Identify the context of this project. Who are relevant stakeholders? What are expected to be done? What are the scopes of the project?

2.1.2 The context of the project

- Restaurants are at greater danger than ever during the coronavirus outbreak.
- Require technologies that improve business intelligence, decrease wasted effort, and provide the ability to expand to a huge company.
- Such systems should provide for take-out options.

Therefore, Point of Sale (POS) offers a good choice for the demanding system. A point-of-sale (POS) transaction is what takes place between a merchant and a customer when a product or service is purchased, commonly using a point of sale system to complete the transaction.

2.1.3 Relevant stakeholders

1. Project manager.
2. Team members.
3. Managers.
4. Resource managers.
5. Executives.
6. Senior manager.
7. Company owners.
8. Investors.
9. Sponsors.
10. Financiers.
11. Clerks.
12. Consultants.
13. Customers.
14. End users.

2.1.4 Expectations

1. Requirement elicitation: Identify the general requirements and draw its use-case diagram.
2. System modeling: Draw an activity diagram, sequence diagram and class diagram.
3. Architecture design: Describe an architectural approach used to implement the desired system and draw its diagram.
4. Implementation – Sprint 1 : Implement a Minimum Viable Product (MVP) based on the given menu screen.
5. Implementation – Sprint 2 : Implement the MVP for shown screens and demonstrate the whole project.
6. Web-based application.

2.1.5 Scopes

1. Features of project's website:
    - Friendly-user items menu.
    - Friendly-manager order and transaction management.
    - Manage Orders.
    - Manage User's Profile.
    - Notification Configure.
2. Purpose of Project:
    - Understand the use-case of a Restaurant Management Project, particularly use-case of POS system.
    - Implementing a real website with some basic features: payment, billing, food order,...
3. Resources Specification:
    - 5 CS members.
    - Some members has experienced on Mobile. application front and back end.
4. Zero demand:
    - Support with some basic functions: payment, billing, food order,...
    - Non-direct contact between Clerks and Customers.
    - QR code to browser instead of install app.
    - Usable from a mobile device, a tablet device or a normal computer/ laptop.
    - Extendable to use in multiple restaurants in the future.
    - Current transactions = 300 orders per day.
5. Outcome of this project:
    - Understand and implement the MVP for screen showing in Figure 2 and Figure 3.
    - Build a real mini website that follow the MVP for screen.

2.2 Task 1.2

2.2.1 Requirement

Question: Describe all functional and non-functional requirements of the desired system. Draw a use-case diagram for the whole system.

2.2.2 Functional and non-functional of the desired system

Functional:

- User:
  - The user can log in to the system by scanning QR codes.
  - The user can choose from a menu of options.
  - The user can modify their existing order by adding additional items.
  - The user has the option of removing individual items or all items from their current order.
  - The user can check the status of their current order.
  - The user can place an order.
  - The user can view payment information.
- Admin:
  - Admin can add/update/delete food items to/from the menu.
  - Admin can update the price for a given food item.
  - Admin can update additional information for a given food item (out of service,sale,. . .)
  - Admin can view transaction bills.
- System:
  - System could display the order process.
- Kitchen:
  - Kitchen can view a list of confirmed orders.
  - Kitchen can update the status of the dishes (ready or not).
- Clerk:
  - Clerk can confirm the order from the customer.
  - Clerk can view feedback from customers.
  - Clerk can update the status of the dishes (delivered or not).

Non-functional:

- The system should allow non-direct contact between Clerks and Customers.
- The system should be extendable to use in multiple restaurants in the future.
- The website should respond in less than 0.5 second.
- The system should operate normally from 7am to 9pm, 7days/week.
- The number of current transactions is about 300 orders per day.
- The maintenance fee for the system should be less than $300 a month.

2.2.3 Use-case diagram of the whole system

Figure. POS system Use-case

2.3 Task 1.3

2.3.1 Requirement

Question: Choose one specific feature, i.e. food ordering, table reservation, customer management. Draw its use-case diagram and describe the use-case using a table format.

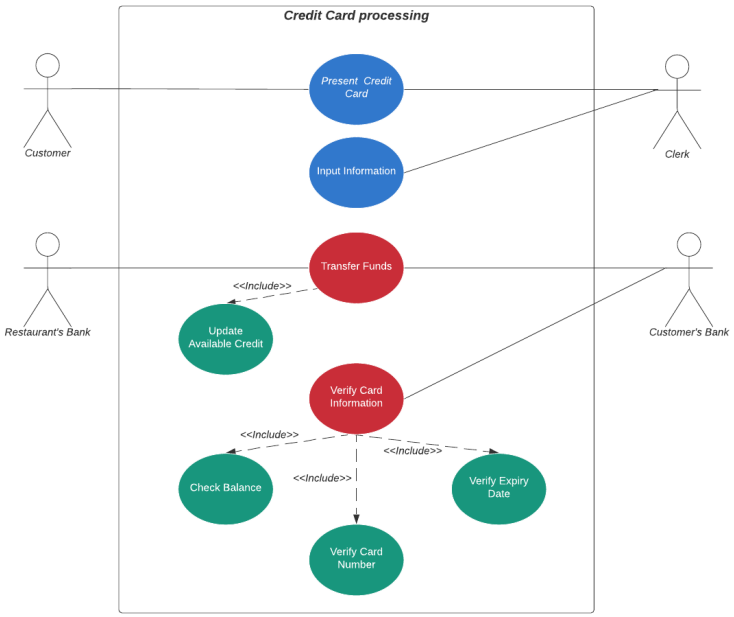
2.3.2 Overview

POS system consists of many features in order to give customers a best service without having direct contact, one of those is Credit Card Processing system. Most businesses rely on credit card processors to handle the details of accepting credit and debit cards. Credit card processing is a critical service—it ensures that customers can simply and quickly checkout.

2.3.3 How does credit card processing work?

First, customers will present their credit card information for payment. In store, consumers swipe cards on payment terminal. Online, consumers present credit and debit cards through websites and apps via many payment gateways.
The payment information is next sent to the processor, who communicates with the customer's bank via the appropriate card networks (such as Visa or Mastercard). The customer's bank approves or denies the transaction. Approval is dependent on detailed verification including card number validity, sufficient available funds, and other factors.
That approval is sent back through to merchant's payment processor and then finally back to the terminal or credit card reader. Approved transactions are batched for settlement typically at the end of each business day. The customers' accounts are charged for the transactions, with deposits then made into the merchant bank account.

2.3.4 Credit Card Processing Use Case



2.3.5 Describe Use Case in Table format

Credit card processing feature is quite complicated due the involvement of many actors, the tables below will give valuable insights into it.

Name	**Input Information**
Actor	Clerk
Description	The clerk uses the payment terminal to setup the payment processor, input the amount of credit that should be paid, the Account number of Restaurant's Bank
Pre-condition	The customers have already placed an order of dishes
Normal flow	1. Clerk sends the information to payment terminal

Name	**Present Credit Card**
Actor	Customer, Clerk
Description	The customers will have to present the Credit card to the clerk for the credit card payment process if in restaurant, enter the information about the card if online
Pre-condition	The clerk has already submitted the information of the restaurant's bank and bill to the payment terminal
Normal flow	1. Clerk sends the information to payment terminal 2. Present the Credit card 3. Enter the PIN code

Name	**Verify Card Information**
Actor	Customer's Bank
Description	The payment processor gets the credit card information that has been embedded inside the card together with the PIN code, sends it to the terminal then customer's bank to process, the bank checks the information if correct then approves the payment else denies the payment
Pre-condition	Customers must insert a PIN code so that the payment terminal has enough information to make a link to the bank
Normal flow	1. Clerk sends the information to payment terminal 2. Present the Credit card 3. Enter the PIN code 4. Verify card number, expiry date, available credit if it is satisfy to proceed payment
Exception	Exception 1: at step 3 3a. If the PIN code is not correct, the payment processor will ask the customers to re-enter the code maximum to three times.

Name	**Transfer Funds**
Actor	Customer's Bank, Restaurant's Bank
Description	Like an usual money transaction, the customer's bank updates the amount of credit available in customer's account and transfer an amount in the billing to the restaurant's account
Pre-condition	Card information has been checked fully and the customer's bank has approved the payment
Normal flow	1. Clerk sends the information to payment terminal 2. Present the Credit card 3. Enter the PIN code 4. Verify card number, expiry date, available credit if it is satisfy to proceed payment 5. Minus an amount in customer's credit and plus it to the restaurant's credit. Check the payment has been done. 6. Confirmation
Exception	Exception 2: at step 4 4a. If any of the information is not correct, the customer's bank will deny the payment and customers have to re-do from step 2

3 Task 2: System modelling

3.1 Task 2.1

3.1.1 Requirement

Question: Draw an activity diagram to capture Major (not all) functional requirements of the desired system.

3.1.2 Actors, flows, and activity diagram

The POS system consists of four actors that are *Customer*, *Clerk*, *POS terminal*, and *Kitchen*. The activity process includes **12 main steps**:

Step 1: Customers scan the QR code to sign in the restaurant web application.

Step 2: The customers then can select a menu and choose dishes they want to enjoy.

Step 3: Customers can simultaneously edit the order, which is inserting a new dish, deleting a dish, modifying the quantity of a dish, and view the order if it's necessary.

Step 4: When the customers finally finish their selection, the order is placed to clerks.

Step 5: The clerks receive the order, make confirmation and send it to the POS terminal.

Step 6: Based on the inputted calculations of ingredients, the terminal processes the order.

Step 7: If there are enough ingredients, the terminal return a message "Successfully ordered" to the customers, records the order to the database and sends it to the kitchen. Otherwise, a notification "Failed ordering" will be sent to the customers by the clerks requiring them to make another order based on provided information about availability of food and the process comes back again at *step 2*.

Step 8: If their order is successfully confirmed, they have to make a payment.

Step 9: If the customers want to pay by cash, they have to send an amount of money to the clerks. The clerks check the money. If it's sufficient corresponding to the order, the payment is successfully confirmed. Otherwise the process comes back to *step 8*. Or if the customers want to pay by credit card, the payment is successfully confirmed unless either their balance is insufficient or they enter wrong PIN code, which leads the process to *step 8* again.

Step 10: The chiefs in the kitchen workplace cook the dishes corresponding to the order.

Step 11: The cooked food is then sent to the clerks.

Step 12: The customers are served the food by the clerks.

Step 13: The customers then enjoy their dishes. Whenever they finish their food, they can send feedback to the clerks.

