VIET NAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING

Software Development Assignment

Smart Food Court System

Instructor: Quản Thành Thơ
Bùi Hoài Thắng
Student: Trần Hoàng Long - 1852545
Bùi Quang Lộc - 1852552
Trần Nguyễn - 1852623
Vũ Lê Thiện Minh - 1852590
Trần Vũ Hồng Thiên - 1752506

Ho Chi Minh City, April 2020

Contents

1 Document History

Date	Version	Changes	Changed by
20/4/2020	1.0	- Write report - Describe initial main function (13 functions)	Trần Hoàng Long
21/4/2020	1.1	- Work assgignment - Initial use-case demo	Trần Hoàng Long
27/4/2020	2.0	- Generalize functions (7 main functions) - Added general and comprehensive use-case diagram - Added task assignment and detaled feature description - Minor text changes	Trần Hoàng Long
27/4/2020	2.1	- Added detailed use-case diagram, user memory, main flow chart and mock-up for Manipulate Data feature - Minor changes in Manipulate Data's use-case scenario	Trần Hoàng Long
27/4/2020	2.2	- Added detailed use-case diagram, user memory, main flow chart and mock-up for Search feature - Update use-case detail for above feature	Vũ Lê Thiện Minh
27/4/2020	2.3	- Added detailed use-case diagram, user memory, main flow chart and mock-up for Place Order feature - Update use-case detail for above feature	Bùi Quang Lộc
27/4/2020	2.4	- Added detailed use-case diagram, user memory, main flow chart and mock-up for Manage Order and Payment feature - Update use-case detail for above feature	Trần Vũ Hồng Thiên
27/4/2020	2.5	- Added detailed use-case diagram, user memory, main flow chart and mock-up for View Report feature - Update use-case detail for above feature	Trần Nguyễn
9/5/2020	3.0	- Change from flow-chart to activity diagram and added more details for Manipulate Data feature - Added sequence diagram and state diagram for the above feature. - Changed details in the use-case scenario of the above feature	Trần Hoàng Long
24/5/2020	3.1	- Added Class diagram for Search feature -Make adjustment to Search use-case diagram	Vũ Lê Thiện Minh
24/5/2020	4.0	- Added Deployment View of the whole system -Added component diagram of Data Manipulation use-case	Bùi Quang Lộc
13/6/2020	5.0	Manipulate Data Feature: - Added Class diagram - Added method description for class diagram - Change all sequence diagram to more simplistic manner by reducing the GUI classes to a single interface	Trần Hoàng Long
13/6/2020	5.1	Add View list feature: - Added use-case diagram - Add mockup for View list feature	Vũ Lê Thiện Minh
14/6/2020	5.1.1	- Add design pattern description Manipulate Data Feature: - Add module interface into class diagram - Small text edits	Trần Hoàng Long

25/6/2020	5.1.2	- Add Class Diagram for Report Management feature	Trần Nguyễn
28/6/2020	5.2	- Add Class diagram for View list feature - Add method description for class diagram	Vũ Lê Thiện Minh
30/6/2020	5.2.1	Manipulate Data Feature: - Remove module interface cause they're unnecessary - Small tweak in class diagram - Small text edits - Small tweak in sequence diagram - Small tweak in use case	Trần Hoàng Long
2/7/2020	5.3	- Add new use case description - Add Mock ups for all features	Bùi Quang Lộc
4/7/2020	5.3.1	- Add class diagram for "Place order"features	Bùi Quang Lộc
5/7/2020	5.4	Search feature: - Add method description for class diagram View list feature: Add use-case detail	Vũ Lê Thiện Minh
6/7/2020	5.4.1	- Make changes to class diagram of Search features	Vũ Lê Thiện Minh
6/7/2020	5.4.2	- Add flowchart for View list feature - Change use-case detail for View list	Trần Hoàng Long
15/7/2020	5.5	- Add description for methods in the class diagram of the report management feature	Trần Nguyễn
16/7/2020	5.6	- Add sequence and state diagram for view report	Trần Nguyễn
26/7/2020	5.6.1	- Update class diagram for "Place order"features	Bùi Quang Lộc
30/7/2020	5.6.2	- Add sequence diagram for generating report feature and detailed description	Trần Nguyễn

2 Introduction

The university is currently has one food court located in its Ly Thuong Kiet campus and is going to build another one in Di An campus. All food courts consist of a number of vendors at food vendors or service counters. Meals are ordered at one of the vendors and then carried to a common area for consumption. The food may also be ordered to take-away. All food are self serviced and there's no food delivering system (maybe in the future).
In 2020, the university wish to build a smart food court system (SFCS) to make the university more smart. The system is for customers to order foods from the tip of their fingers - their mobile device.
The mobile app allows users to order food anywhere, before they come to the food court. Notification will be pushed to the app to notify order ready to be picked up.
Payment can be made on-site or through other online methods like Pay-pal, Momo,...

Of course, the application have other features and interfaces for the managers and staffs of the store:

- For employees: Order queueing, stock management,..
- For vendor: Menu editor, employee management, discount plans,..

- For foodcourt manager: Authentication, Overall management and statistics,..

We'll go in detail about the features in this document.

3 Functional Requirement

3.1 Features (Functions)

1. View list: After login customer can see a list of available vendors in the Food Court. The customer can also click on a vendor in the list to see the menu of that vendor.
2. Searching: User can search for items in their current working tag, like searching food for customer, staffs for manager,... User can also choose a filter category to narrow down the search result.
3. Place order: while browsing through dishes gallery, customer can pick items to add to their shopping cart. They also can choose item quantity and remove items from cart,... before proceeding to payment.
4. Payment: After confirming their order choice, customer can choose their payment method, either by online payment through third-parties like Momo, Paypal,.. or directly pay at the vendor cashier. (In our project, online payment is not implemented).
5. Manage orders: The vendor staff can view a dynamic queue of waiting order, notify the customer when the order is ready and finish the order when payment is finished.
6. User profile: Every user must access the system through an authentication system, the account types include: Customer, staff, technician, vendor owner and food court manager. Each actor can also view and edit their profile. (In our project, profile page is not implemented)
7. Manipulate data: Include the methods for managing and manipulate data base for each actor. Such as view and modify vendors, vendor menu, staffs,...
8. View Report: Food court manager and vendor manager can view reports and statistics for their respective business.
9. Generate Report: Generate reports for vendors and food court. Depend on types of reports, the ways to generate them are different.
10. Maintenance: Technician can switch the system between two states: business (server online) and maintenance (offline) in order for maintenance, updates and bug fix.

3.2 Non-interactive functions

1. The data for each order of each food court is automatically stored in data base for analysis.
2. Every operating day, at a specific pre-defined time, report of statistics of stalls and the whole food court is auto-generated for future view.
3. When system is put into maintenance, all processes are saved for for when the system comes back online.

3.3 Use-case Diagram

3.3.1 General Use-case Diagram

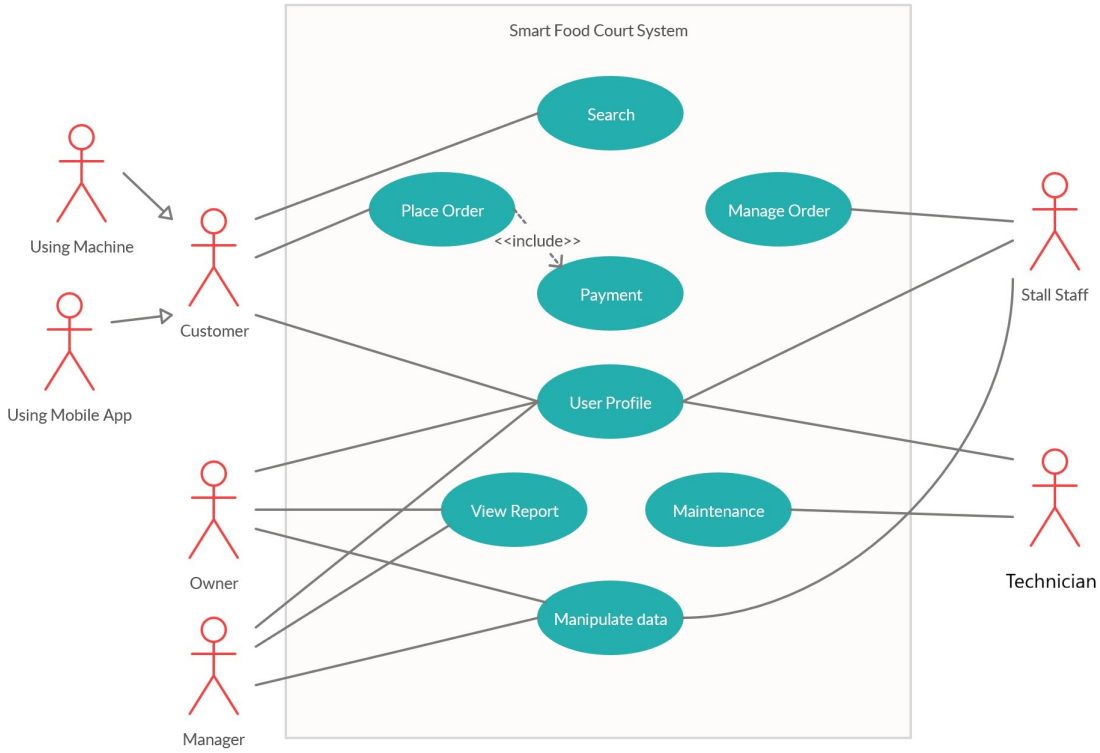


Figure 1: Smart food court system simple use-case diagram

4 Non-functional requirement

A. Ease of use

- Training time for users like vendor owner and staffs must be short, within 2-3 hours for all features
- Simplistic and clear layout, fonts.
- Design should help use easy to understand the function of each modules.

B. Accessibility

- App is operational on Android and Ios mobile platform

C. Stability

- App server must be online 24/7
- Downtime for server (error) must be under 30 minutes. Except for maintenance time.

D. Speed

- App must be responsive to user's choices
- App should register user's order to the system and confirm order to the customer in under 20 seconds
- The system should complete the billing process within 20 seconds
- Updates that affects the whole system network must be sent to database and synchronized between all interfaces with in 1 minutes

E. Usability

- App is available for users in the university domain

F. Security

- The system should validate the user username and password in order to let user log in and make change to the system.
- System should request current password of the user in order to let them to a new one.
- User's information (log-in, profile, payment,..) must be secured and used for in app purposes only.

G. Data integrity

- All records of order and payment must be store as unique entity with a unique ID.
- The system must check for duplicated data be when user add a new record that should be unique (log-in username for example)

H. Portability

- The app should have portable size (under 30Mb) in order for easy user install and future OS migration.

5 Task Assignment

ID	Function Name	Assigned to	Student ID
1	View list and Search	Vũ Lê Thiện Minh	1852590
2	Place Order	Bùi Quang Lộc	1852552
3	Manipulate Data	Trần Hoàng Long	1852545
4	Manage Order	Trần Vũ Hồng Thiên	1752506
5	Report Management	Trần Nguyễn	1852623

Below is a more detailed feature list and their ID:

1. View list: Customers can view a list of all the vendors available in the Food Court, then they can choose a vendor to view the menu of that vendor

2. Search: Customers can search for a specific dish and they can use a filter to narrow down the search results
3. Place Order:
   A. Cart Managing: This feature includes actions such as adding dishes to cart, managing their quantity as well as remove them from the cart. This feature also provide the GUI of cart.
   B. Place Order: From the cart view, customer can place his order. Doing this will make the system send the order to database server so that the vendor's staff can get it. It also informs customer if their cart is empty. Customer can order from many vendors at the same time too.
   C. Order tracking: After placing an order successfully, the view in cart will change into in which customer can track if his order from each vendor is ready to served or not.
4. Data manipulation (Business Management features):
   A. Modify menu for vendor's owner: The owner of the vendor can add/remove dishes to/from the menu showed in the app, change the description, the price or replace the photos of a dish. He/She can also apply a temporarily price (discount) to a dish.
   B. Modify staff list for managers: Owners can create staff account for their employees as well as manage a list of all staff accounts from his/her domain (add, remove, edit profile).
   C. Modify list of vendors for food court manager: The food court manager is responsible for adjusting the vendor list when there are any new vendors or ones that leave the food court. The list is displayed in the app. He/she can also change their profiles info.
5. Manage order features:
   A. Show order queue for vendors' staff: Accept order of the user and put on the queue waiting to be served.
   B. Inform customer that an order is ready for serving: Send a notification from the kitchen to user's phone when food is done and will repeat after an amount of time if the user has not come to collect food.
   C. Notify product is out of stock for vendors' staff: Inform the system and mark some food is currently out-of-order. The user can not order a product which is out of stock.
6. Viewing report
   A. Vendor Report:
      - Daily Report: show the statistics of orders that were sold and total daily sale on that day.
      - Monthly Report: show the statistics of all the opening days in that month.
   B. Food court Report:
      - Monthly Report: show the statistics of all vendors in that month, along with the total paid these vendors have to pay and total proceed of that month.
7. Generating report
   A. Vendor Report:

- Daily Report: Created when the first staff logging in the app and updated when the staffs finish orders from customer
- Monthly Report: Auto-generated at the end of the month in the background of the device

B. Food court Report:

- Monthly Report: The same as Vendor monthly report

6 Deployment View

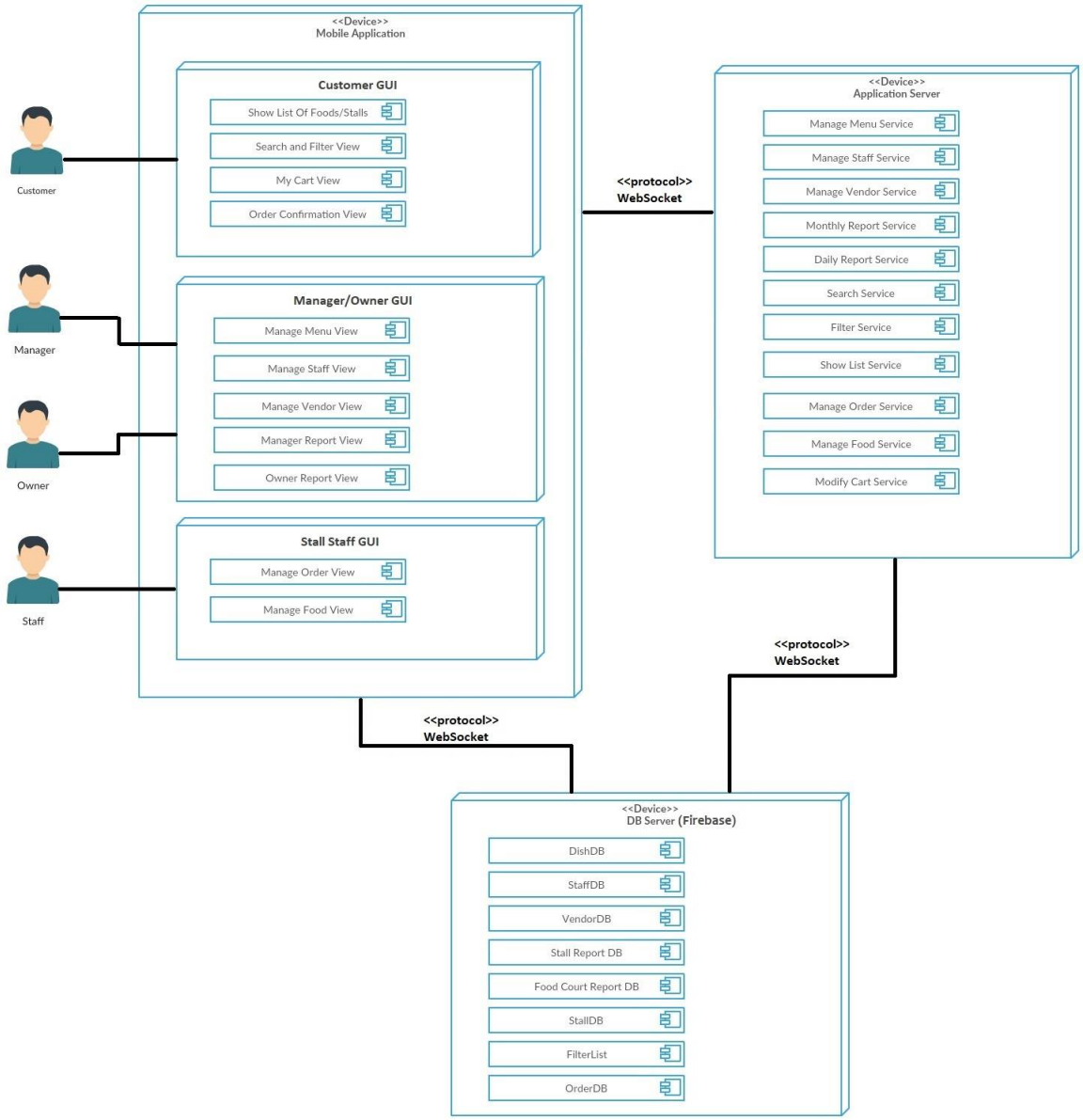


Figure 2: Smart food court system comprehensive use-case diagram

7 Design Pattern

- For implementing the database and local data. I will use Observer pattern so that the database (would be implemented with FireBase) will send notification to the app, and the view classes observing the DB when there's change to our data. Then the views will pull

and "reprocess" the local data for display.

- As for the design of our application, the MVC pattern is used to separate View(UI), Service(Data Helper) and Model(Local data and DB).
- The facade pattern is use as a wrapper for our app, it's easily seen that our app will react and provide different interface/functionality for each type of user. This is used in the logging in and user feature specification.
- The iterator pattern is used to iterate and display our different types of data (Dishes, Vendors, Staffs) without caring about the actual type of data that need to be process
- The singleton pattern can be use to initialize "constant object" such as log-in information, configuration settings,..

8 Technology Stack

To implement the system, we develop an mobile app for Android which will provide features based on the role of users' account. **Flutter** and **Firebase** are mainly used.

9 Use-case description

9.1 Manipulate Data

Features assigned: Business Management

5. Data manipulation (Business Management features):
   A. Modify menu for vendor's owner: The owner of the vendor can add/remove dishes to/from the menu showed in the app, change the description, the price or replace the photos of a dish. He/She can also apply a temporarily price (discount) to a dish.
   B. Modify staff list for managers (FC Manager and Vendor Manager: Owners can create staff account for their employees as well as manage a list of all staff accounts from his/her domain (add, remove, edit profile).
   C. Modify list of stalls for food court manager: The food court manager is responsible for adjusting the stall list when there are any new stalls or ones that leave the food court. The list is displayed in the app. He/she can also change their profiles info.

9.1.1 Use-case diagram

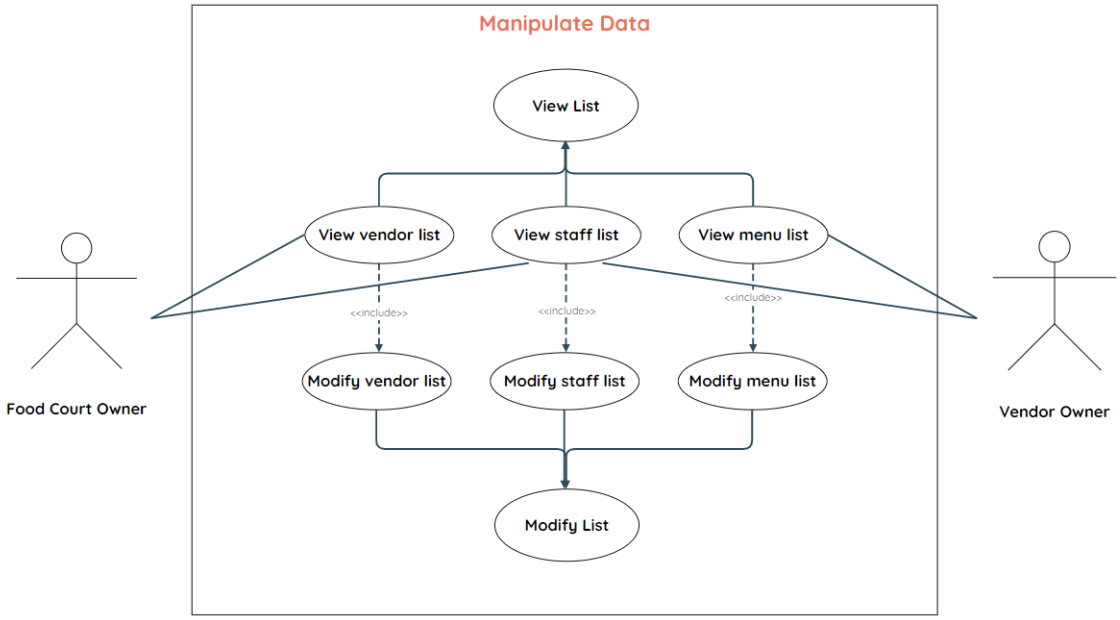


Figure 3: Use case diagram for manipulate data (manage business)

9.1.2 Class Diagram

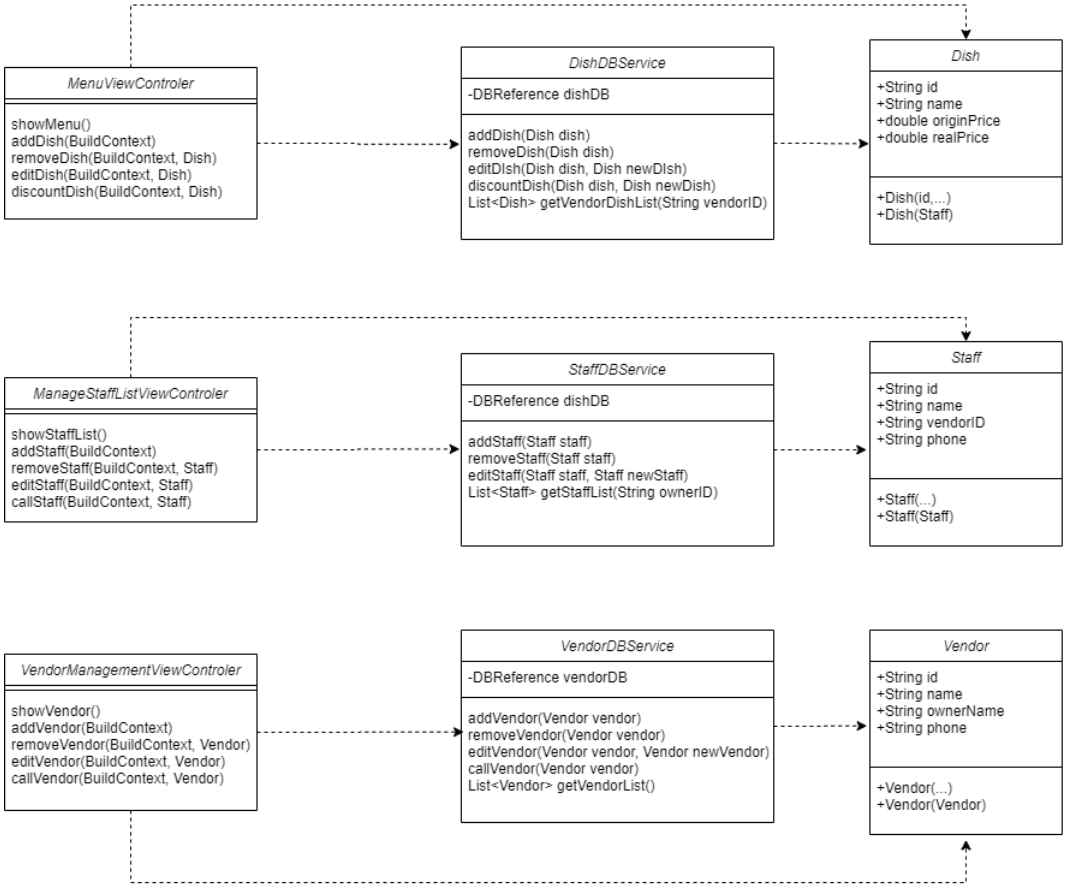


Figure 4: Class diagram for the Business Management Feature

9.1.3 Method description

- Class: DishDBService

Method Name	Input	Output	Description
addDish	new dish	none	- Add a new dish to DishDB - Request view to reload
removeDish	dish to remove	none	- Remove dish from DishDB - Request view to reload
editDish	dish to edit new dish	none	-Get the current dish's reference on DishDB -Change the needed fields (name,originPrice) -Reset discount status of dish -Request view to reload
discountDish	dish to discount new dish	none	-Get the current dish's reference on DishDB -Change the needed fields (realPrice, discountPercentage) -Request view to reload
getVendorDishList	vendorID	Dish list of vendor	- Get all dish of that vendor ID from DishDB - Return as a List<Dish>

- Class: StaffDBService

Method Name	Input	Output	Description
addStaff	new staff	none	- Add a new staff to StaffDB - Request view to reload
removeStaff	staff to remove	none	- Remove staff from StaffDB - Request view to reload
editStaff	staff to edit new staff	none	-Get the current staff's reference on StaffDB -Change the needed fields (name,phone, position) -Request view to reload
getStaffList	vendorID	Staff list of owner	- Get all staff of that owner ID from StaffDB - Return as a List<Staff>

- Class: VendorDBService

Method Name	Input	Output	Description
addVendor	new vendor	none	- Add a new vendor to VendorDB - Request view to reload
removeVendor	vendor to remove	none	- Remove vendor from VendorDB - Request view to reload
editVendor	vendor to edit new vendor	none	-Get the current vendor's reference on VendorDB -Change the needed fields (name,phone) -Request view to reload
getVendorVendorList	vendorID	Vendor list of vendor	- Get all vendor from VendorDB - Return as a List<Vendor>

- Class: MenuViewControler: This is in charge of controlling views and view logic in a vendor menu.

Method Name	Input	Output	Description
addDish	context	none	- Open a form to take user input for a new dish - Take user confirmation, also check for correct input type - Call service to add a new Dish to DB and refresh
editDish	dish to edit context	none	- Open a form to take user input to edit current dish - Take user confirmation, also check for correct input type - Call service to edit the current dish on DB and refresh
removeDish	dish to remove context	none	- Take user confirmation - Call service to remove current Dish from DB and refresh
discountDish	dish to discount context	none	- Open a form to take user input to edit current dish - Take user confirmation, also check for correct input type - Call service to change dish info on DB and refresh
showMenu	none	none	- Call service to get all dishes of current vendor and show them

- Class: ManageStaffViewController: This is in charge of controlling views and view logic in a staff list of Vendor Manager or FC Manager.

Method Name	Input	Output	Description
addStaff	context	none	- Open a form to take user input for a new staff - Take user confirmation, also check for correct input type - Call service to add a new Staff to DB and refresh
editStaff	staff to edit context	none	- Open a form to take user input to edit current staff - Take user confirmation, also check for correct input type - Call service to edit the current staff on DB and refresh
removeStaff	staff to remove context	none	- Take user confirmation - Call service to remove current staff from DB and refresh
callStaff	staff to call context	none	- Use service to call staff by their phone number
showStaffList	none	none	- Call service to get all staffs of the current user and show them

- Class: ManageVendorViewController: This is in charge of controlling views and view logic in a vendor list of Food Court Manager

Method Name	Input	Output	Description
addVendor	context	none	- Open a form to take user input for a new vendor - Take user confirmation, also check for correct input type - Call service to add a new vendor to DB and refresh
editVendor	vendor to edit context	none	- Open a form to take user input to edit current vendor - Take user confirmation, also check for correct input type - Call service to edit the current vendor on DB and refresh
removeVendor	vendor to remove context	none	- Take user confirmation - Call service to remove current vendor from DB and refresh
callVendor	vendor to call context	none	- Use service to call vendor by their phone number
showVendorList	none	none	- Call service to get all vendors and show them

The detailed scenario of each function use-case is described in detailed below

9.1.4 Modify menu use-case

9.1.4.1 User Story

As a vendor owner, I want to see my vendor's menu listed out and be able to modify them:

- Add new dish to menu
- Remove dish from menu
- Modify profile of some dishes
- Make discount option on some dishes

9.1.4.2 Main-flow

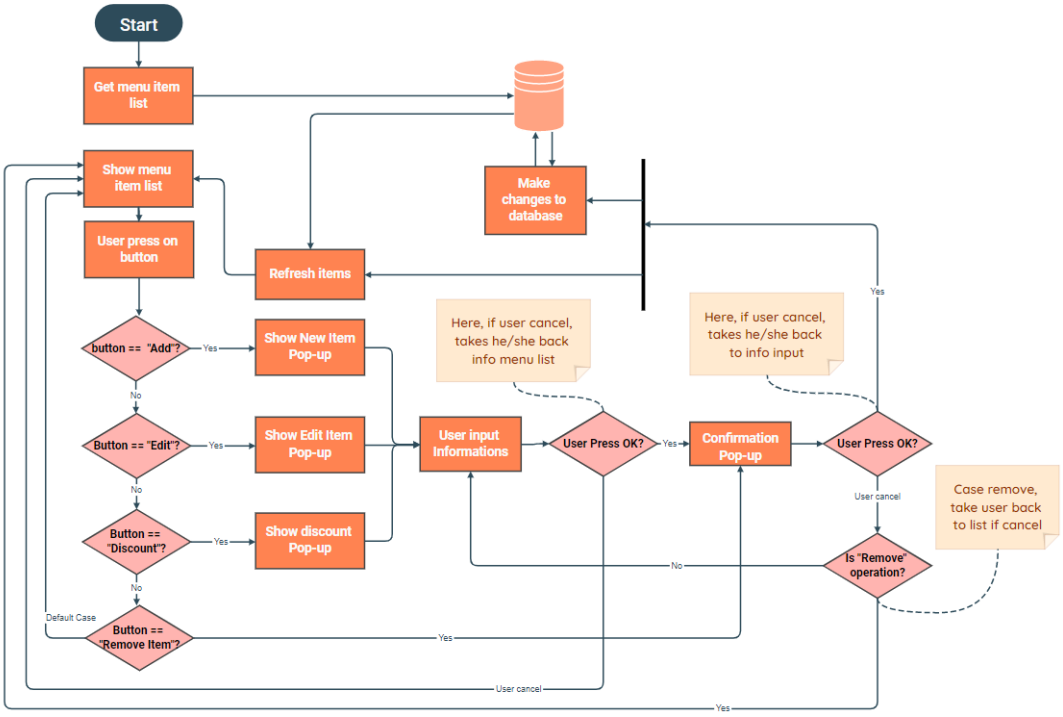


Figure 5: Activity Diagram for menu management of vendor manager

9.1.4.3 Use-case detail/scenario

Use case ID	5.A
Use case name	Modify menu

Actor	Stall's owner
Description	Owner can see the menu of the store and modify it's content like add/remove/discount dishes.
Trigger	User click on the tab "Stall's Menu" or is moved there by the system.
Preconditions	User has to be logged in as a stall owner
Normal flow	1. System shows all items from stall's menu on screen in a list view 2. User choose button to add new item 3. User inputs information of new item and confirms 4. Menu list refreshes
Exceptions	Exception 1 in step 5: a. User does not input information into required fields before confirm b. System announce that user needs to provide information Exception 2 in step 5: a. User input item with same name in the current menu b. System announce that item already exist
Alternative flow	Alternative 1 in step 4: a. User choose remove button on an item and confirm b. Menu list refreshes Alternative 2 in step 4: a. User choose discount option on an item b. User add discount information and confirm c. Menu list refreshes Alternate 3 in step 4: a. User choose edit profile option on an item b. User modify information of item and confirm c. Menu list refreshes Alternate 4 in step 5: a. User do not confirm and instead cancel the operation b. Operation canceled, user sent back to menu list and no updates