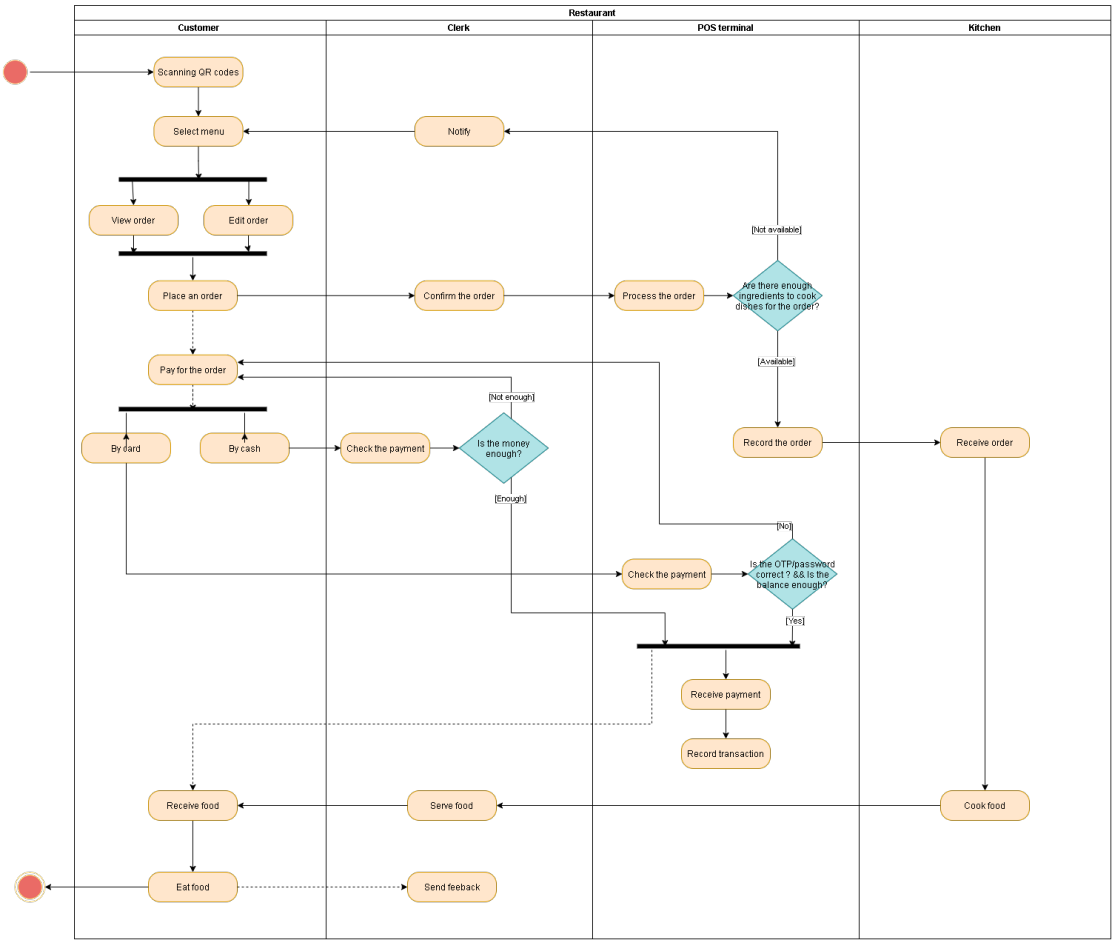


Figure. The activity diagram

This is the link for fully view purpose: Click me

3.2 Task 2.2

3.2.1 Requirement

Question: Draw a sequence diagram for use-case in Task 1.3.

3.2.2 Sequence diagram description

The payment flow by credit card will have three processes namely as follow:

- Checking card invalidation.
- Checking PIN invalidation.
- Checking funds is sufficient.

Firstly, the customer presents the bank card to the clerk who will insert the card to the POS machine, after that the POS machine communicates with the bank of the card to check if the credit card is valid or not. In this process, we have 2 situations:

- If the card is invalid, the bank service sends a response to the POS machine and the clerk will inform the customer that there is a problem with his/her card.
- Else, the card is accepted and a PIN is required from the customer.

Secondly, PIN will be sent to the bank service to check, the POS machine receives a response from the bank service:

- If the PIN is wrong, the customer will know that and he/she can fill it again.
- Else, the clerk will enter the total amount to be paid into the POS machine.

Thirdly, the request to make a transaction is sent by POS machine to the bank service, the bank will check the customer's account balance:

- If the balance is insufficient, the transaction fails.
- Else, the customer has enough money in his/her account, the transaction is successful and the amount of money will be transferred from the customer's account to the restaurant's bank account.