9.1.4.4 State Diagram

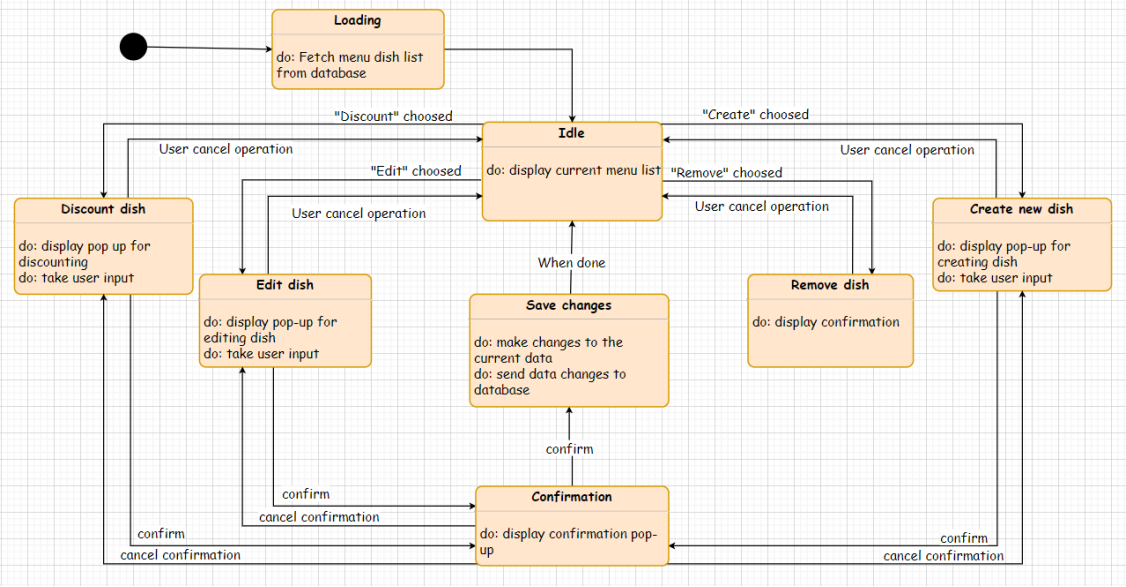


Figure 6: State Diagram for menu management of vendor manager

9.1.4.5 Mock-up

Main components

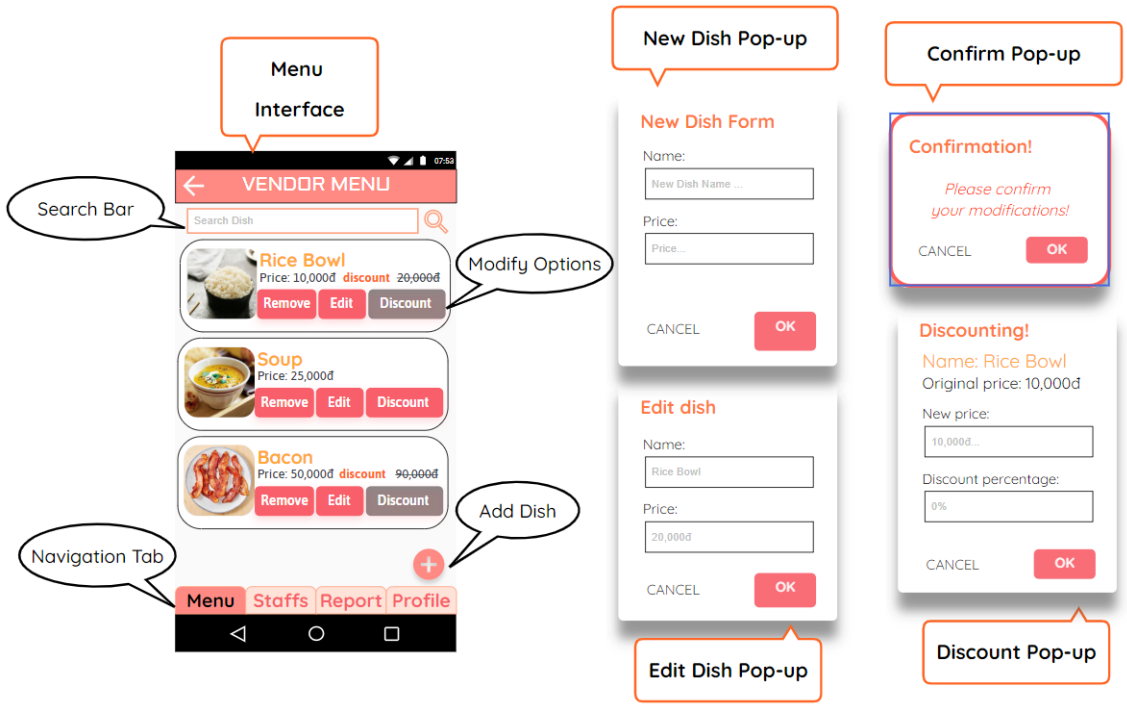


Figure 7: Main parts for Menu Interface

Detailed

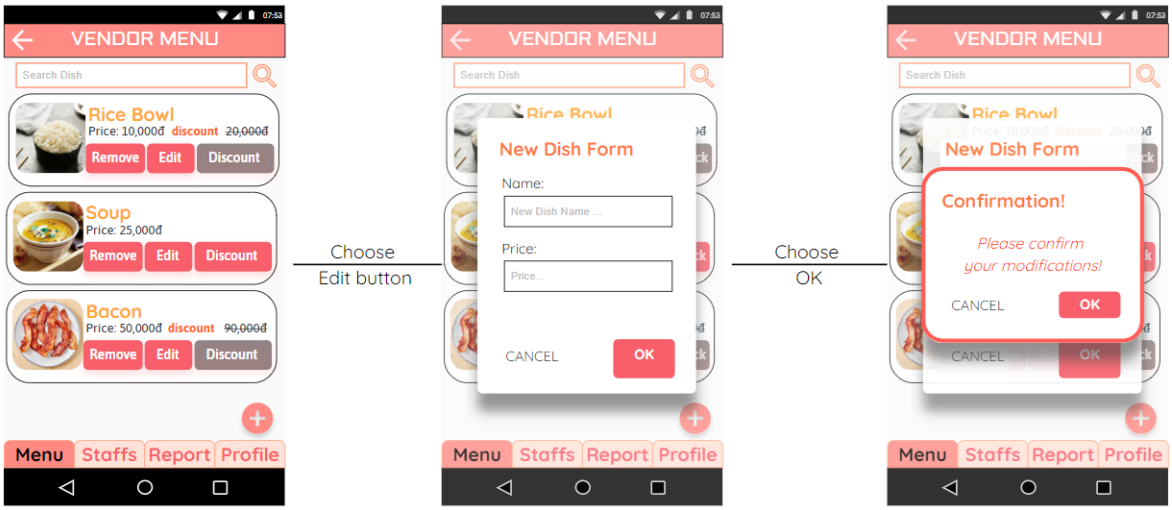


Figure 8: Flow mock-up for adding new dish

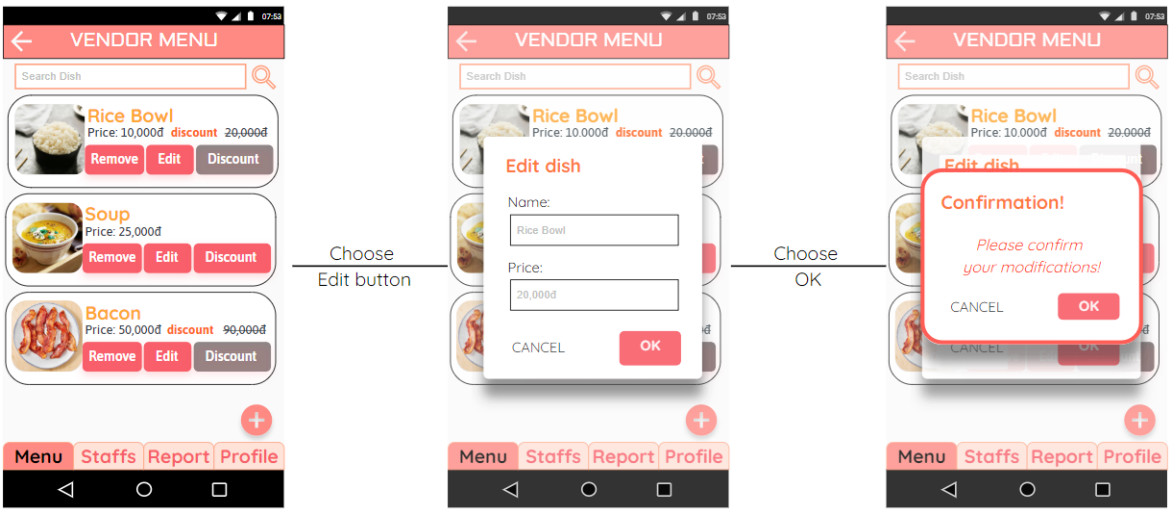


Figure 9: Flow mock-up for editting dish

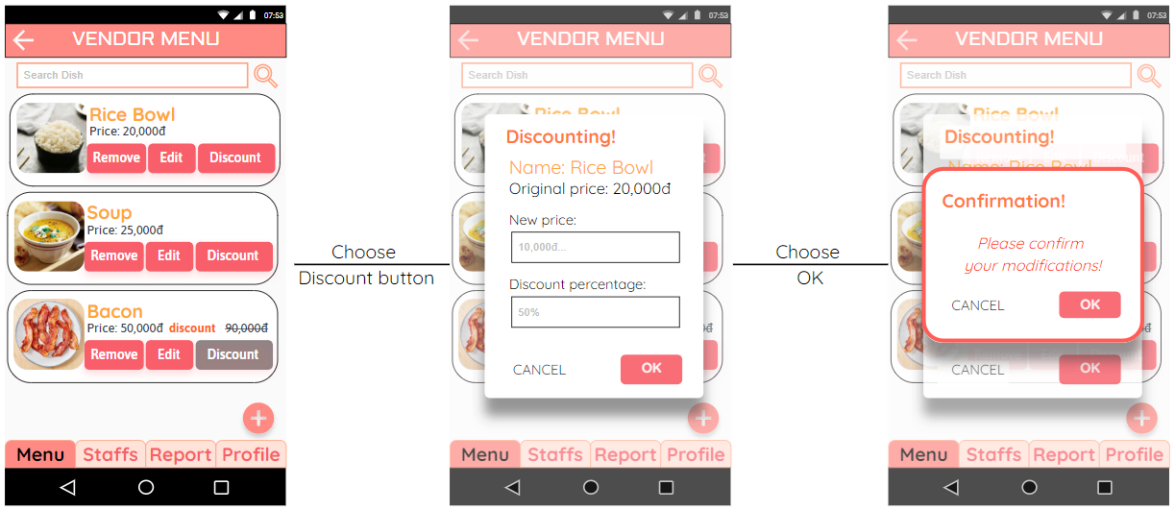


Figure 10: Flow mock-up for dicounting dish

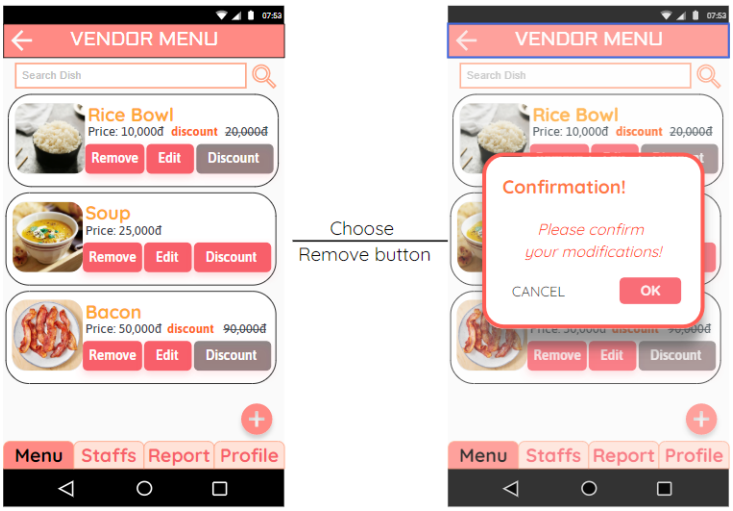


Figure 11: Flow mock-up for removing dish

9.1.4.6 Sequence Diagram

The sequence diagram focuses on the interaction of back-end component rather than the UI (front end). The front end is controlled by the view controller. This controller take user inputs and call the according services to do task, as well and handling display the List¡Dish¿ receive after fetching from database.

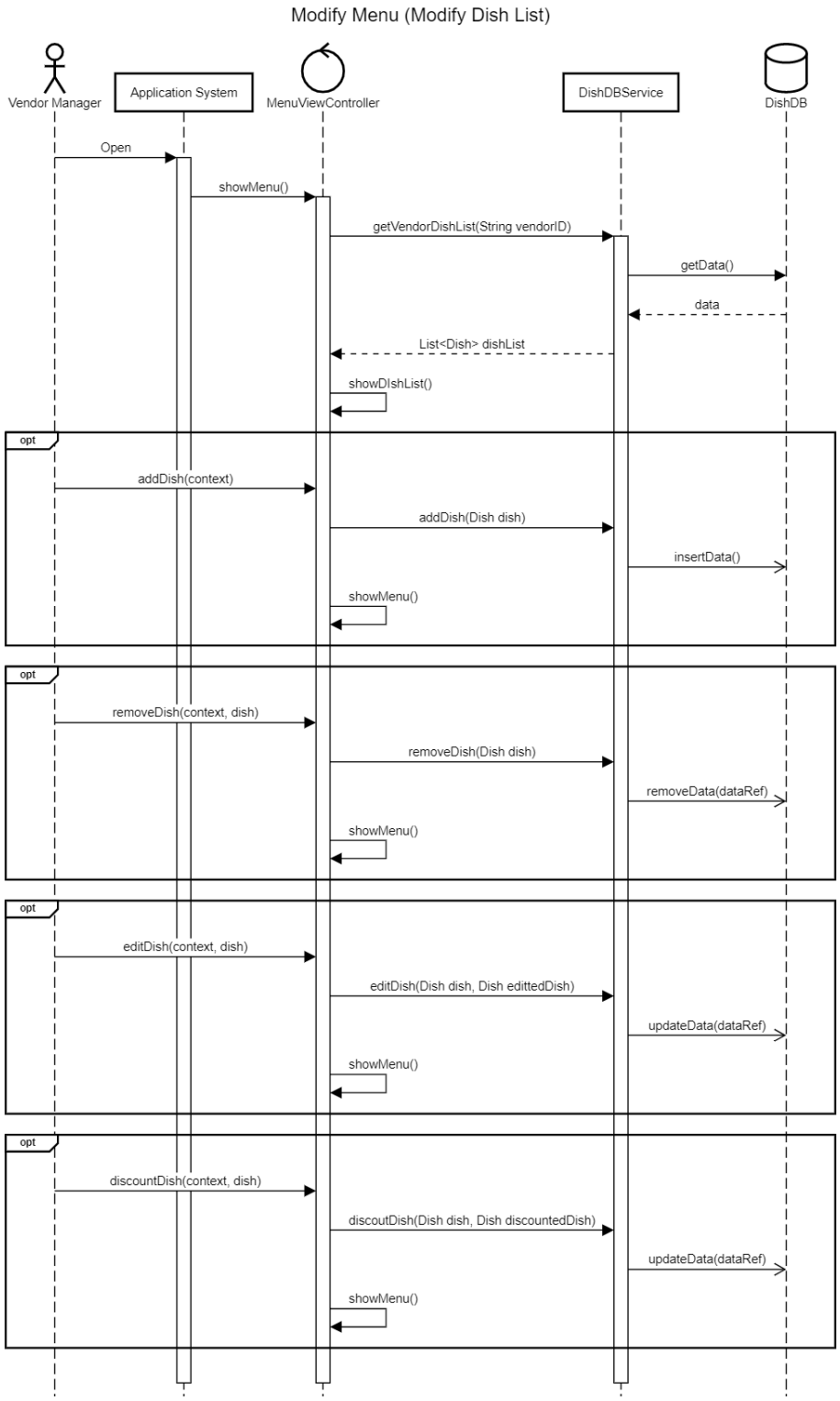


Figure 12: Sequence Diagram Modifying Menu (Dish List)

9.1.4.7 Component Diagram

As the application follows the MVC pattern, the main Package and Component for the whole Use-case of Data Manipulate are Data Manipulate View, Control and Database. Each of which contains the specific components for specific data that we aim to manage. The general picture of the component diagram for the whole "Data manipulation" use-case can be seen Here
For the current sub use-case:

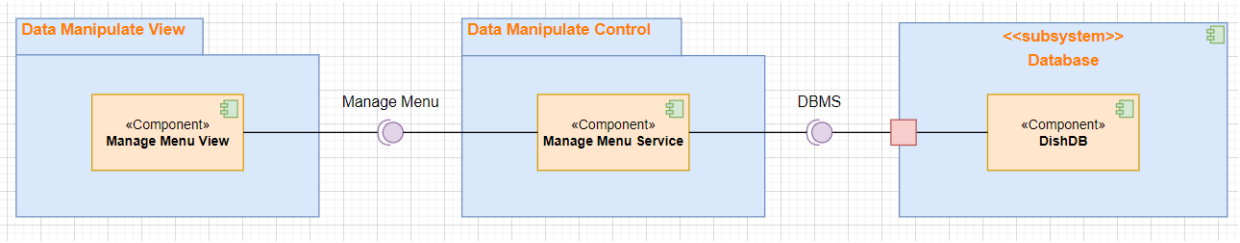


Figure 13: Components involved in Managing Menu use-case

9.1.5 Modify staff list use-case

9.1.5.1 User Story

As a vendor owner or the Food Court manager, i want to see all my hired staff in a list and manage them:

- Add new employees
- Remove employees
- Edit employee's profile

9.1.5.2 Main Flow

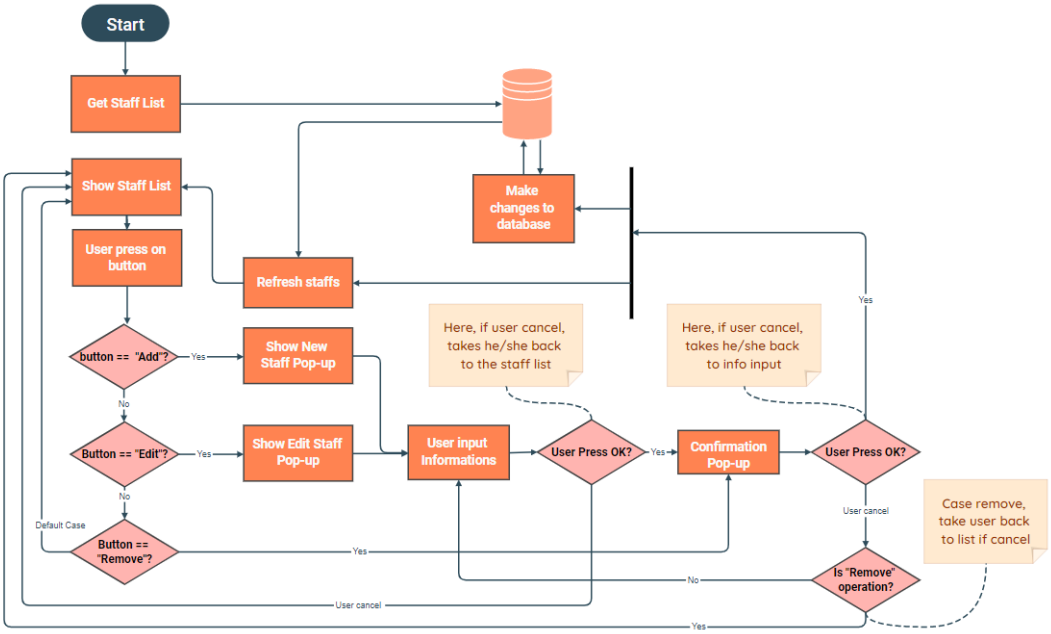


Figure 14: Activity Diagram for staff management of vendor manager or Food Court manager

9.1.5.3 Use-case detail/scenario

Use case ID	5.B
Use case name	Modify staff list
Actor	Stall's owner or Food Court Manager
Description	Owner can see list of all staff and modify staff information, as well as add new staff account.
Trigger	Owner go to "Staff" tab
Preconditions	User needs to be logged-in as a stall's owner or food court manager
Normal flow	1. System shows list view of all staffs on screen 2. User can choose one staff profile to look in a detail page 3. User choose "Register Staff" option 4. User provide log-in information for new staff account and confirm 5. Staff list refreshes

Exceptions	Exception 1 in step 5: a. User do not provide needed information fields b. System announce that user needs to provide information Exception 2 in step 5: a. User provide already existing account name b. System announce that account already exist
Alternative flow	Alternative 1 in step 4: a. User choose remove staff option on a staff and confirm b. Staff list refreshes Alternative 2 in step 4: a. User choose edit profile option in a staff's profile b. User modify needed information and confirm c. Profile refreshes Alternative 3 in step 5: a. User do not confirm but instead cancel operation b. Operation canceled, user is sent back to staff list view and nothing updates.

9.1.5.4 State Diagram

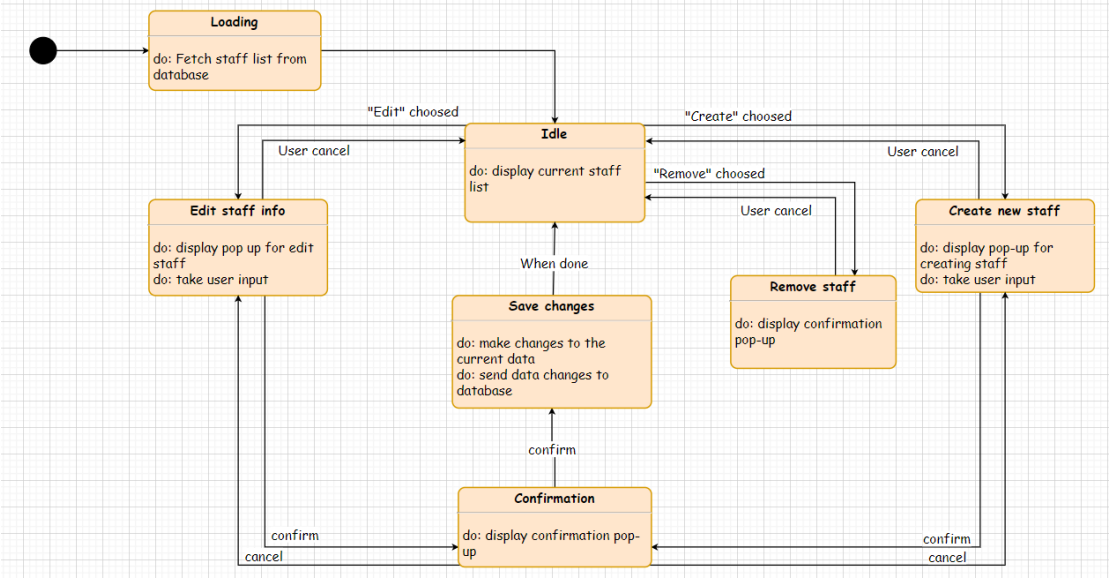


Figure 15: State Diagram for staff management of vendor and food court manager

9.1.5.5 Mock-up

Main parts

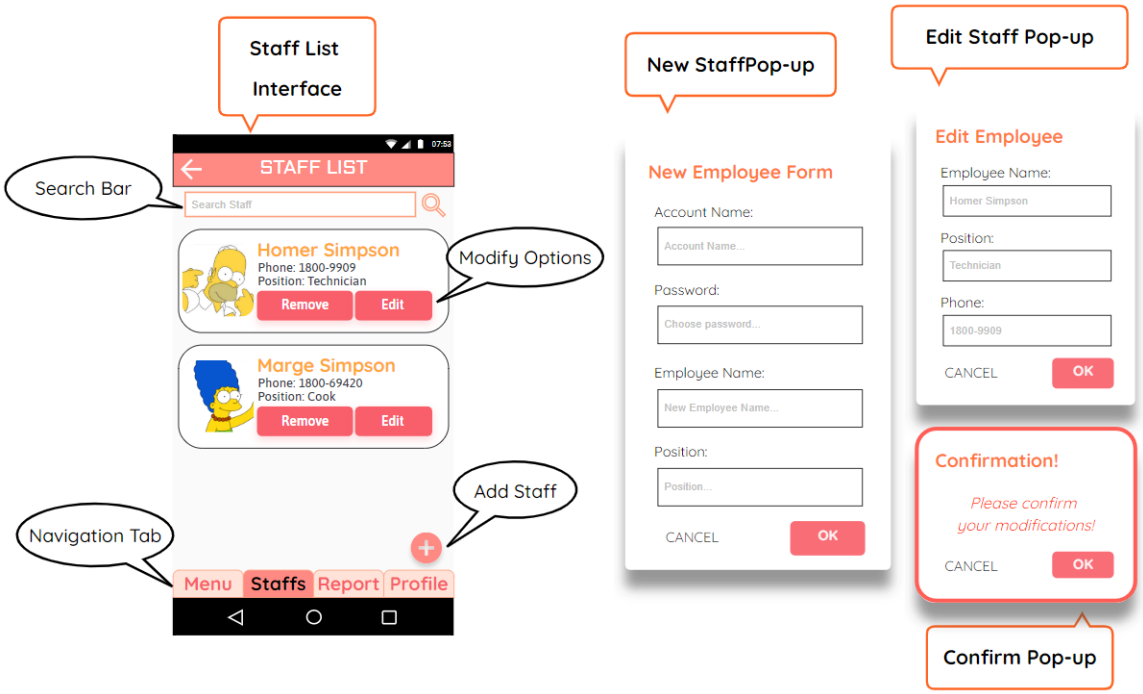


Figure 16: Main parts for Staff List Interface

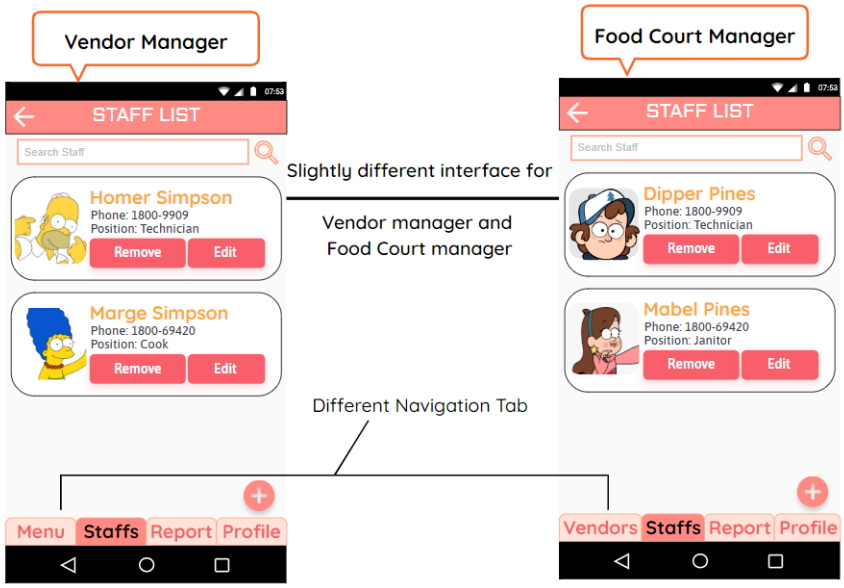


Figure 17: Staff list interfaces

Detailed

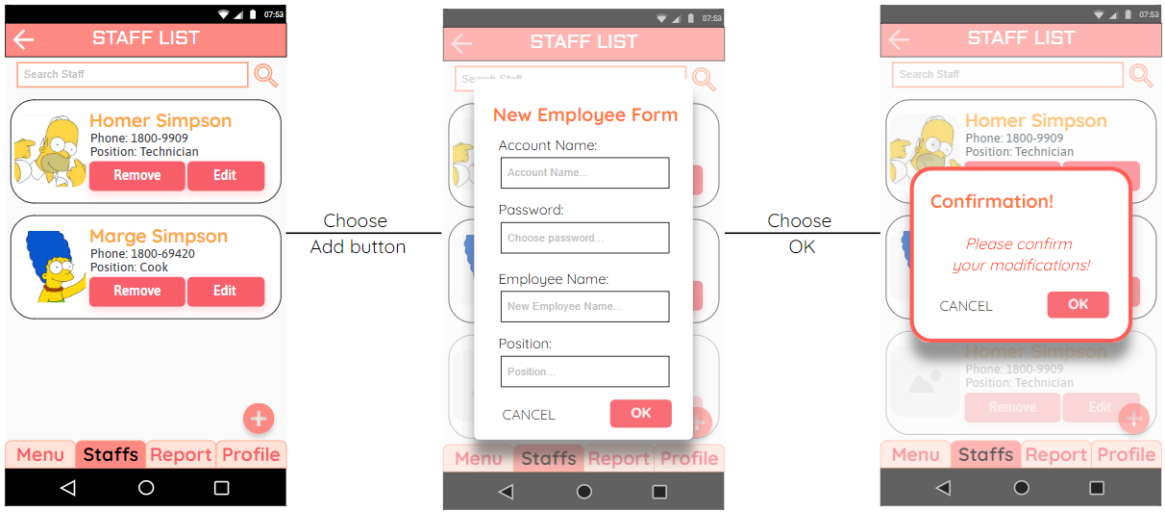


Figure 18: Flow mock-up for adding new staff

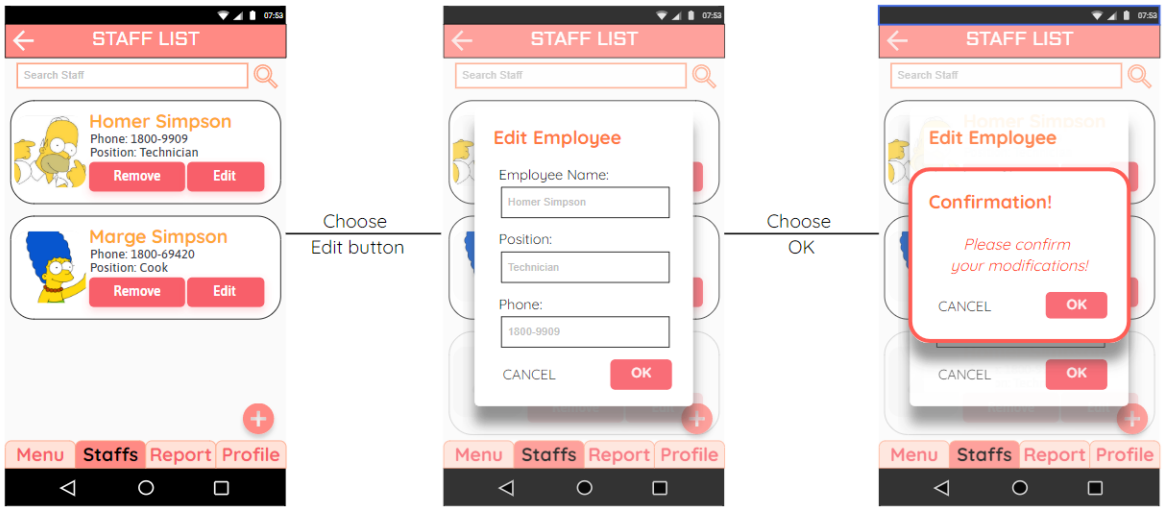


Figure 19: Flow mock-up for editting a staff

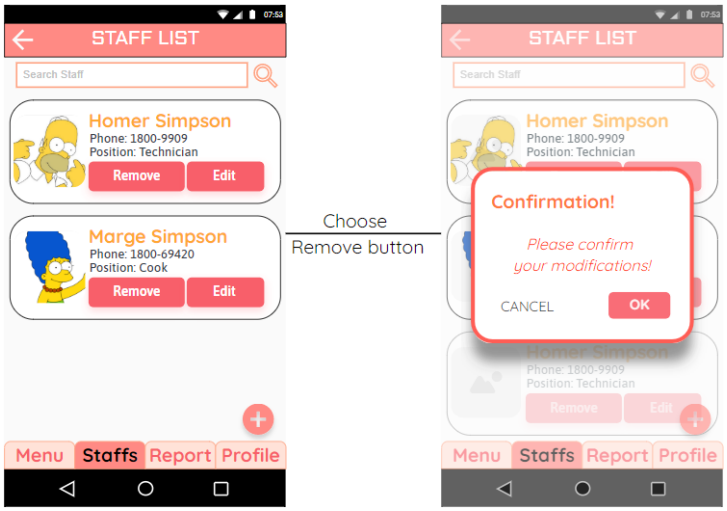


Figure 20: Flow mock-up for removing a staff

9.1.5.6 Sequence Diagram

For the staff list modify use-case, the Vendor Manager and Food Court Manager share the same services, but have different views. This can be seen in the below diagrams.

The sequence diagram focuses on the interaction of back-end component rather than the UI (front end). The front end (Views) are controlled by the View Controller. This controller take user inputs and call the according services to do task, as well and handling display the List¡Staff¿ receive after fetching from database.

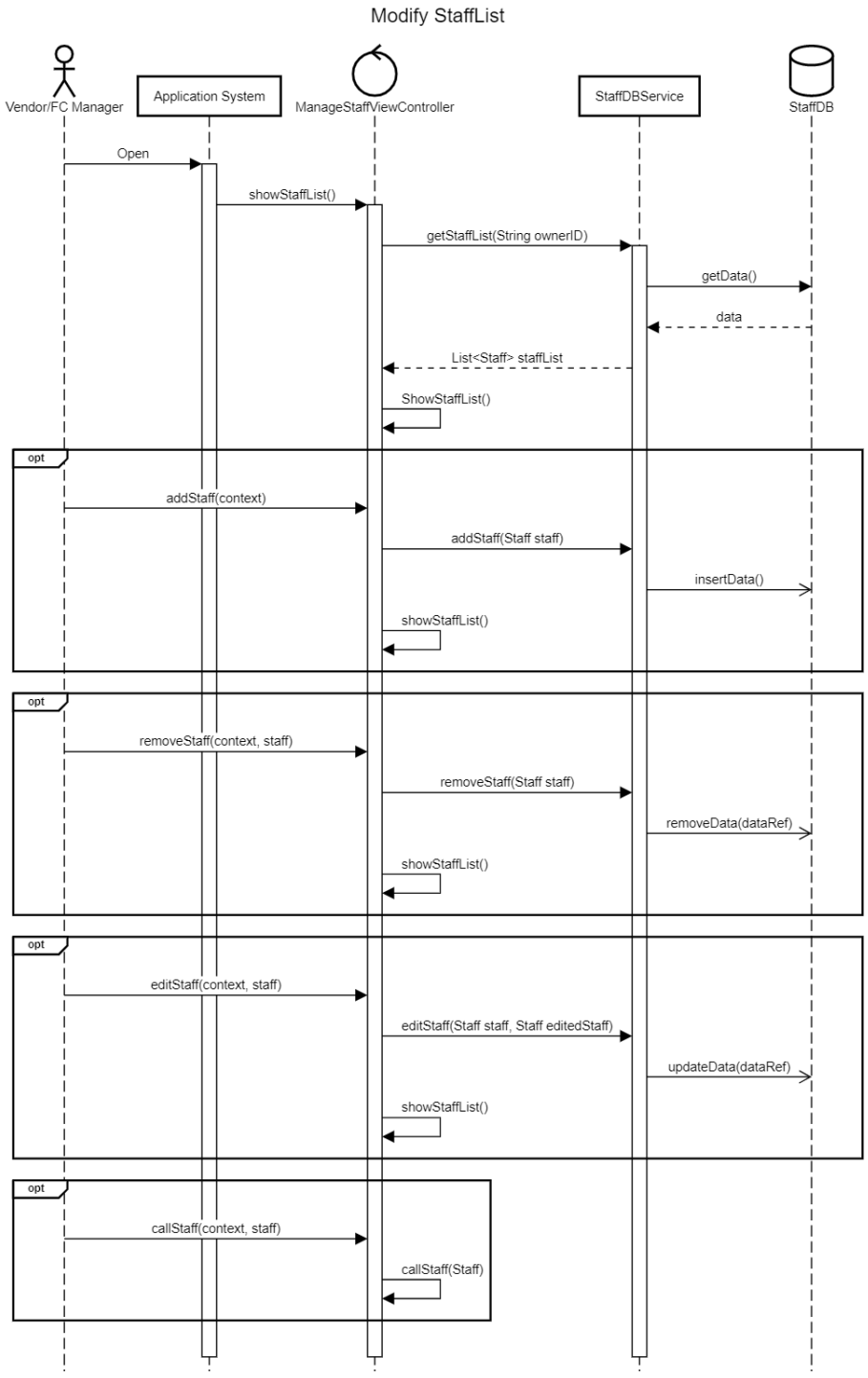


Figure 21: Sequence Diagram Modifying Staff List

9.1.5.7 Component Diagram

As the application follows the MVC pattern, the main Package and Component for the whole Use-case of Data Manipulate are Data Manipulate View, Control and Database. Each of which contains the specific components for specific data that we aim to manage. The general picture of the component diagram for the whole "Data manipulation" use-case can be seen Here
For the current sub use-case:

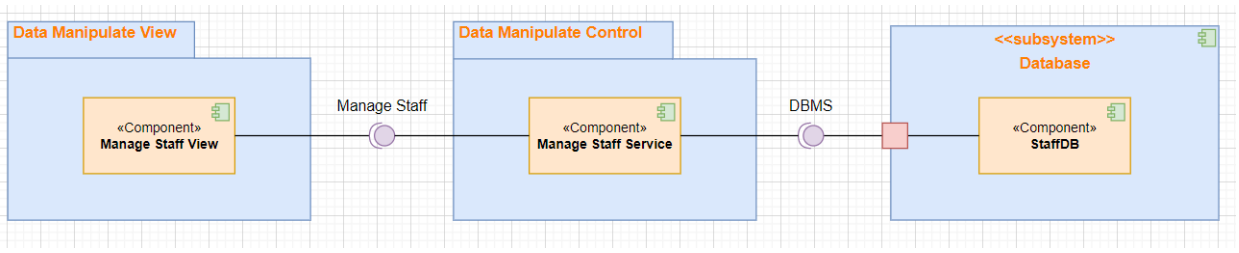


Figure 22: Components involved in Managing Staffs use-case

9.1.6 Modify list of vendors use-case

9.1.6.1 User Story

As the Food Court manager, i want to see all my registered vendors:

- Add new vendor
- Remove vendor
- Edit vendor's profile

9.1.6.2 Main Flow

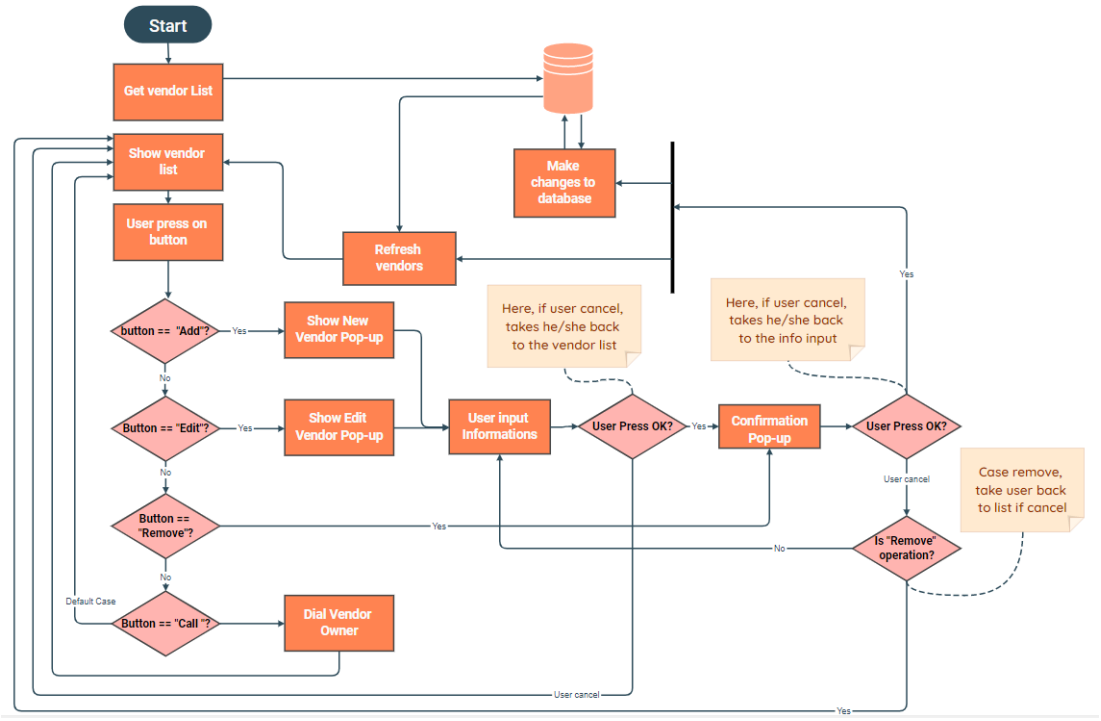


Figure 23: Activity Diagram for vendor management of Food Court manager

9.1.6.3 Use-case detail/scenario

Use case ID	5.C
Use case name	Modify list of stalls
Actor	Food court manager
Description	Food court manager can see list of all stores and add new ones
Trigger	User click on "Stalls" tab
Preconditions	User needs to be logged-in as food court manager
Normal flow	1. System shows list view off all stalls on screen 2. User can choose one stall profile to look in detail 3. User choose "Register Stall" option 4. User provide information for new stall account and confirm 5. Stall list refreshes

Exceptions	Exception 1 in step 5: a. User do not provide needed information fields b. System announce that user needs to provide information Exception 2 in step 5: a. User provide already existing stall name b. System announce that stall already exist
Alternative flow	Alternative 1 in step 4: a. User choose remove stall option on a stall and confirm b. Stall list refreshes Alternative 2 in step 4: a. User choose edit option in a stall element b. User modify needed information and confirm c. Stall profile refreshes Alternative 3 in step 4: a. User choose dial vendor owner b. User redirected to platform system's call service