3.2.3 Sequence diagram for Credit card processing use-case

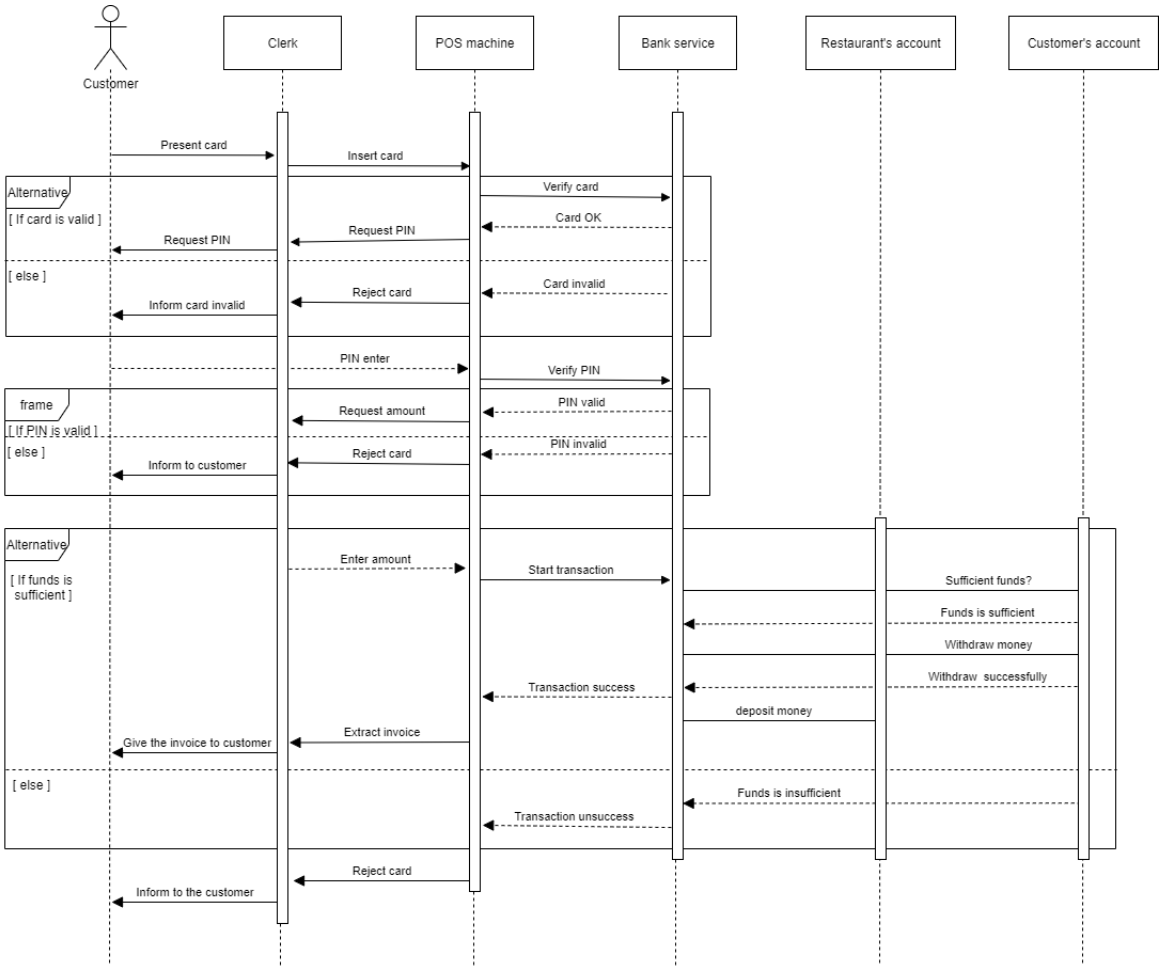


Figure. Sequence diagram for Credit card processing

This is the link for fully view purpose: Click me

3.3 Task 2.3

3.3.1 Requirement

Question: Draw a class diagram

3.3.2 Class diagram

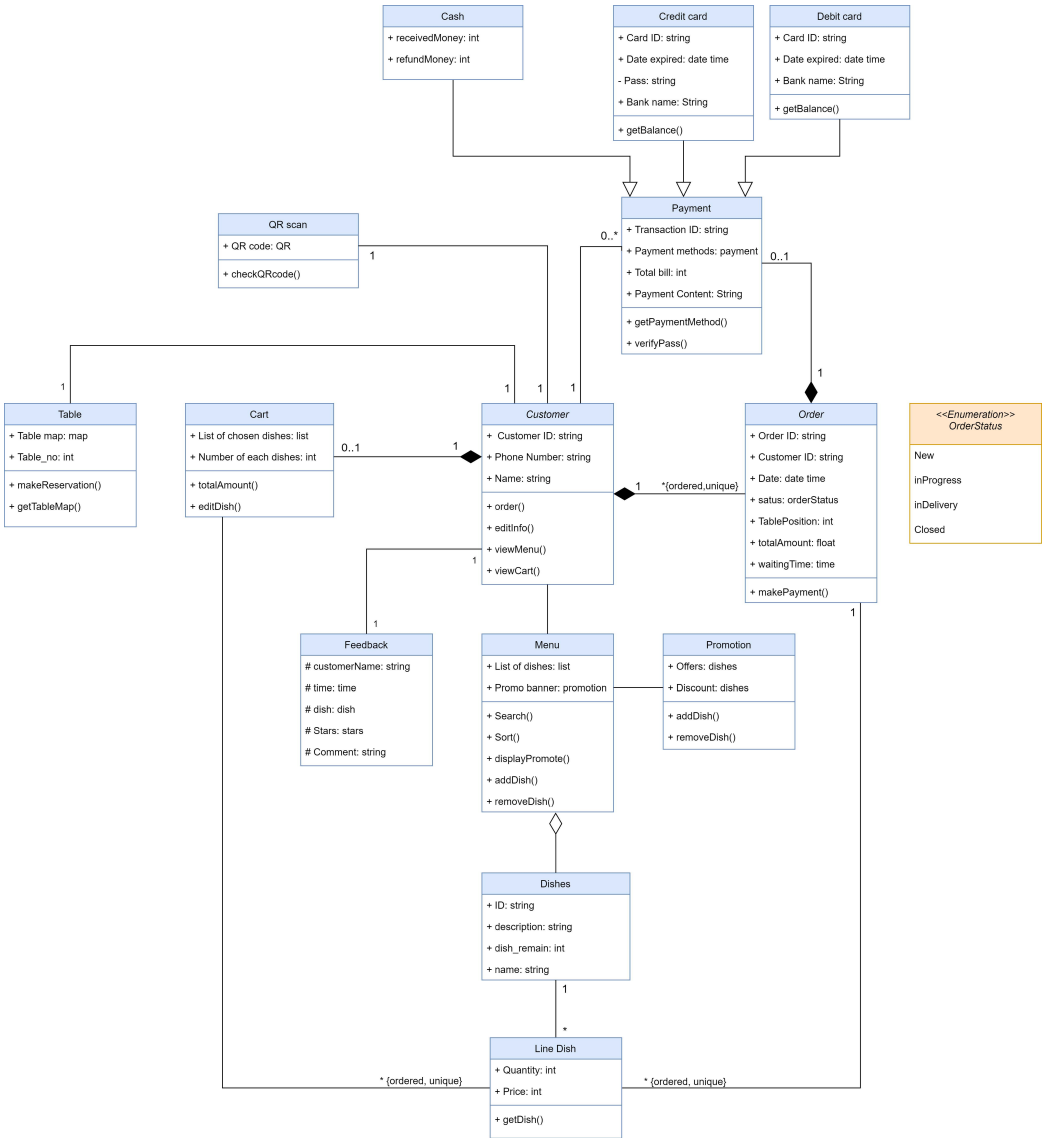


Figure. Customer class diagram

This is the link for fully view purpose: Click me

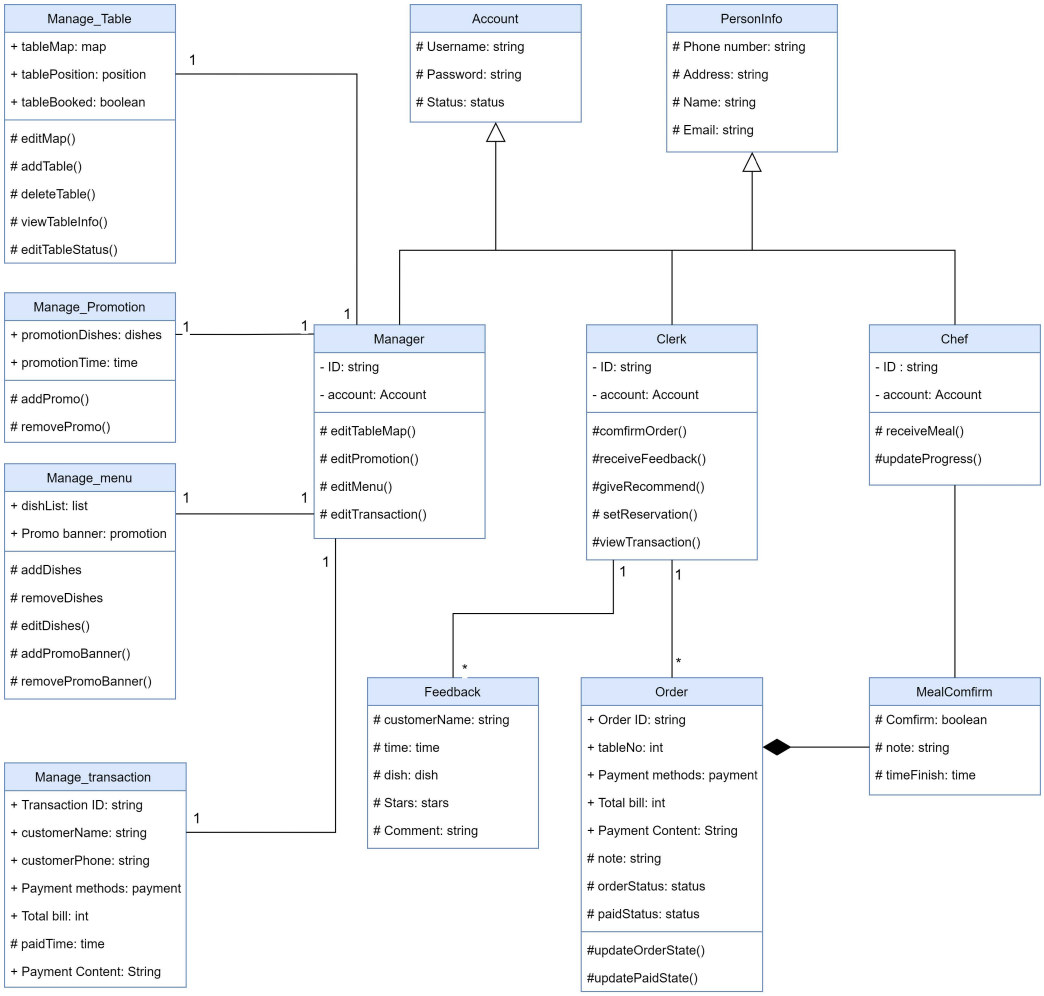


Figure. Manager class diagram

This is the link for fully view purpose: Click me

4 Task 3: Architecture Design

4.1 Task 3.1

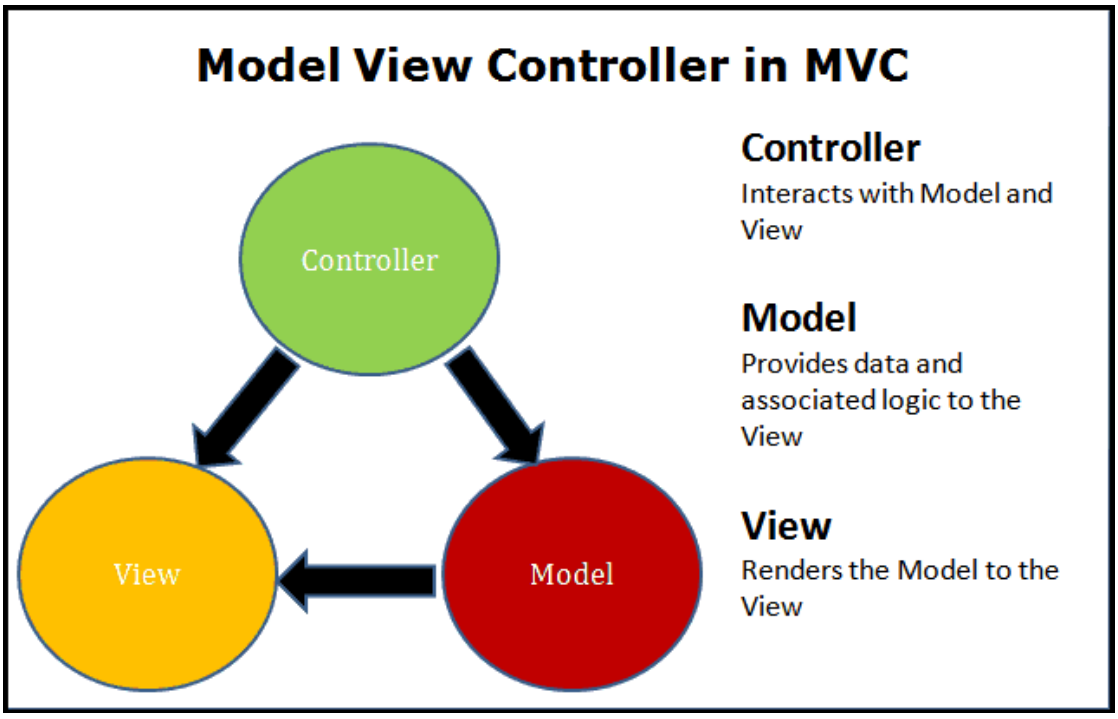
4.1.1 Requirement

Task 3.1. Describe an architectural approach you will use to implement the desired system.

4.1.2 System Overview

The system to be implemented in this assignment is the Point of sale (POS) or point of purchase system. The requirements for the system involve a lot of information from the restaurant and the customers. Customers have to view the data about foods for the food ordering step and select them, and the restaurant has to update the availability of the foods... There will be view and interaction with the data for foods, payments, tables...

Therefore, this results in the choice for an appropriate architecture approach to best describe the system for the implementation: The Model-View-Controller (PVC) pattern.



The Model-View-Controller (MVC) pattern overview

MVC is known as an architectural pattern, which embodies 3 parts: Model, View, Controller. Each of these components are built to handle specific development aspects of an application. It was used for desktop graphical user interface but nowadays is used in designing mobile apps and web apps.

Model
The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, the Customer object will retrieve the customer information from the database, manipulate it and update its data back to the database or use it to render data.

View
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

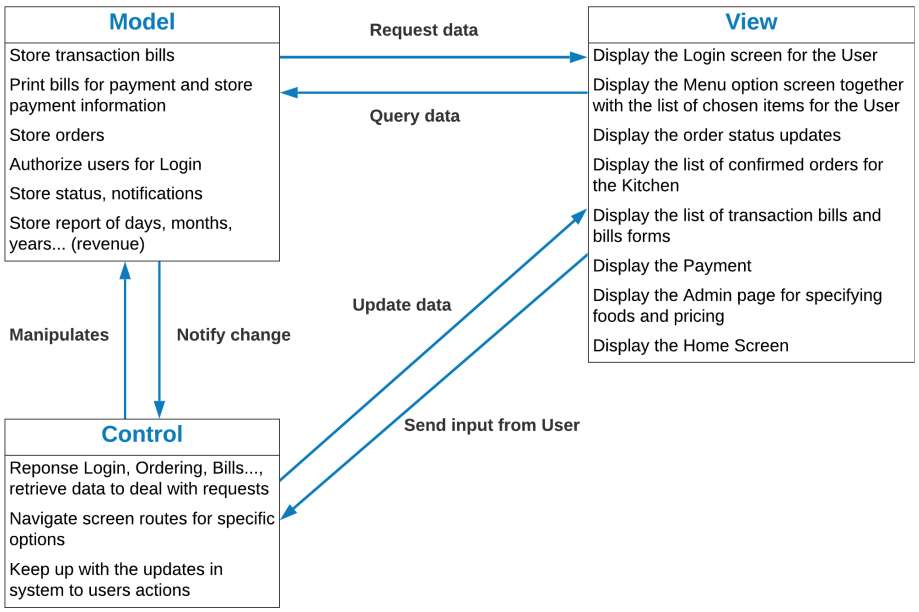
Controller
Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

4.1.3 Advantages of MVC

- MVC architecture will separate the user interface from business logic and business logic.
- Components are reusable.
- Easy to maintain.
- Different components of the application in MVC can be independently deployed and maintained.
- This architecture helps to test components independently.