9.1.6.4 State Diagram

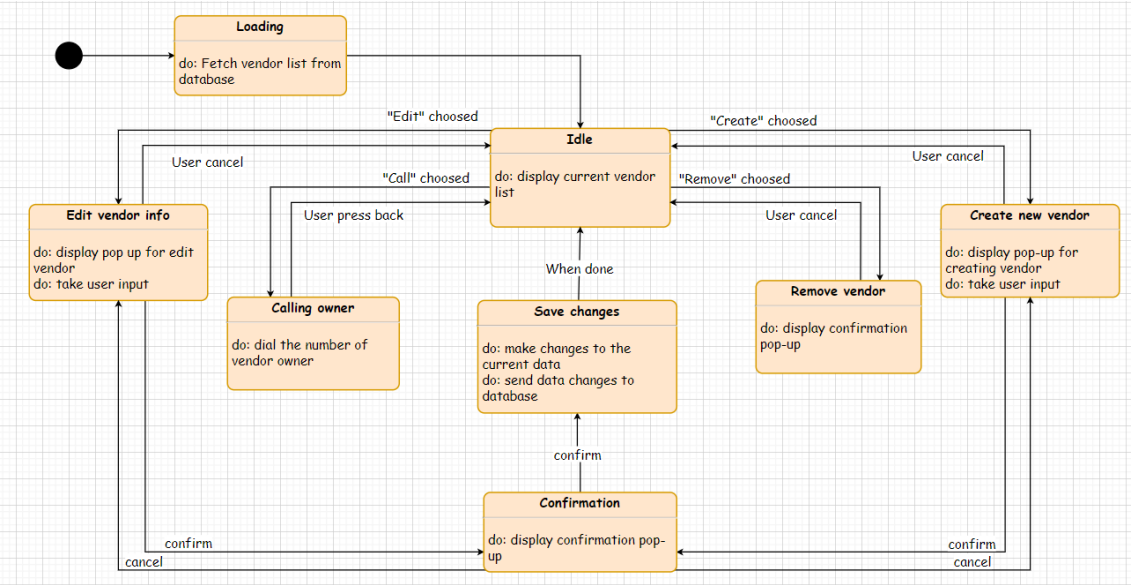


Figure 24: State Diagram for vendor management of food court manager

9.1.6.5 Mock-up

Main components

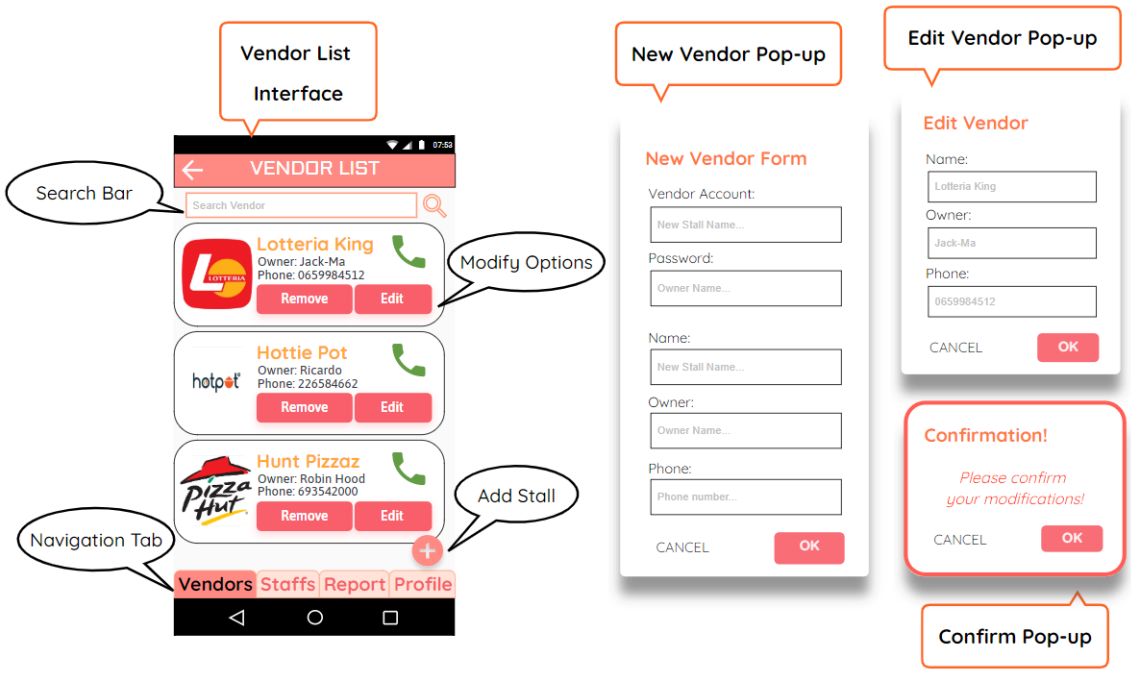


Figure 25: Main parts for Vendor List interface

Detailed

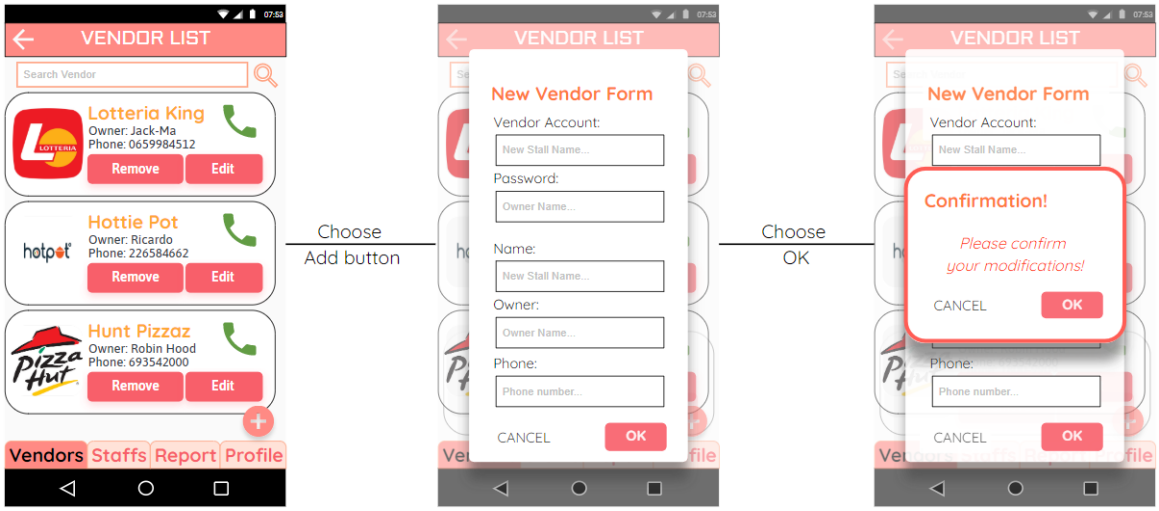


Figure 26: Flow mock-up for adding new vendor

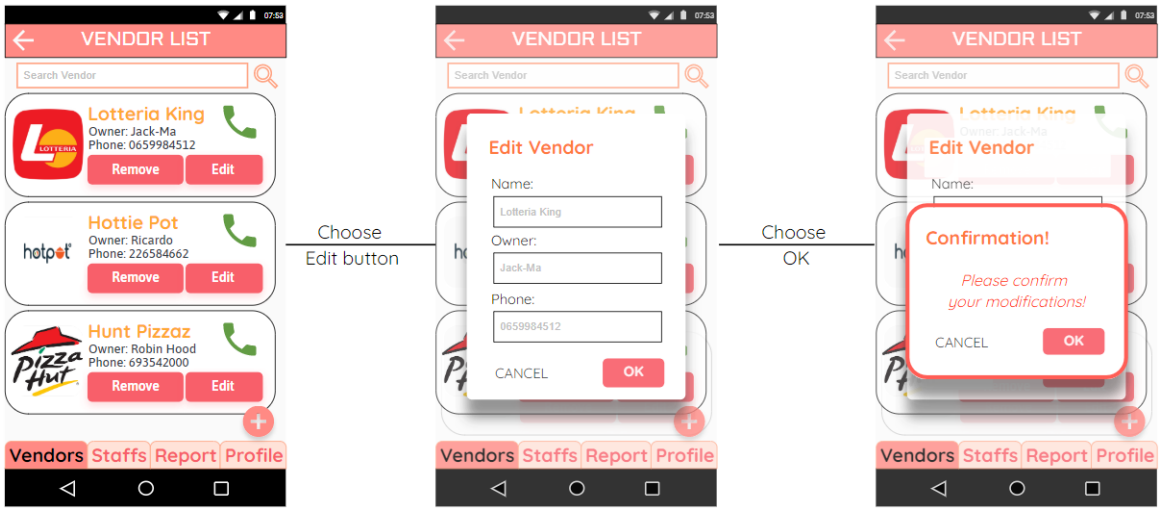


Figure 27: Flow mock-up for editing a vendor

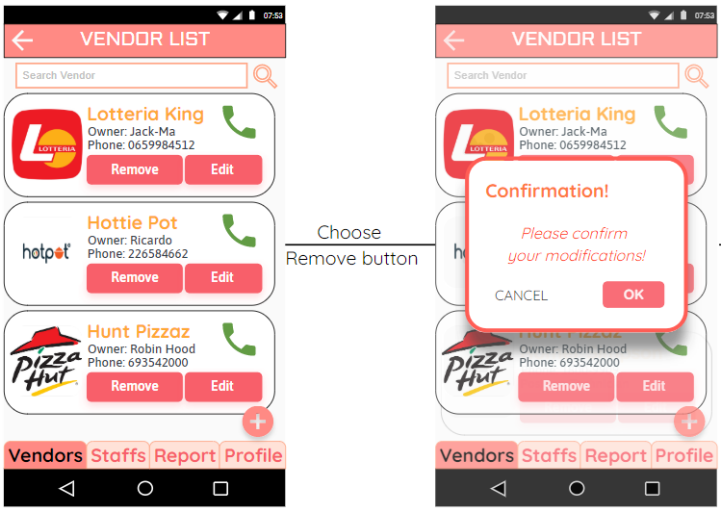


Figure 28: Flow mock-up for removing vendor

9.1.6.6 Sequence Diagram

The sequence diagram focuses on the interaction of back-end component rather than the UI (front end). The front end is controlled by the view controller. This controller take user inputs and call the according services to do task, as well and handling display the List¡Vendor¿ receive after fetching from database.

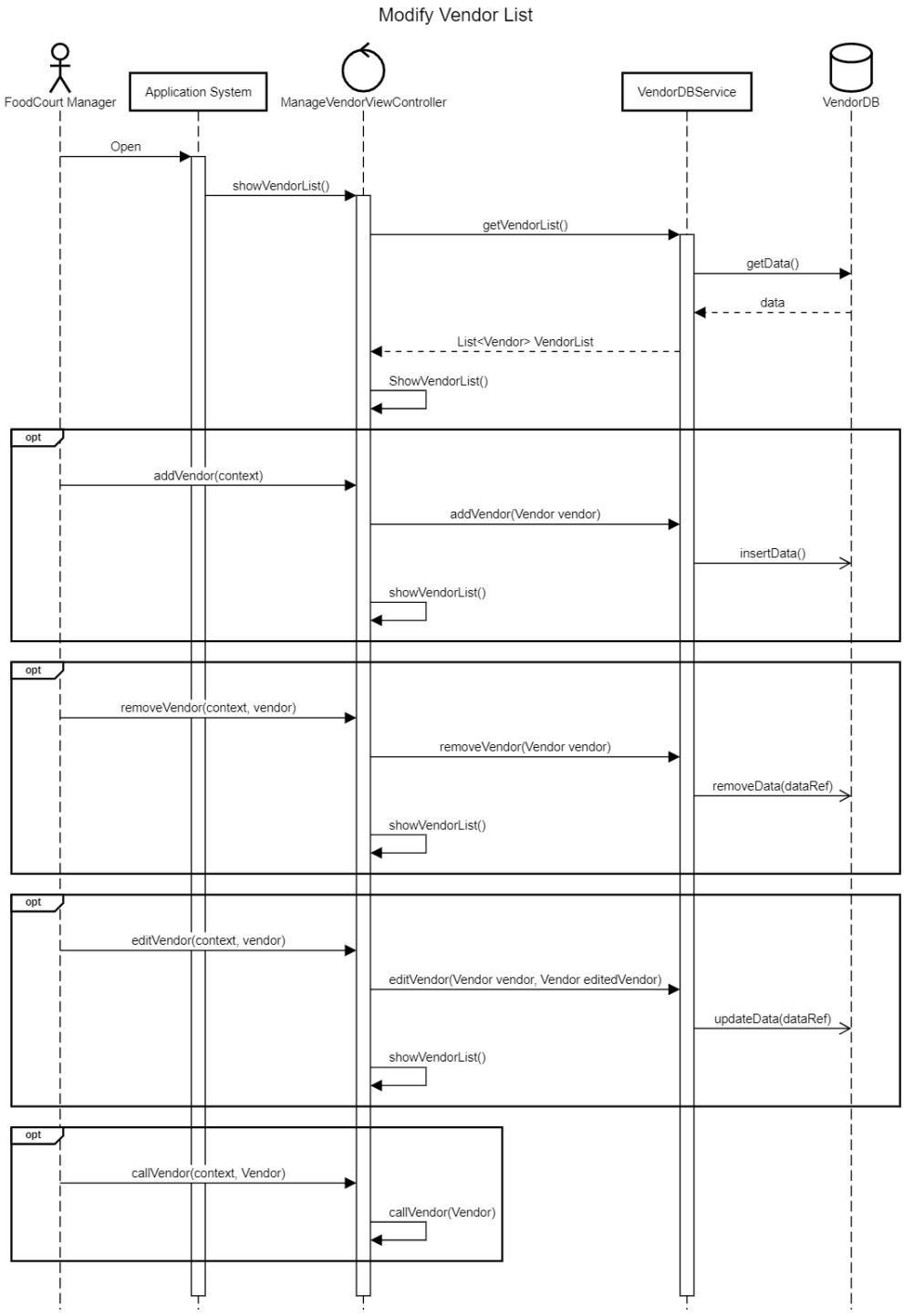


Figure 29: Sequence Diagram Modifying Staff List of Vendor Manager

9.1.6.7 Component Diagram

As the application follows the MVC pattern, the main Package and Component for the whole Use-case of Data Manipulate are Data Manipulate View, Control and Database. Each of which contains the specific components for specific data that we aim to manage. The general picture of the component diagram for the whole "Data manipulation" use-case can be seen Here
For the current sub use-case:

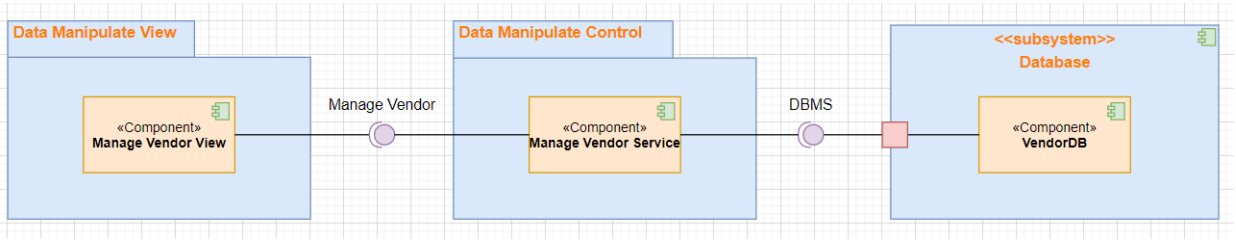


Figure 30: Components involved in Managing Vendors use-case

9.2 Place order

This is use case composed of:

3.A. Add to cart

3.B. Cart Managing

3.C Place order

3.D Track order

Use-case diagram:

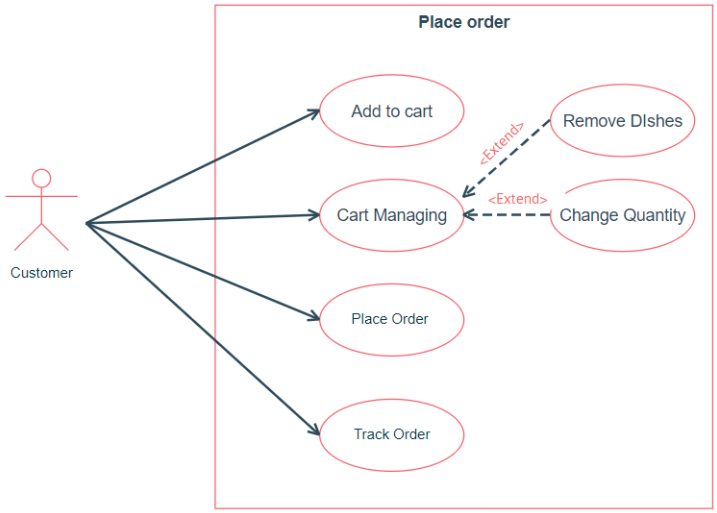


Figure 31: Use case diagram for "Place order"

9.2.1 Add to cart

9.2.1.1 User Story

As a customer, i want to add numbers of product of different vendor to my cart.

9.2.1.2 Main-flow

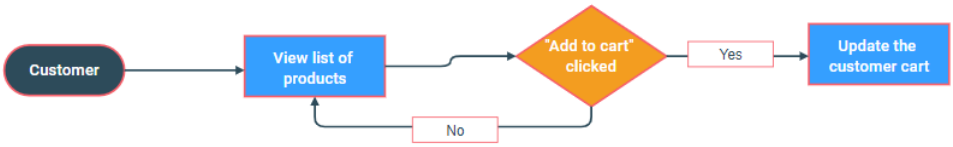


Figure 32: Flow of "Add to cart"

9.2.1.3 Use-case detail/scenario

Use case ID	3A
Use case name	Add to cart
Actor	Customer
Description	Customer add some products to their cart.
Preconditions	Owner has to be logged-in an be at the "Home" tab
Normal flow	1. Customer goes to the "Home" tab 2. After finding a desired product, he can click "Add to cart" button to add it to cart 3. The list of product in cart will be updated
Exceptions	Exception in step 2: a. Customer can't choose a product that is out of stock
Alternative flow	

9.2.1.4 Mock-up

Add to cart

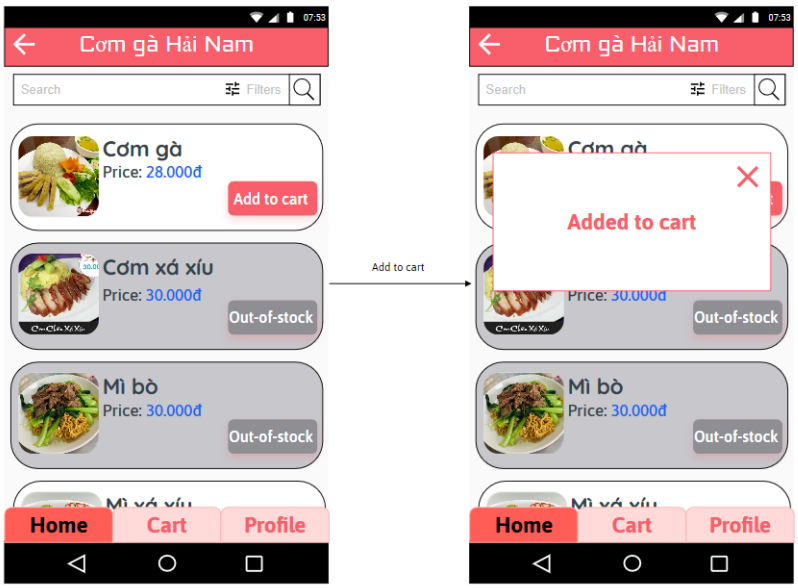


Figure 33: Add a product to cart

9.2.2 Cart Managing

9.2.2.1 User Story

After adding some dishes to cart, customer may want to do some changes, such as removing or changing the quantity of them.

9.2.2.2 Main-flow

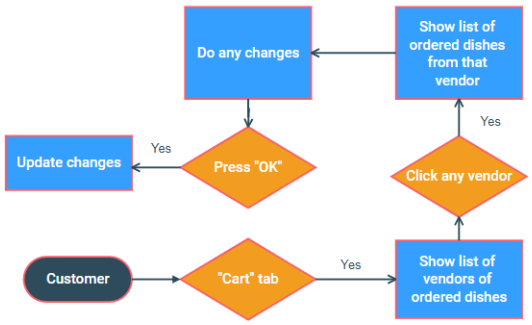


Figure 34: Flow of "Cart Managing"

9.2.2.3 Use-case detail/scenario

Use case ID	3B
Use case name	Cart Managing
Actor	Customer
Description	Customer manages dishes in his cart (removing, changing quantity).
Preconditions	There are at least one dish added to cart
Normal flow	1. Customer goes to the "Cart" tab 2. The tab will show all vendors of dishes that customer has added and the total price of each as well as the total price of whole cart. 3. Click any vendor, the customer will see list of ordered dishes from that vendor. 4. The customer can remove a products from their cart, change quantity or keep everything the same 5. Customer clicks "OK" button 6. The system will update the view with the changes.
Exceptions	
Alternative flow	In step 5, click "Cancel" instead of "OK" a. Back to cart view without updating any changes.

9.2.2.4 Mock-up

In any cases, if the changes is confirmed, the price and quantity will be updated correctly. While pressing "Cancel" will ignore all changes.
Changing quantity (for example, increase by one). Decreasing also work the same way. However, customer can not the press "-" when the quantity is 1, instead he can press "remove" as demonstrated below).

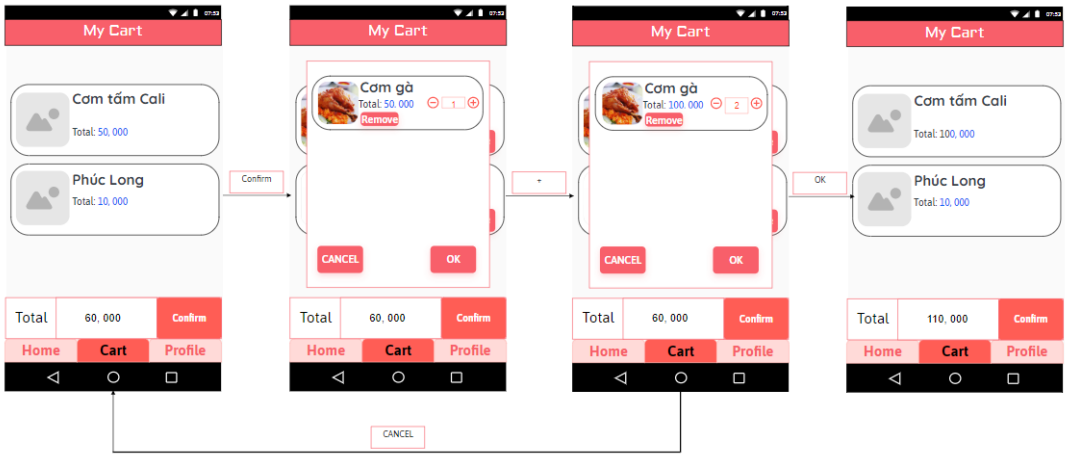


Figure 35: Increase quantity by one

Remove a dish. If there are not any dishes left in that vendor, it will also be removed from the cart.

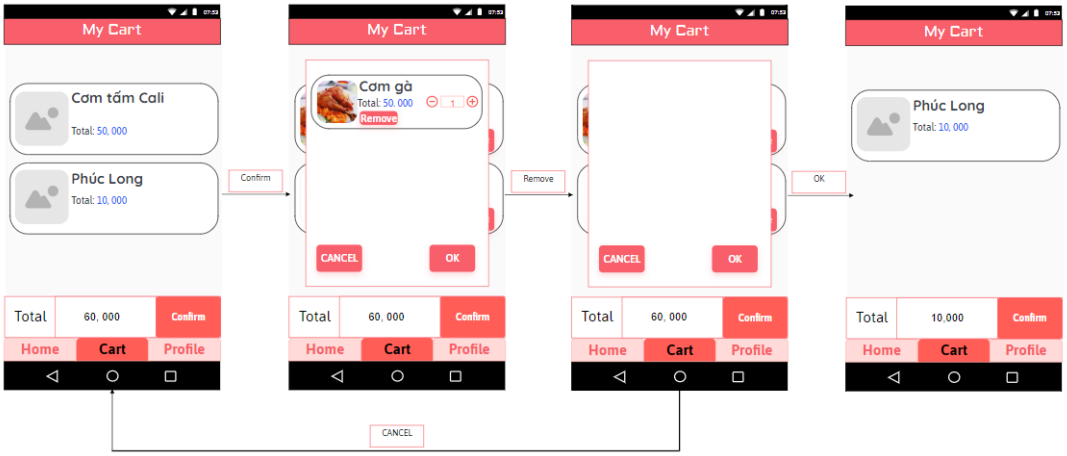


Figure 36: Remove the only dish of the vendor

9.2.3 Place Order

9.2.3.1 User Story

Customer places order.

9.2.3.2 Main-flow

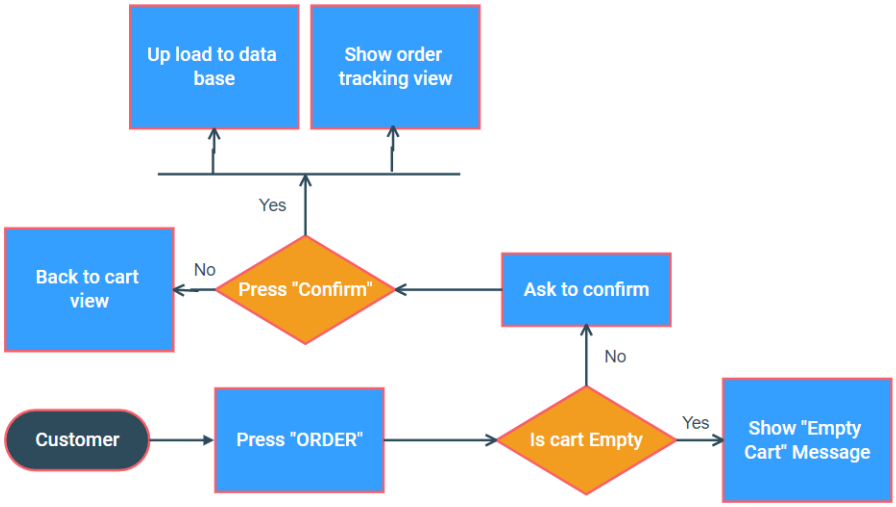


Figure 37: Flow of "Place order"

9.2.3.3 Use-case detail/scenario

Use case ID	3C
Use case name	Place Order
Actor	Customer
Description	Customer places order.
Preconditions	
Normal flow	1. Customer goes to the "Cart" tab 2. Customer press "ORDER" 3. A form will pop up to ask whether they want to order 4. Customer clicks "Confirm" button 5. The system will upload the order to the database.
Exceptions	
Alternative flow	In step 4, click "Cancel" instead of "OK" a. Back to cart view. In step 2, if the cart is empty a. Inform the customer that the cart is empty.

9.2.3.4 Mock-up

Here is only the mockup for the case that the cart is empty. For the case that the order is successfully placed, we will described it clearly in the next feature.

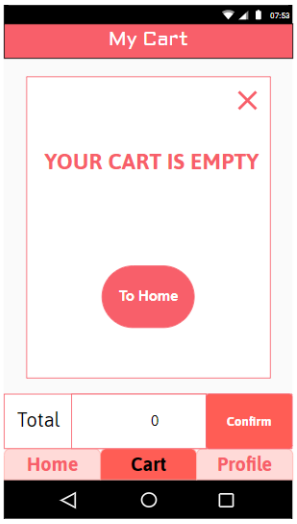


Figure 38: The cart is empty

9.2.4 Track Order

9.2.4.1 User Story

After placing orders, customer can track the status of order sent to each vendor.

9.2.4.2 Main-flow

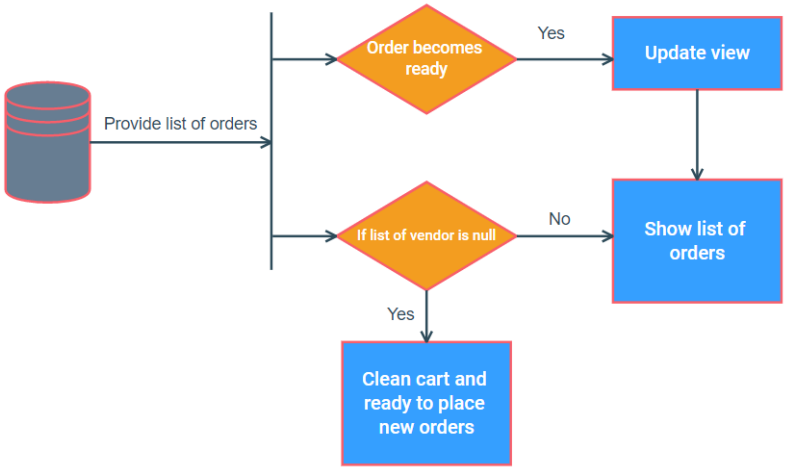


Figure 39: Flow of "Track Order"

9.2.4.3 Use-case detail/scenario

Use case ID	3D
Use case name	Track Order
Actor	Customer
Description	Customer track their order.
Preconditions	After successfully placing orders
Normal flow	1. After pressing "Confirm" in previous step. 2. A windown will pop up with a list of order associated with each vendor 3. If an order is ready served (as the staff click inform in their app) the status of the order will change into "ready" so that the customer can come to pick it up, 4. After the customer gets their order, the order will be delete from the list. 5. When there are no orders left, the cart will be back to normal and ready to place new order.
Exceptions	
Alternative flow	

9.2.4.4 Mock-up

After pressing "confirm" in the previous use case, these following screens will appear.

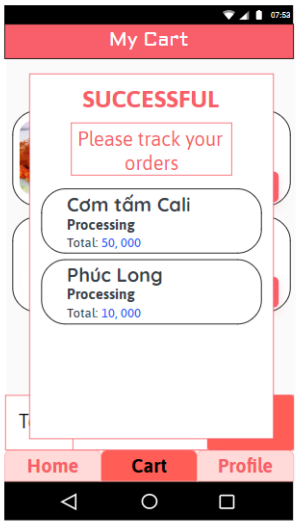


Figure 40: Tracking the order

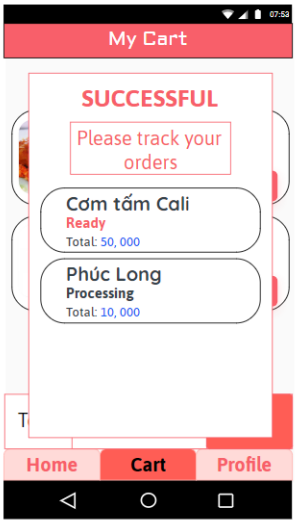


Figure 41: When there are any "ready" orders

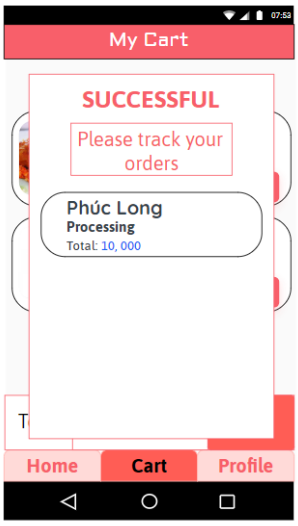


Figure 42: When customer has picked the ready order

After all orders have been picked, the cart will be clean and customer can start ordering again.

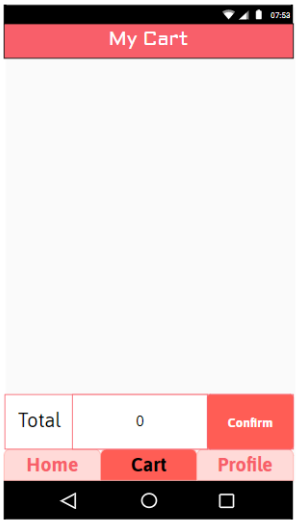


Figure 43: All orders have been picked

9.2.5 Implementation

Class diagram

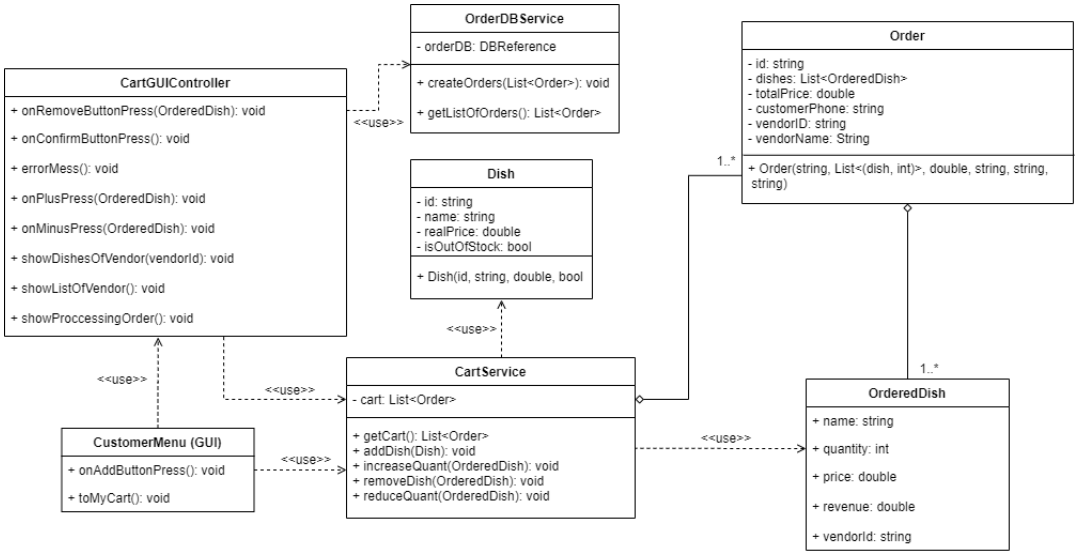


Figure 44: Class diagram of place order features

Method Description

Class: CartGUIController

Name	Description
onRemoveButtonPress	When the remove button of a dish is press in the cart view, this will call **removeDish()** in the service to remove it.
onPlusPress	To increase the quantity of a dish in the cart by 1. This function will call **increaseQuant()** in **CartService** to do the job.
onConfirmButtonPress	This will confirm that the order is ready to be send to the vendor. This will call **createOrder().**
errorMess	Inform user that the cart is empty when **confirmOrder()** return **false.**
onMinusPress	To decrease the quantity of a dish in the cart by 1. This function will call **reduceQuant()** in **CartService** to do the job.
showListOfVendor	Show list of vendor of which dishes are ordered. By clicking each vendor, customer can see the dishes ordered from that vendor.
showDishesOfVendor	Show dishes ordered from that vendor.
showProcessingOrders	Show the status of orders to customer

Class: CartService

Name	Description
getCart	Get the **cart** so that it can be show to customer
addDish	Add dish to the **Order** which has associated **vendorID**, if that object is not exist, create new one then add to **cart**
removeDish	Remove the element (include dish and quantity) associated to the dish from the list. This will also update the **totalPrice.**
increaseQuant	Increase the quantity of chosen dish by 1. This will also update the **totalPrice** in both **CartService** and the associated **Order.**
reduceQuant	Decrease the quantity of chosen dish by 1. This will also update the **totalPrice** in both **CartService** and the associated **Order.**

Class: CustomerMenu (GUI)

Name	Description
onAddButtonPress	When the user press the add button of a dish in the menu view, this will call **addDish()** to add a dish to cart.
toMyCart	When the user switch to cart view, this will call **showListOfVendors()** to show the detail of the order in the cart as well as provide the functionality of modify it.

Class: OrderDBService

Name	Description
createOrder	Upload orders with associated **vendorId** to database so that staff of those vendors can get it.
getListOfOrders	Stream the list of **Order** from the database, which call automatically when there is any changes of Order in the database.