4.1.4 Disadvantages of MVC

- The complexity is high.
- Not suitable for small applications.
- The inefficiency of data access in view.



MVC Architecture Pattern for the system

4.2 Task 3.2

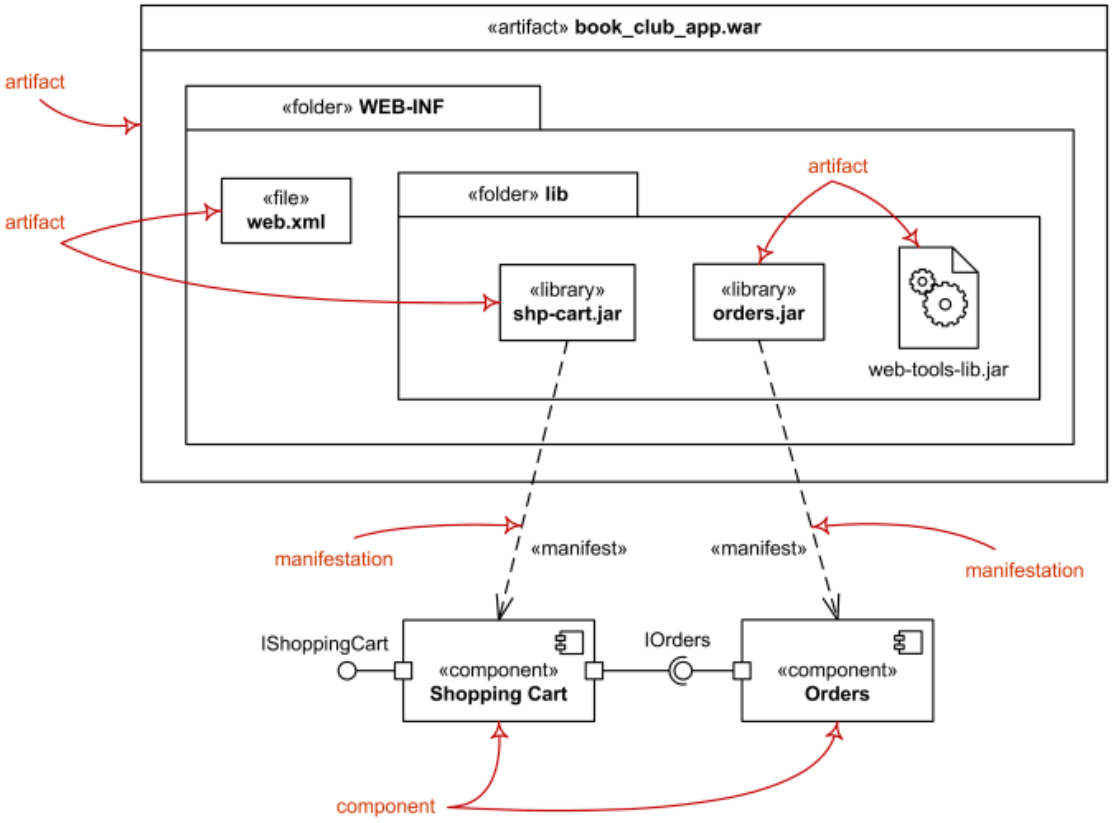
4.2.1 Requirement

Task 3.2. Draw an implementation diagram for Major (not all) functional requirements.

4.2.2 Overview

- **Deployment diagram** is a structural diagram that depicts the system's architecture as the deployment (distribution) of software artifacts to deployment targets.
- **Artifacts** represent concrete elements in the physical world that are the result of a development process. Examples of artifacts are executable files, libraries, archives, database schemas, configuration files, etc.
- **Deployment target** is often represented by a node, which can be either a hardware device or a software execution environment. Nodes might be linked together via communication paths to form networked systems of any complexity.
- In UML 1.x deployment diagrams, components were deployed directly to nodes. Artifacts in UML 2.x are distributed to nodes and can manifest (implement) components. Artifacts are used to deploy components to nodes indirectly.
- Without referring to individual instances of artifacts or nodes, the **specification level deployment diagram** depicts an overview of artifact deployment to deployment targets.
- The **instance level deployment diagram** depicts the deployment of artifact instances to individual instances of deployment targets. It might be used to highlight variations in deployments to development, staging, or production environments using the names/ids of certain build or deployment servers or devices, for example.

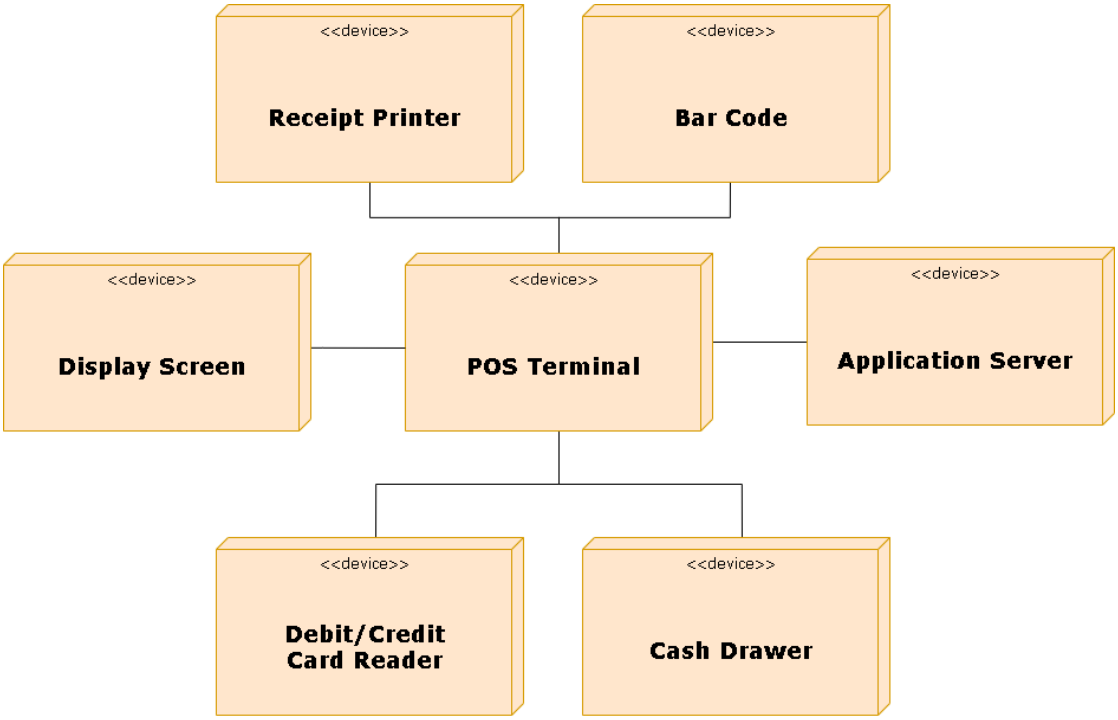
4.2.3 Manifestation of components by artifacts



Manifestation of components by artifacts

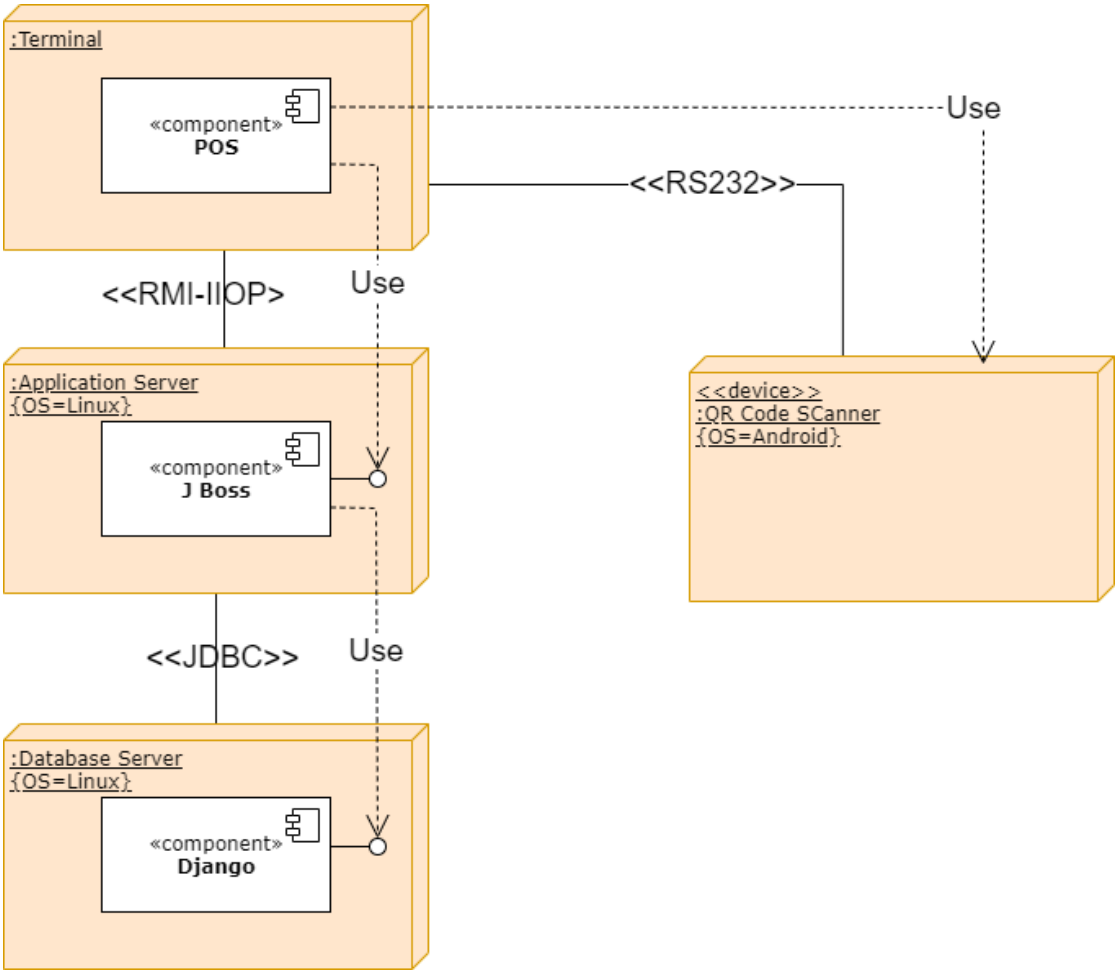
4.2.4 Deployment diagram and deployment view for POS system

Deployment diagram for POS system Figure 2 illustrates the deployment of the POS system using a UML deployment diagram (execute environment is ignored). There are 7 devices included in the whole system, in which POS Terminal is the center device.