9.3 Report Management

This feature is composed of

A. View Report

B. Generate Report

9.3.1 Class Diagram

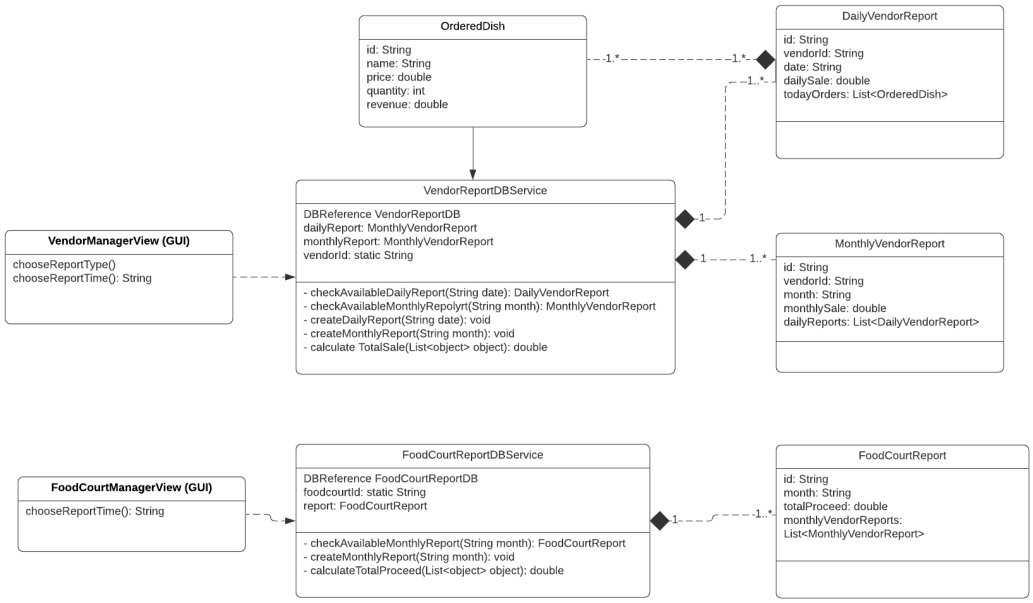


Figure 45: Class diagram for View Report

9.3.2 Method Description

- Class VendorReportDBService

Method Name	Input	Output	Description
checkAvailableDailyReport	date	DailyReport of that date	- Search the report by date - Request to print the report in GUI
checkAvailableMonthlyReport	month	MonthlyReport of that month	- Search the report by month - Request to print the report in GUI
createDailyReport	date	none	- Create a daily report in the firebase - This function is implemented by the staff
createMonthlyReport	month	none	- Create a monthly report in the firebase - This function is implemented in the background
calculateTotalSale	List of DailyVendorReport or orderedDish	total sale	- Calculate the total sale of that report

- FoodCourtReportDBService

Method Name	Input	Output	Description
checkAvailableMonthlyReport	month	MonthlyReport of that month	- Search the report by month - Request to print the report in GUI
createMonthlyReport	month	none	- Create a monthly report in the firebase - This function is implemented in the background
calculateTotalProceed	List of Monthly VendorReport	total proceed	- Calculate the total proceed of the food court

9.3.3 View Report

9.3.3.1 Use-case diagram

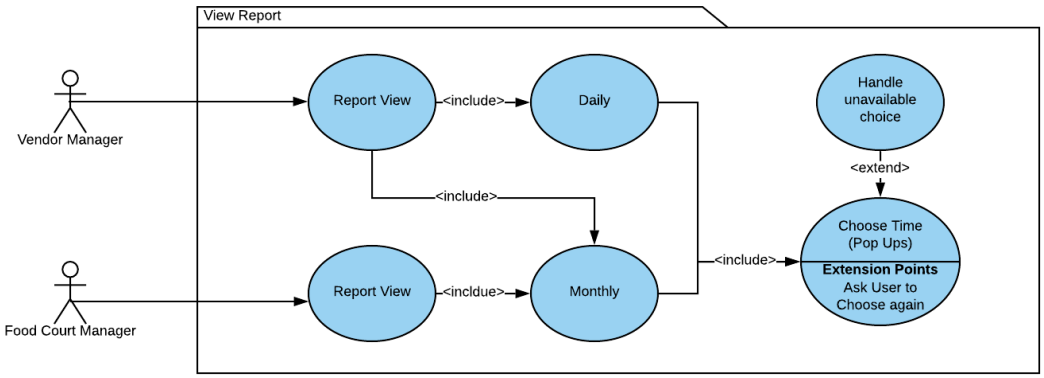


Figure 46: Use case diagram for View Report

9.3.3.2 User Story

As an owner or manager, i want to view the report of the vendor or food court on any day or month.

- Choose report type
- Choose specific time(date or month) depend on report type

9.3.3.3 Main-flow

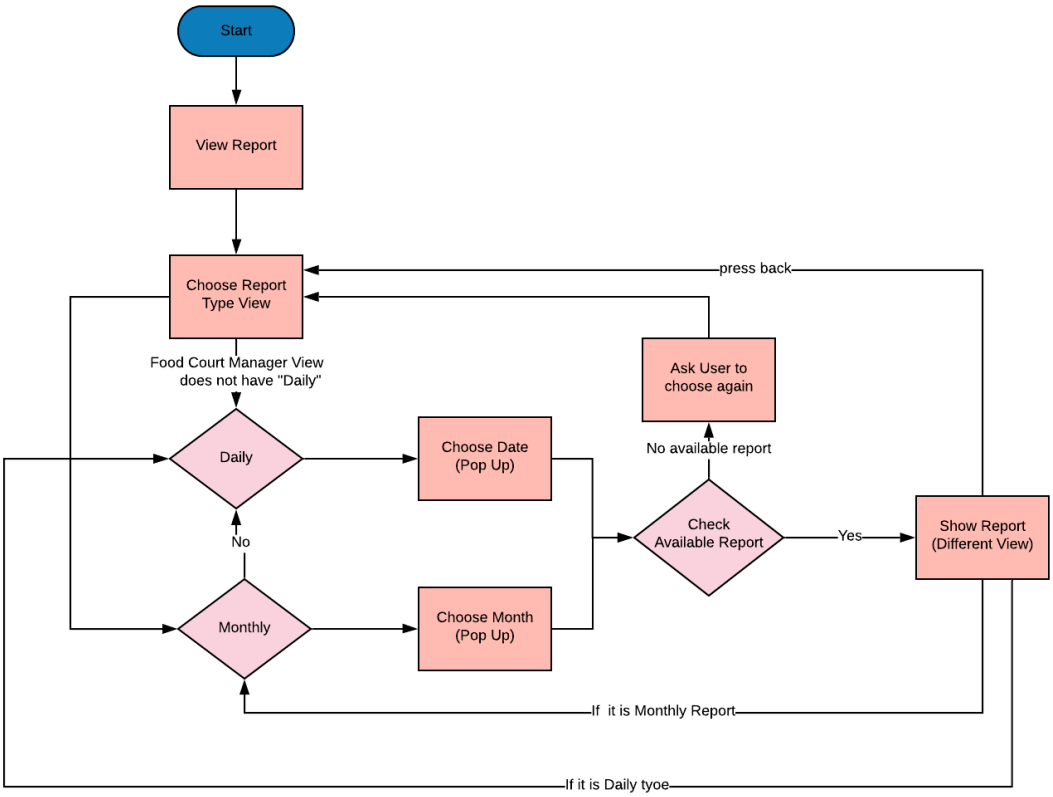


Figure 47: Flow chart for View Report

9.3.3.4 Use-case detail/scenario

Use case ID	5
Use case name	View Report
Actor	Vendor managers Food Court manager

Description	a) Daily: The vendor managers can view a report from the app shows the stats of the day (number of orders of each dish, number of orders in total, ...). b) Monthly: • Vendor Manager: View all the daily reports in that month. Each daily reports shows the total sale, date and the increment compared to the previous day. There is a total sale field at the bottom. • Food Court Manager: View the monthly reports of all vendors in that month. Each monthly reports shows the total sale of that vendor in that month. There are fields such as(commissions, rent, total paid...). At the bottom there is a total proceed field.
Trigger	Managers click "View Report"
Preconditions	Managers have to log in to the app
Normal flow	1. Managers log in to the app by their account type. 2. Managers click "View Report". 3. Managers choose "Monthly". 4. Managers choose a month in the calendar pop up. 5. (a) Owners: The app shows report of statistics every daily reports in that month (b) Manager: The app shows report compiled from every vendors monthly reports. 6. Managers choose "Choose Month"(works the same as "Monthly" but in different view) and move to step 4.
Exceptions	Exception in step 5 and 6: a. If there is no available report in that month, the app will tell the owners and ask them to choose again.

Alternative flow	Alternative in step 3, 4, 5 and 6(continuously): a. step 3: Vendor Managers choose “Daily” instead of “Monthly”(Food Court Manager View does not have this use case(button)) b. step 4: Vendor Managers choose a date calendar pop up. c. step 5: The app shows report of statistics of sold products, sales on that day. d. step 6: Vendor Managers choose ”Choose date”(works the same as ”Daily”) and move to step 4

9.3.3.5 State Diagram

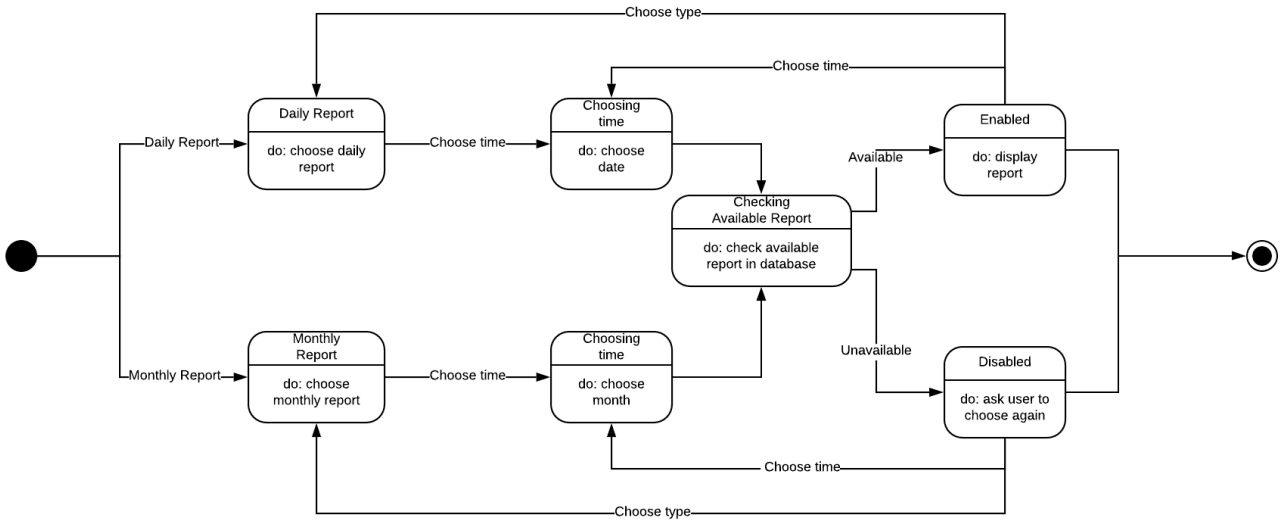


Figure 48: State Diagram for View Report

9.3.3.6 Mock-up

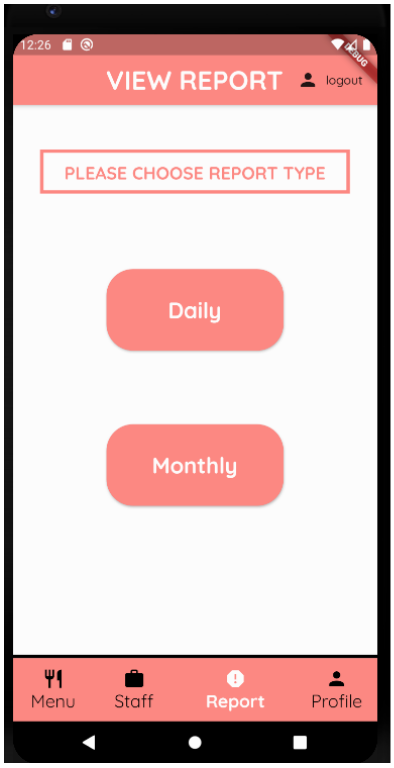


Figure 49: View Report layout for Vendor Manager

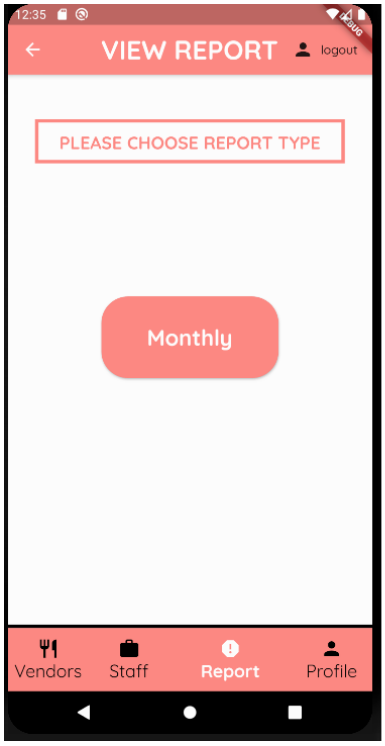


Figure 50: View Report layout for Food Court Manager

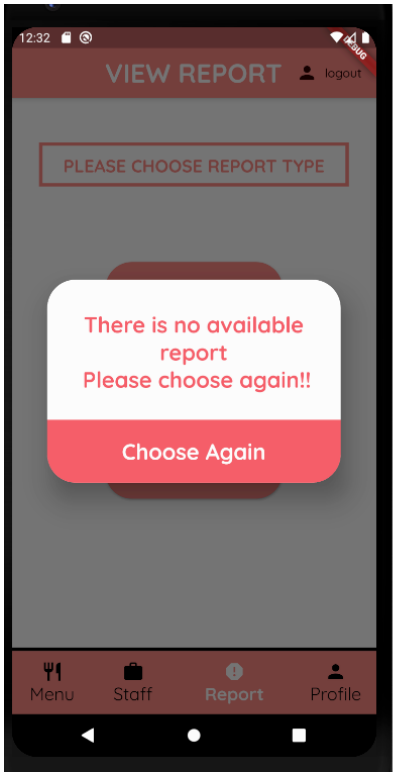


Figure 51: Invalid Pop Up

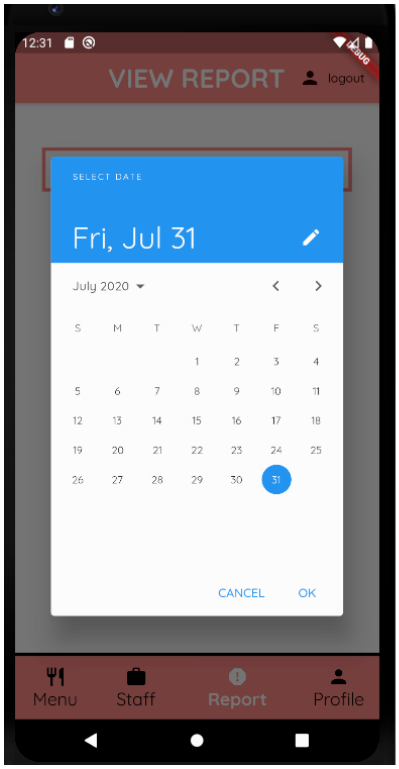


Figure 52: Choose Date Pop Up

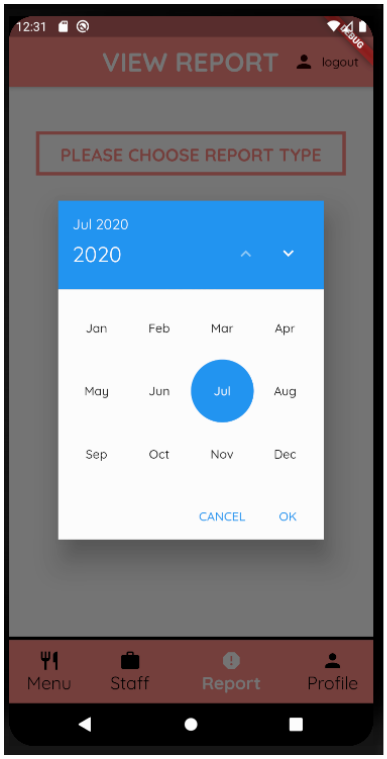


Figure 53: Choose Month Pop Up

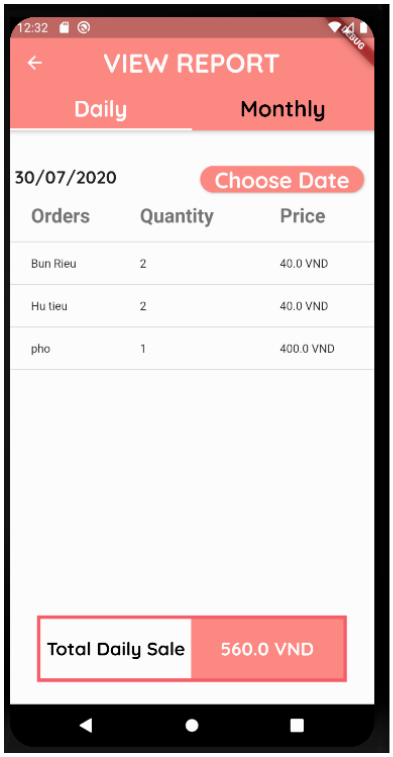


Figure 54: Daily Vendor Report

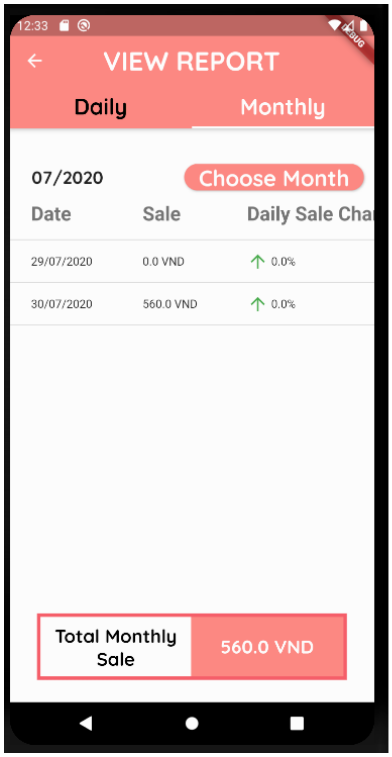


Figure 55: Monthly Vendor Report

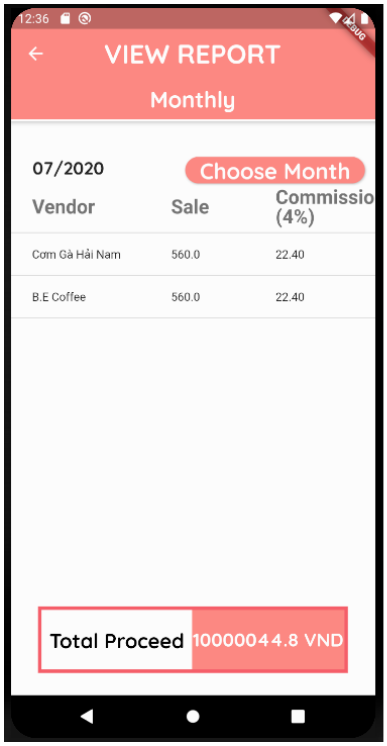


Figure 56: Monthly Food Court Report

9.3.3.7 Sequence Diagram

Both Vendor and Food Court manager share the same view in View Report but the View for Food Court manager does not include "Daily Report" choice. The diagram below represents the sequence of users interaction with the View Report (GUI)