Deployment diagram for POS system

Figure 3 shows the deployment of the POS system using a UML deployment diagram.The deployment is a typical 3-tier deployment in which presentation is run on a client, domain code is run on a J2EE application server, and data is stored on a database server(we just show some major functional requirement)



Deployement view of the POS system

The following aspects are of particular interest:

- Environmental elements (represented by UML nodes)
  - The QR code Scanner is the device used to enter sold items into the system. It is read by the POS Terminals over an RS232 connection.
  - The Terminal is the primary point of contact for POS system users.
  - An Application Server is a server that is dedicated to serving all Terminals on an application level.
  - A Database Server is a machine that provides secondary storage.
- Software parts
  - The POS executable component operates the client side of the POS system, including presentation and device management (viz., the QR code Scanner). It communicates with the ApplicationServer using RMI across the network.
  - JBoss is an open source application server which is used for running the domain-related functionality of the system. It uses the DatabaseServer via JDBC

5 Task 4

5.1 Task 4.1 - Github repository

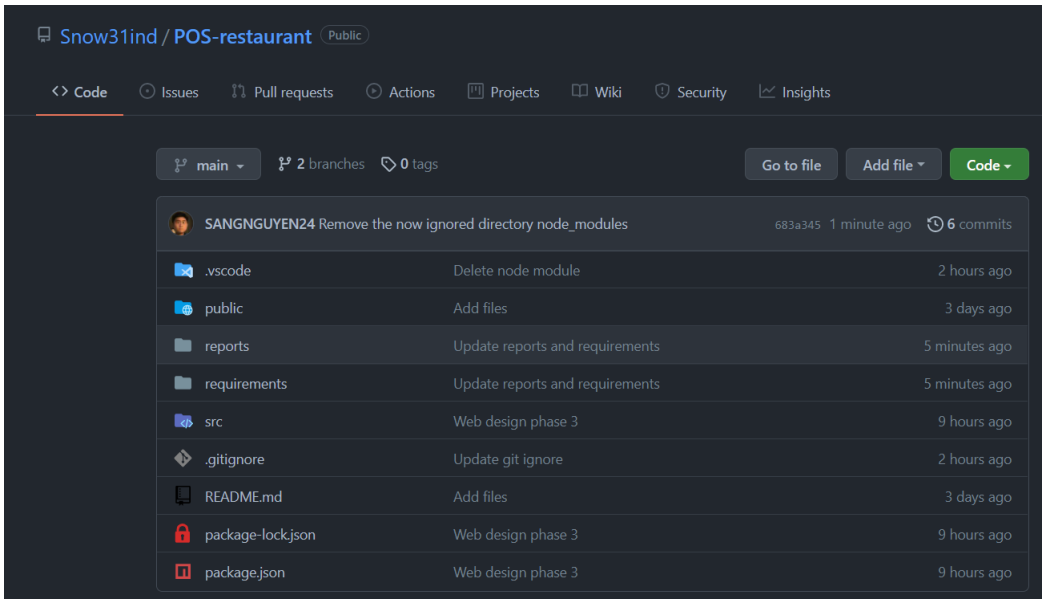
We created a repository on GitHub to contribute code together.

Link to the repo: POS-restaurant

5.2 Task 4.2

Beside the codes, we have 2 additional folders:

- **reports** contains all the reports and changes for documents, materials, System modelling and Architectural design.
- **requirements** contains requirements from the professor.

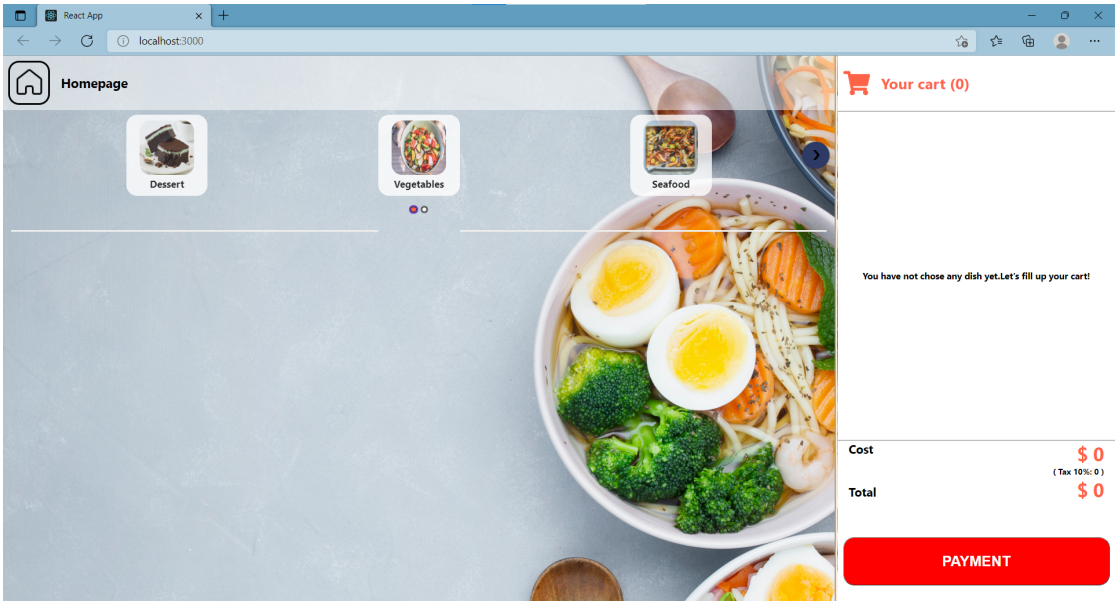


GitHub repository screenshot

5.3 Task 4.3 - POS MVP

5.3.1 User interface

Our group project in this phase is basically a plain easy-to-use menu for users:

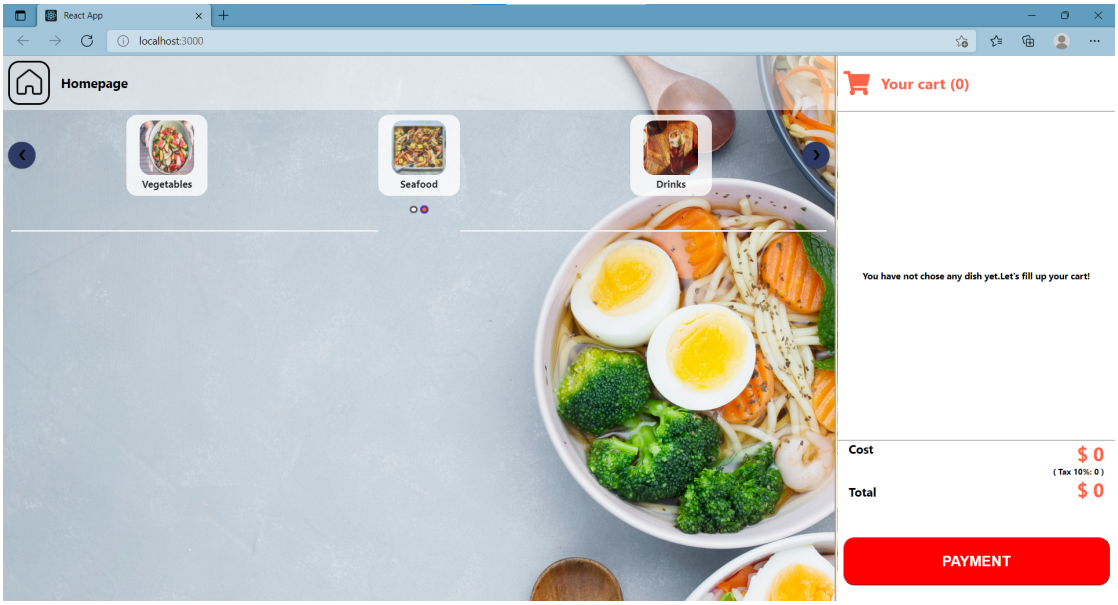


User interface

5.3.2 Features

5.3.2.a Horizontally scrollable menu bar

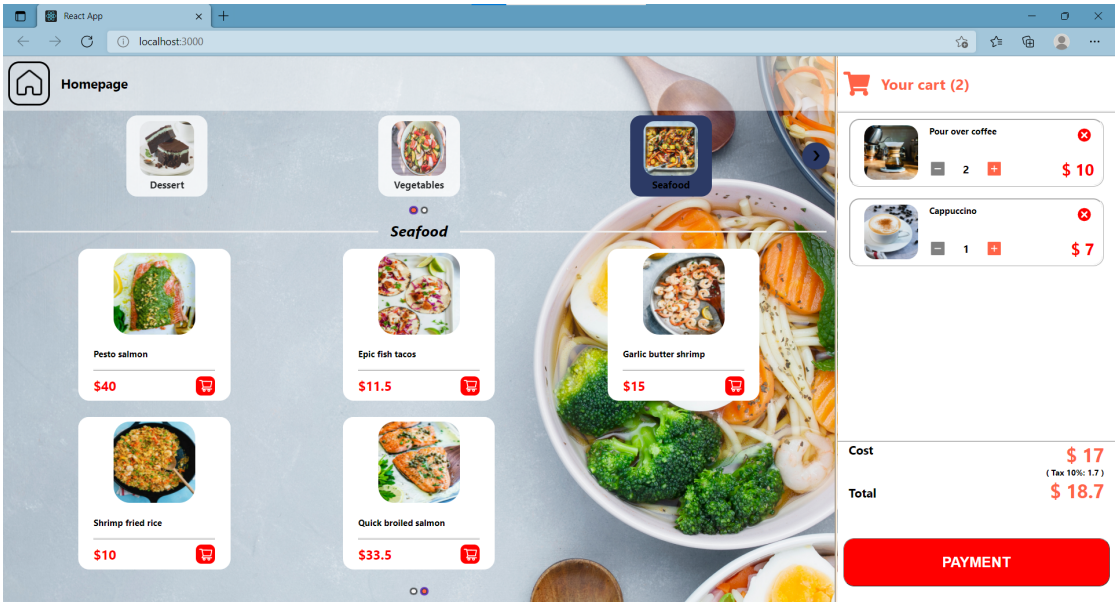
At the first sight, there is a menu that users can **scroll** it **horizontally** to select the menu that they are in favor of.



Menu bar

5.3.2.b Foods filtered by categories

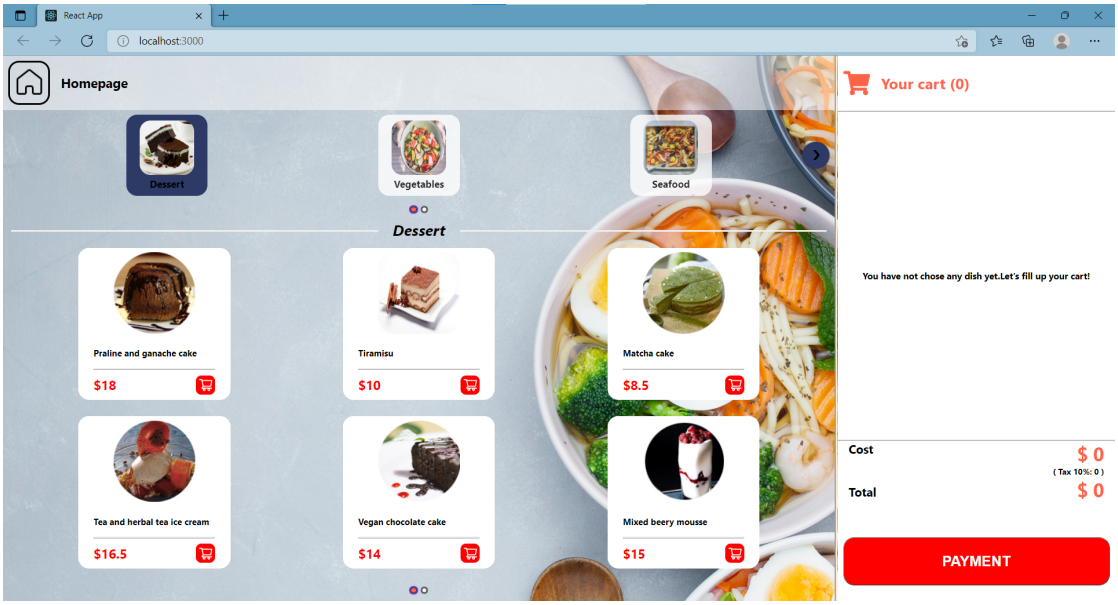
After clicking on a category among display ones in the menu bar, the system will automatically **show** all existing **food of that category** in the food bar.



Filtering menu

5.3.2.c Vertically scrollable food bar

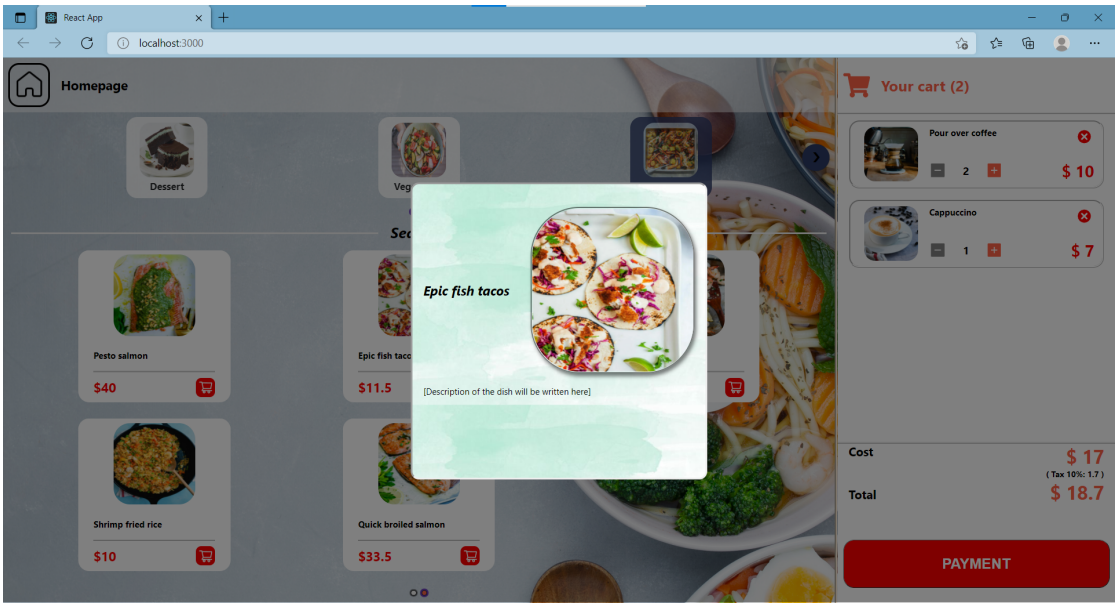
In the food bar, users can also **scroll** the list **vertically** to look up for delicious dishes.



Dish list

5.3.2.d Viewing food information

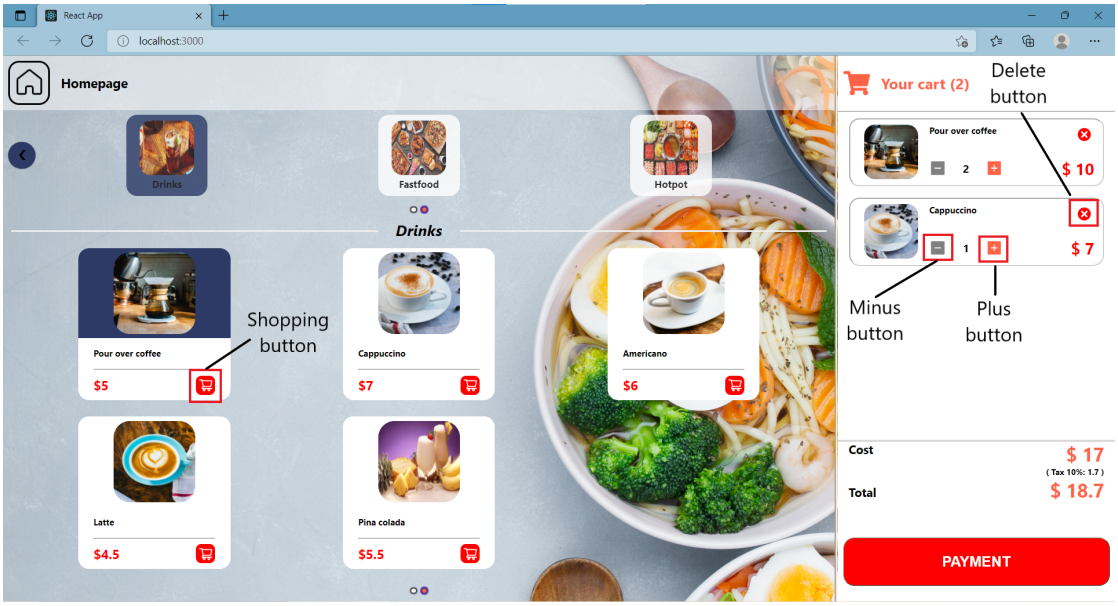
Furthermore, users can also look in **details of a dish**. By click on its image, a **pop-up** containing all detailed information, such as the name, the clearer image, the description, etc, of the dish appears on the screen.



Dish details

5.3.2.e Shopping cart

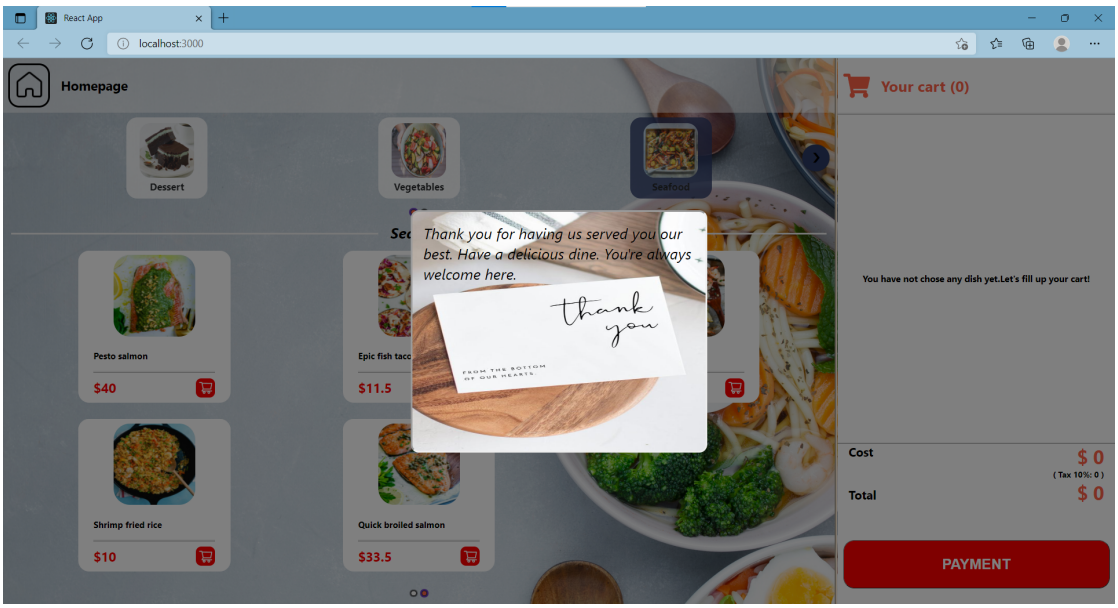
Users can select the food by clicking on the **shopping button** at the lower right corner of the dish. At the end of selecting food, users can **make changes** to their clicked dishes status in the shopping cart. Users can increment, or decrement the number of portions that they want to order by clicking on the **plus button**, or the **minus button**. Also, clicking on the **delete button** help users remove an existing dish from their shopping cart.



Shopping cart

5.3.2.f Payment

Once users have confirmed their shopping cart, the next step is to pay for it. By clicking on the **payment button**, all information about the users' cart will be recorded in a history record and is sent to the back-end system to further process. At this stage, our group implementations is only to show a "thank you" to the customers for having dined in the restaurant.



Payment

5.3.3 Backend

5.3.3.a Introduction to Cloud Firestore

Cloud Firestore is a flexible, horizontally scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps our data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore of Firebase also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

5.3.3.b How Cloud Firestore work?

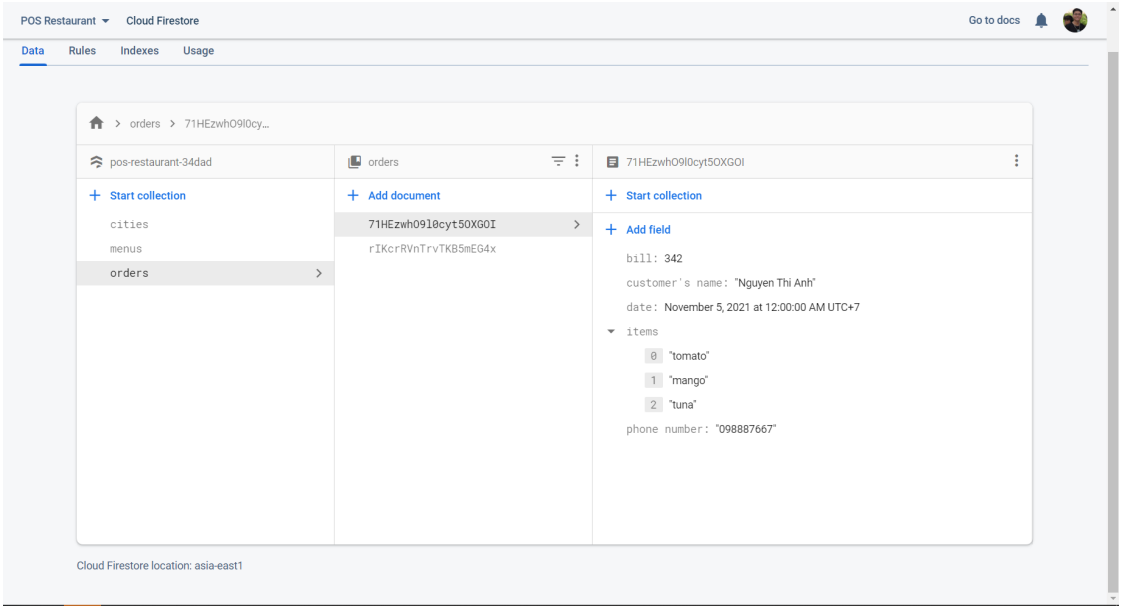
Cloud Firestore is a cloud hosted, NoSQL database that our Apple, Android, and web apps can access directly through native SDKs. Cloud Firestore is also available in native Node.js, Java, Python, Unity, C++ and Go SDKs.

Following Cloud Firestore's NoSQL data model, we store data in documents that contain fields mapping to values. These documents are stored in collections, which are containers for our documents that we can use to organize your data and build queries. Documents support many different data types, from simple strings and numbers, to complex, nested objects. We can also create subcollections within documents and build hierarchical data structures that scale as our database grows. The Cloud Firestore data model supports whatever data structure works best for our app.

5.3.3.c Database implement in Cloud Firestore

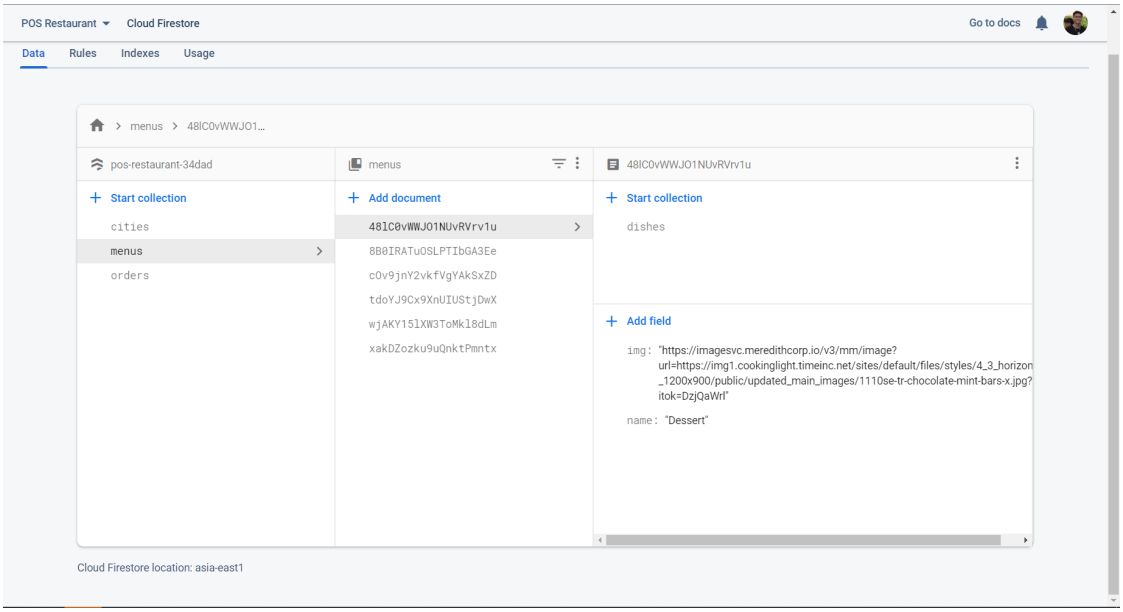
In this website we save all the data in Cloud Firestore as follow:

- **orders** collection contains all invoices each of them has information about bill, created date, items in the order and the customer's phone number.



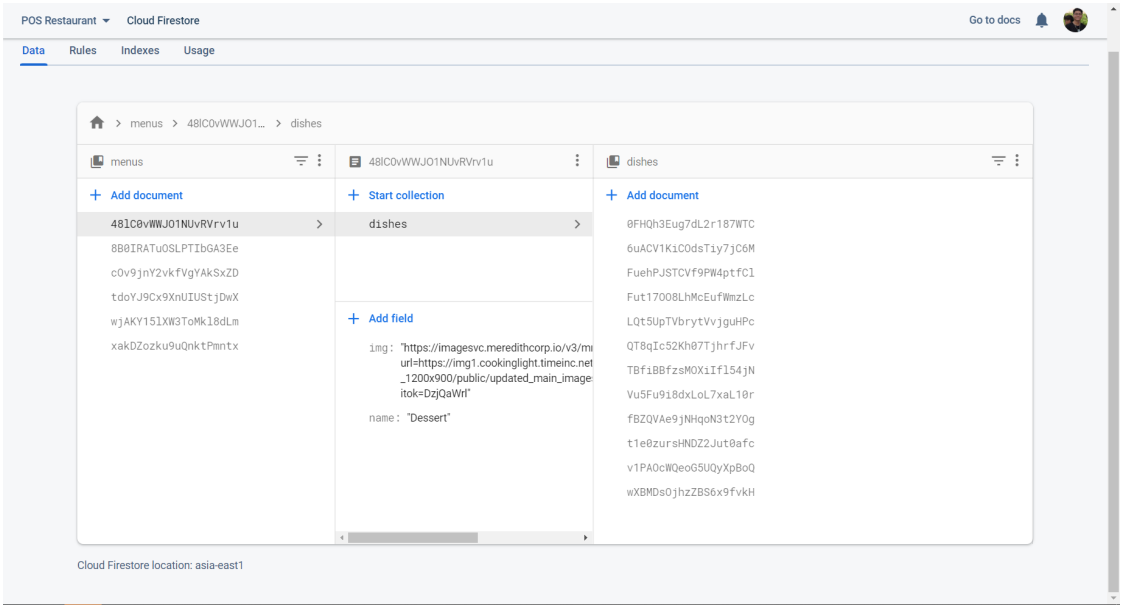
Order collection in Cloud Firestore

- **menus** collection contains all food categories as documents. Each document has category name, image and a subcollection.



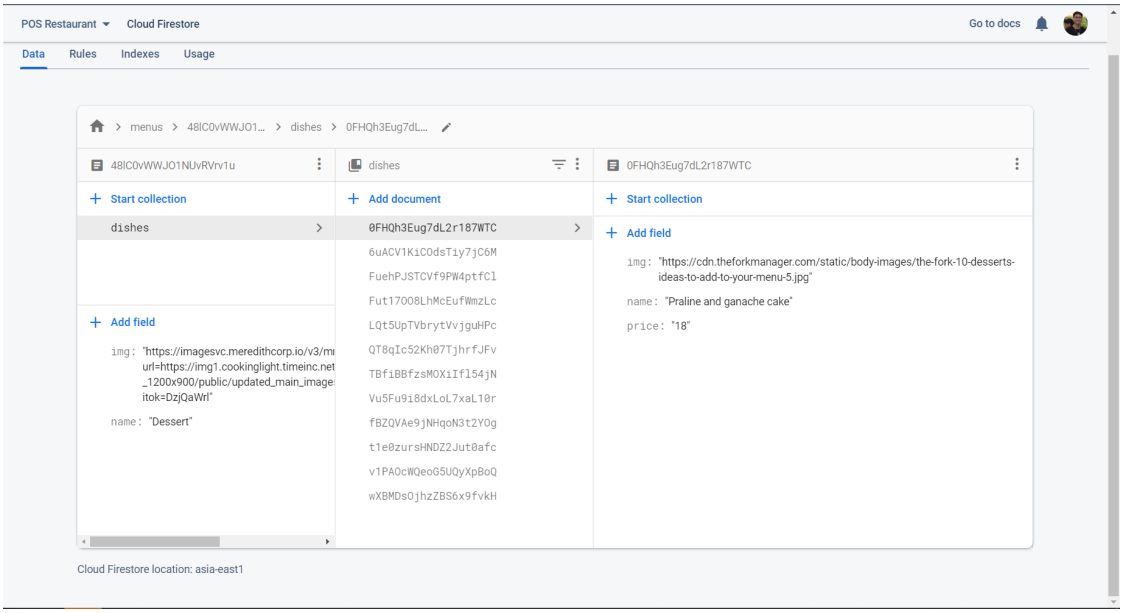
Menu collection in Cloud Firestore

- **dishes** subcollection contains in the selected food category.



Dishes - subcollection in menu collection

- Each document has necessary food information like food image, name of the food and price.



Documents in dishes contain food

6 Task 5

6.1 Website testing

The website is still limited, so we will focus on some main functionalities and non-functionalities for testing.

Functional					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
QR scan	The user can log in to the system by scanning QR codes	100%	0%	High	
select menu	The user can choose from a menu of options.	100%	0%	High	
modify menu	The user can modify their existing order by adding additional items.	100%	0%	High	
remove menu	The user has the option of removing individual items or all items from their current order.	100%	0%	High	
check order	The user can check the status of their current order.	100%	0%	High	
place order	The user can place an order.	100%	0%	High	
view payment	The user can view payment information	100%	0%	High	

Non-functional					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
no contact	The system should allow non-direct contact between Clerks and Customers.	100%	0%	High	
extendable	The system should be extendable to use in multiple restaurants in the future.	100%	0%	High	
time response	The website should respond in less than 0.5 second.	100%	0%	High	
stable operation	The system should operate normally from 7am to 9pm, 7days/week.	100%	0%	High	
transactions number	The number of current transactions is about 300 orders per day.	N/A	N/A	High	Can not test
maintenance fee	The maintenance fee for the system should be less than $300 a month	N/A	N/A	High	Can not test
view payment	The user can view payment information	100%	0%	High	

Forms					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
Scripting check	Scripting checks on the form are working as expected.	80%	20%	High	Non-working email can still sign up
Forms database	The data in the forms is submitted to a live database or is linked to a working email address	80%	20%	High	Non-working email can still sign up
Form readability	Forms are optimally formatted for better readability	100%	0%	High	
stable operation	The system should operate normally from 7am to 9pm, 7days/week.	100%	0%	High	
transactions number	The number of current transactions is about 300 orders per day.	N/A	N/A	High	Can not test
maintenance fee	The maintenance fee for the system should be less than $300 a month	N/A	N/A	High	Can not test
view payment	The user can view payment information	100%	0%	High	

HTML and CSS					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
Syntax	Checking for Syntax Errors	100%	0%	Low	
Readable	Readable Color Schemas	100%	0%	Low	
Form readability	Forms are optimally formatted for better readability	100%	0%	High	
stable operation	The system should operate normally from 7am to 9pm, 7days/week.	100%	0%	High	
transactions number	The number of current transactions is about 300 orders per day.	N/A	N/A	High	Can not test
maintenance fee	The maintenance fee for the system should be less than $300 a month	N/A	N/A	High	Can not test
view payment	The user can view payment information	100%	0%	High	

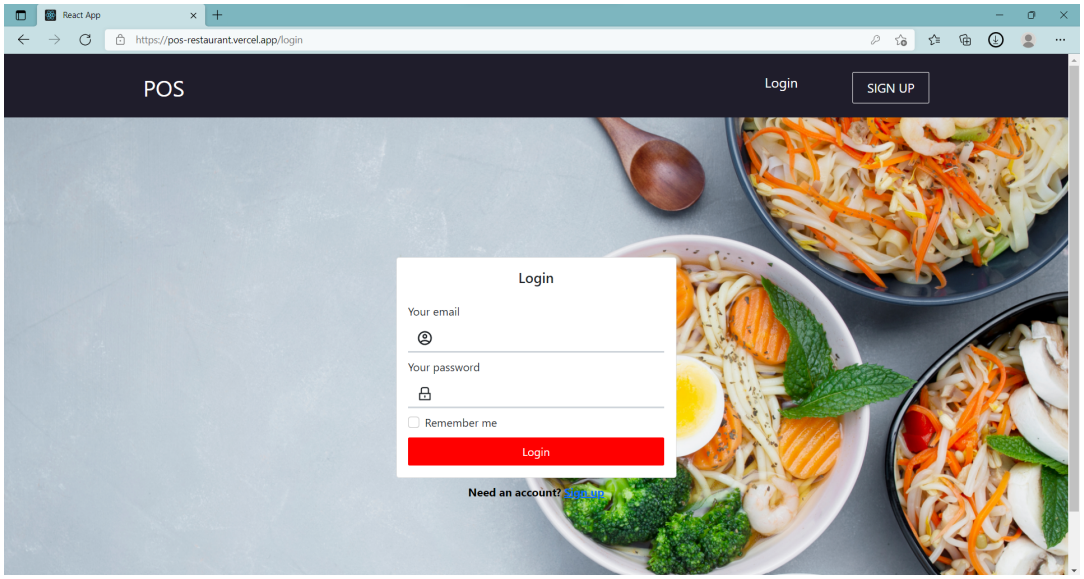
Business workflow					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
workflow	Testing your end – to – end workflow/ business scenarios	100%	0%	High	
negative scenarios	Test when a user executes an unexpected step.	80%	20%	Low	rescaling distort the front-end

Website Usability					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
Website navigation	Menus, buttons should be easily visible and consistent	80%	20%	High	Sign out is not visible
Content spelling	Content should be legible with no spelling or grammatical errors.	100%	0%	Low	
Content testing	Images if present should contain an “alt” text	100%	0%	Low	

Crowd Testing					
Functions	Description	TC% Passed	TC% pending	Priority	Remarks
Crowd Testing	A large number of people (crowd) to execute tests. In this case, we test with 10 clients	100%	0%	High	We test with 15 client

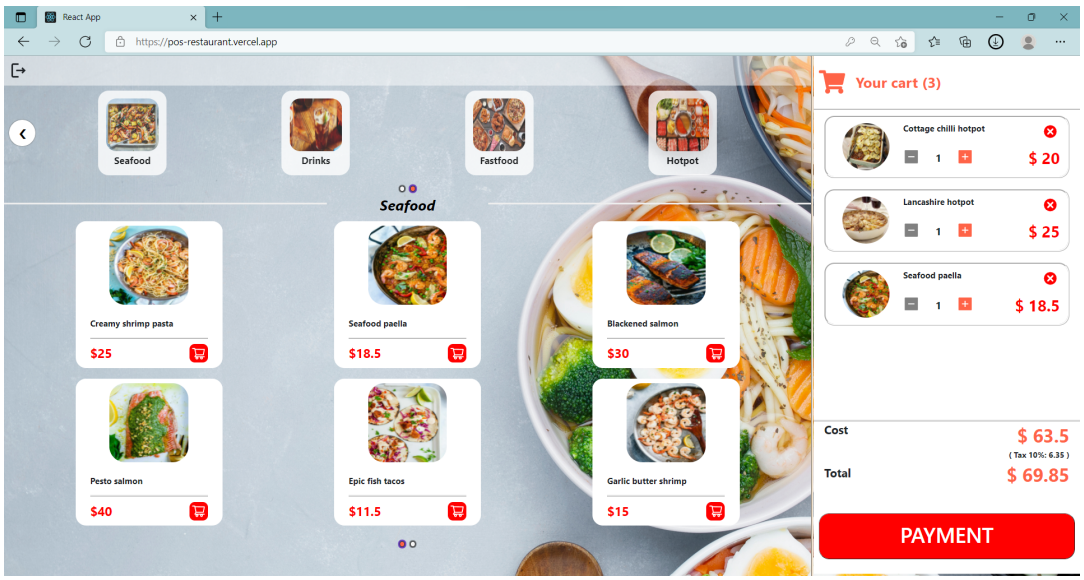
6.2 Improvements over task 4

This is the real deployed link of our project: https://pos-restaurant.vercel.app/
Our project has come to the last step to deploy it to online website. By the task 5, the new pages have been added. First, the login page show that before dishes selection in the next step, customers have to login with their email and password, the remember check box also there as the customers may need to login faster when revisit the restaurant.



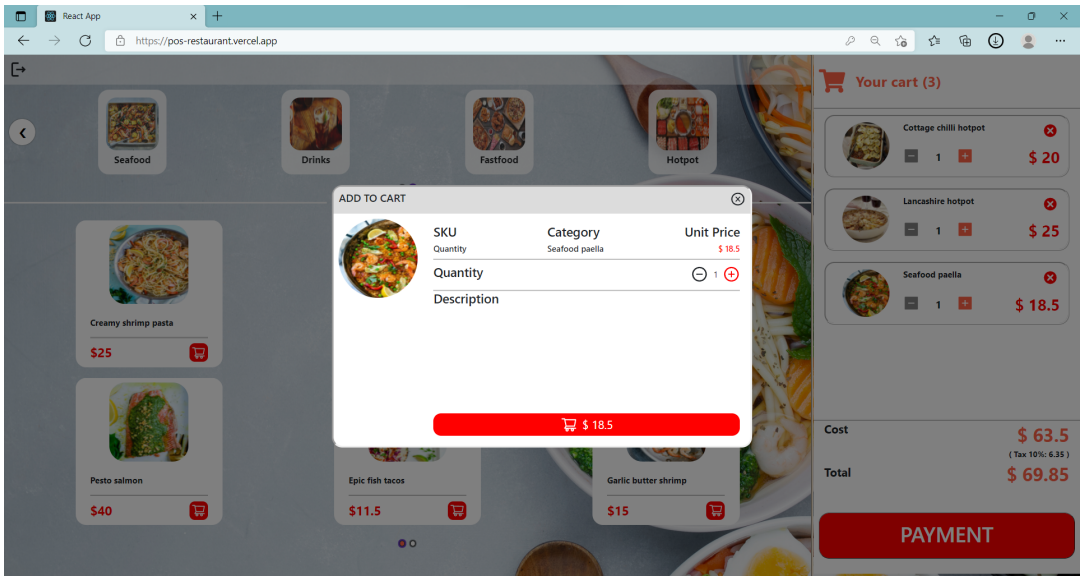
Login page

Our UI for dishes selection page have been improve since the task 4 deployment, with log out button on the top, adding more foods, re-scale some font sizes... This UI is good to go.



Dishes selection page

The dishes information page has been change from task 4 to satisfy the requirements, furthermore, link with database to update the pictures and descriptions.



Dishes information page

7 Conclusion

7.1 Final repository

All our works, reports are uploaded on the Github repository: POS-restaurant.

7.2 Conclusion

We have manage to do:

- For our web front-end: The user interface for the retailer website using React framework is almost done.
  - For the QR: we fully implement the scan QR to log in the restaurant web application.
  - Friendly-user items menu.
  - Friendly-manager order and transaction management.
- For the web back-end: Fast response, fast query, long term data storage.

At the end after we overcome our difficulty we have learned:

- How to teamwork.
- Time management skill.
- Make a professional meeting with group members.
- Making decisions about how a system should work.

- Logical thinking about the end-users experience.
- Cooperating using Github system.
- Fast-learning some new technologies.

Overall, the project runs as expected. The completion rate is 100%.

References

[1] Khaneja, Deepesh. (2018). *POINT OF SALES TERMINAL (POST) using UML*. 10.13140/RG.2.2.12182.65605. from https://www.researchgate.net/publication/323073643_POINT_OF_SALES_TERMINAL_POST_using_UML

[2] Christensen, Henrik and Corry, Aino and Hansen, Klaus. (2004). *An Approach to Software Architecture Description Using UML*. Retrieved 21:39, August 10, 2021, from https://www.researchgate.net/publication/238077147_An_Approach_to_Software_Architecture_Description_Using_UML

[3] Google Developer. (2021) *Cloud Firestore*. Retrieved November 6, 2021, from https://firebase.google.com/docs/firestore