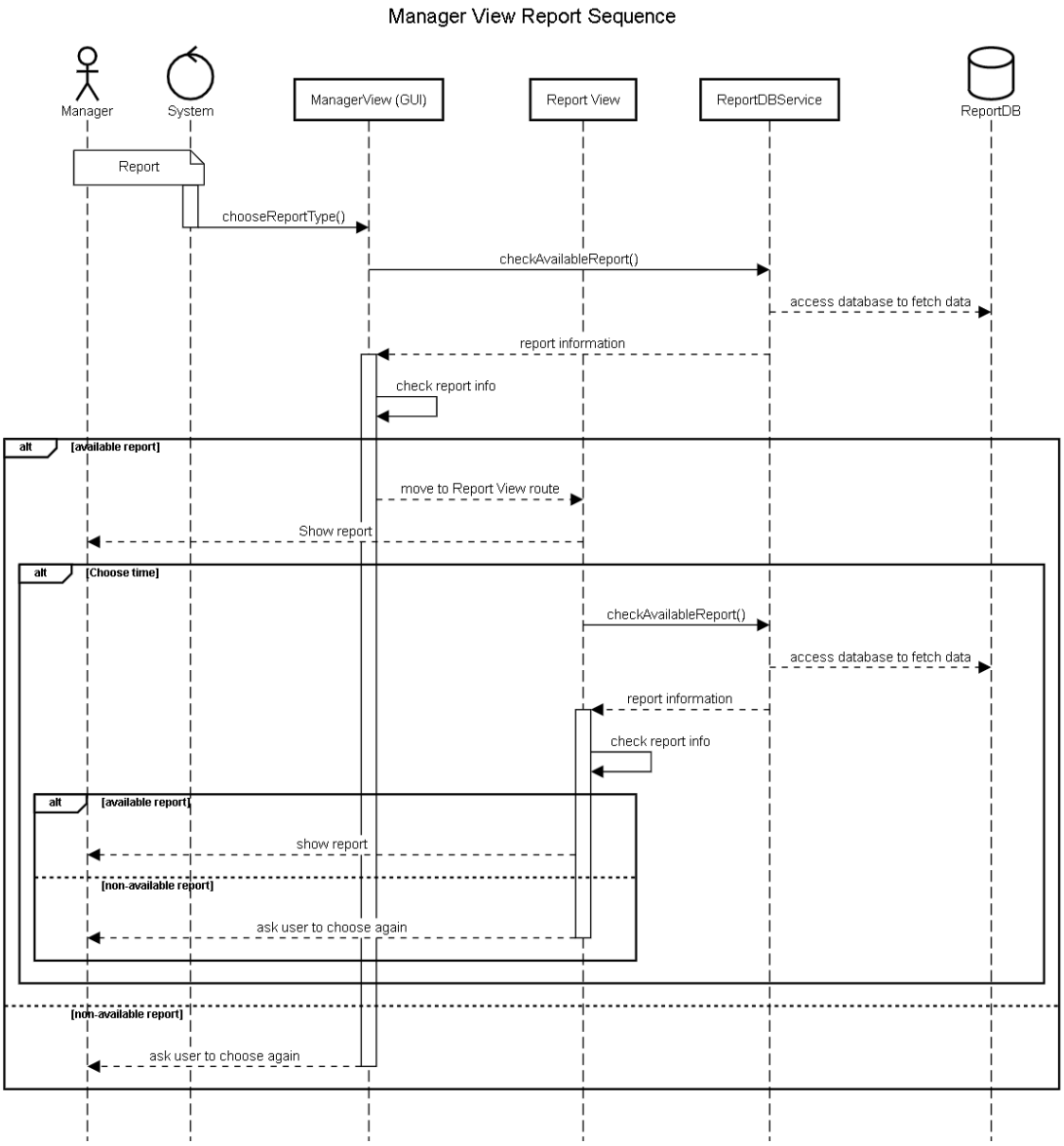


Figure 57: Sequence Diagram for View Report

9.3.3.8 Component Diagram

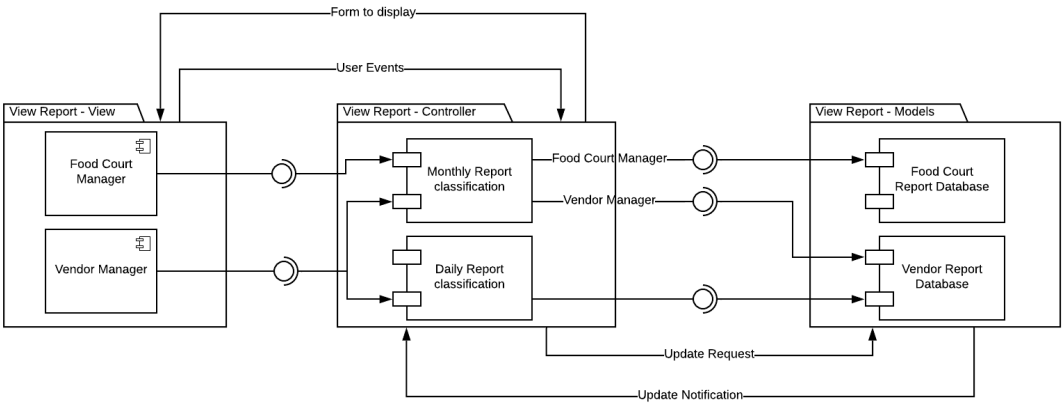


Figure 58: Component Diagram for View Report

9.3.4 Generate Report

9.3.4.1 User Story

The Food Court and Vendor Managers expect the reports to be auto-generated on time and include all required details depend on different types.

a. Vendor:

- Daily Report: At the beginning and After opening hour on the working day, the daily report of each vendors is generated and completed.
- Monthly Report: At the end of the month, after the opening hour the monthly report of each vendors is auto-generated by the system. This task must be completed before 22:00 PM.

b. Food Court:

- Monthly Report: At the end of the month, after 22:00 PM, the monthly report of the food court is auto-generated by the system.

9.3.4.2 Sequence Diagram

1. Generating Daily Report

- Description: The report is generated when the first staff running the app on the working day. After that the report keeps updating when the staffs finish the orders from the customers
- Diagram:

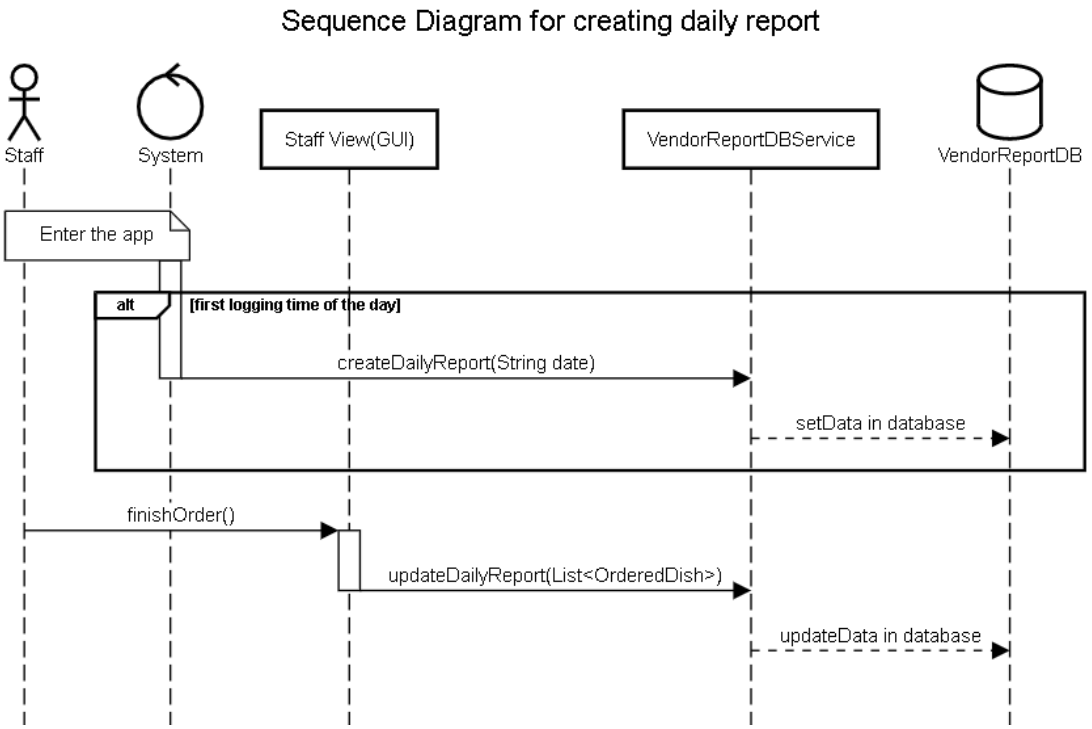


Figure 59: Sequence Diagram for Generating Daily Report

2. Generating Monthly Report

   - Description: This task is run frequently in the background of the device. At some specific time the monthly report is auto-generated
   - Diagram:

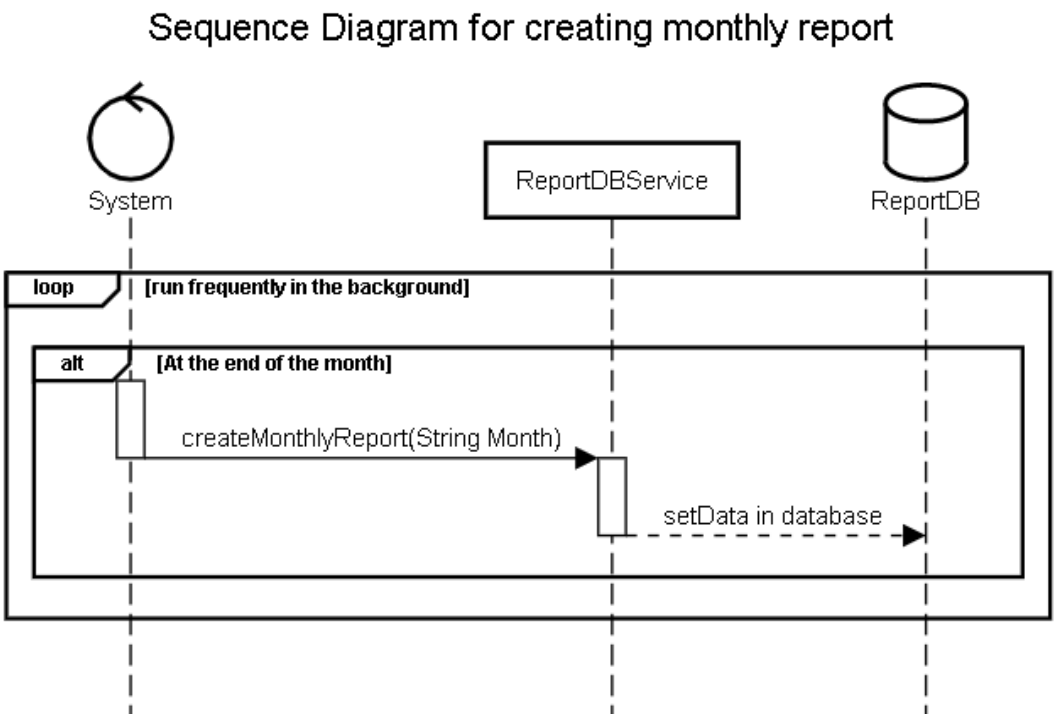


Figure 60: Sequence Diagram for Generating Monthly Report

9.4 View list

9.4.1 Use-case diagram

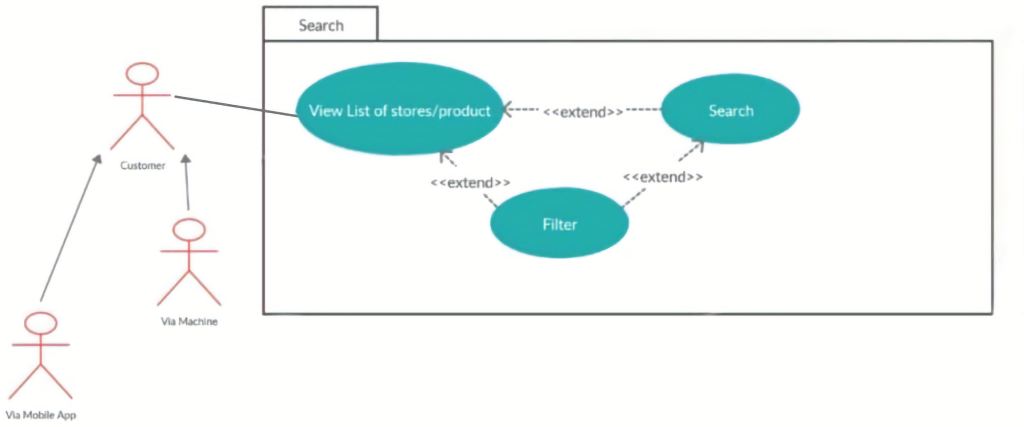


Figure 61: Use case diagram for View list feature

9.4.2 Class diagram

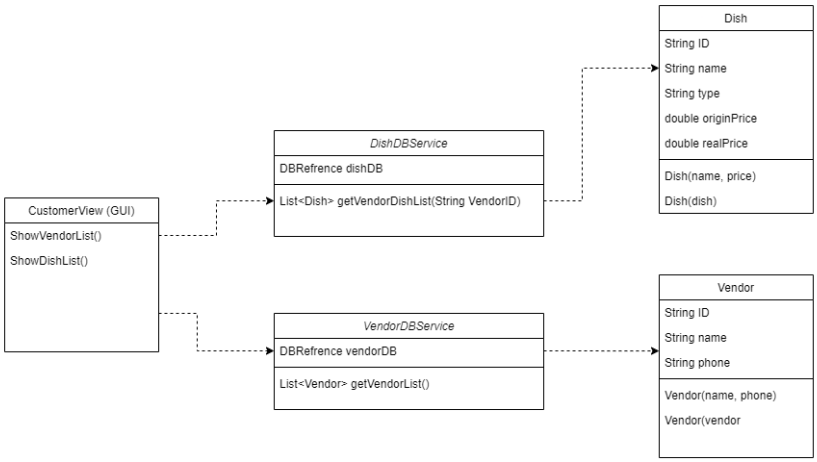


Figure 62: Class diagram for View list feature

9.4.3 Method description

- Class DishDBService

Method Name	Description
getVendorDishList	- Get all dish of that vendor ID from dishDB - Return as a List<Dish>

- Class VendorDBService

Method Name	Description
getVendorList	- Get all vendor from vendorDB - Return as a List<Dish>

9.4.4 User Story

As a customer, he/she will want to see all the vendor that are in the Food Court and the menu of each vendor

9.4.5 Main-flow

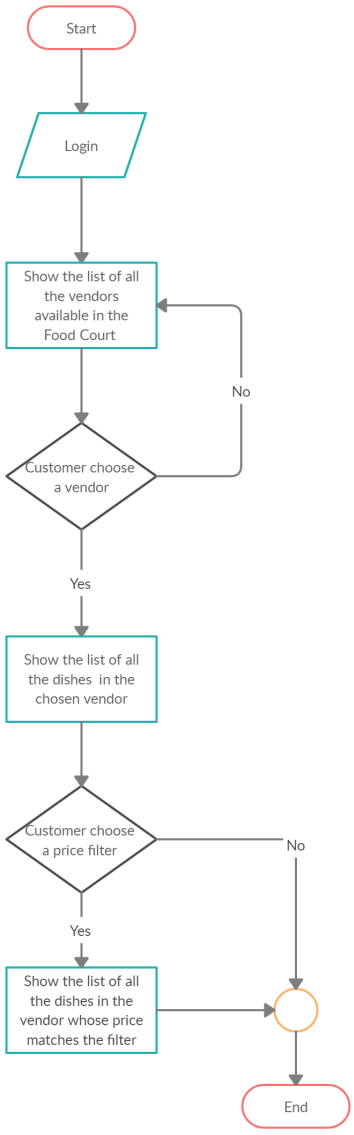


Figure 63: Flowchart for View list feature

9.4.6 Use-case detail/scenario

Use case ID	1
Use case name	View list

Actor	Customer
Description	Customers will be provided a list of available vendors as well as their dishes
Trigger	Customers open the app
Preconditions	Customers have to log in to the app
Normal flow	1. Customers log in to the app 2. The app shows a list of all the vendors available in the Food Court 3. Customers click on a vendor 4. The app shows a list of all the dishes of that vendor
Exceptions	Exception in step 4: a. If a dish is out of order, it will be grey-ed out and the customer cannot add it to the cart
Alternative flow	Alternative in step 3: a. If a customer hasn't logged in, the app will display the log in/ sign in interface

9.4.7 Mock-up

Figure 64: Home screen when customers login to the app

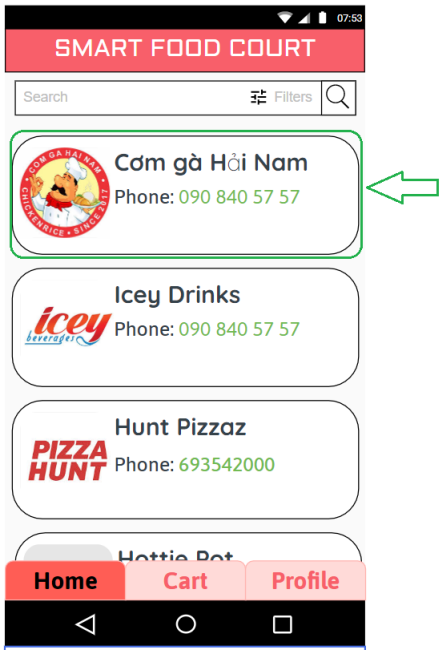


Figure 65: Click a vendor from the list to view menu of that vendor

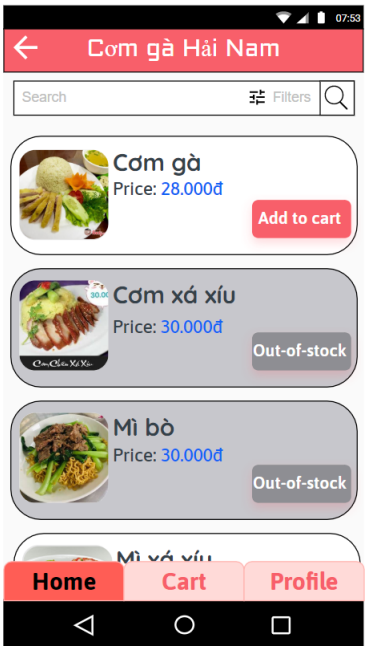


Figure 66: The app will display the menu of the chosen vendor

9.5 Search and Filter feature

9.5.1 Use-case diagram:

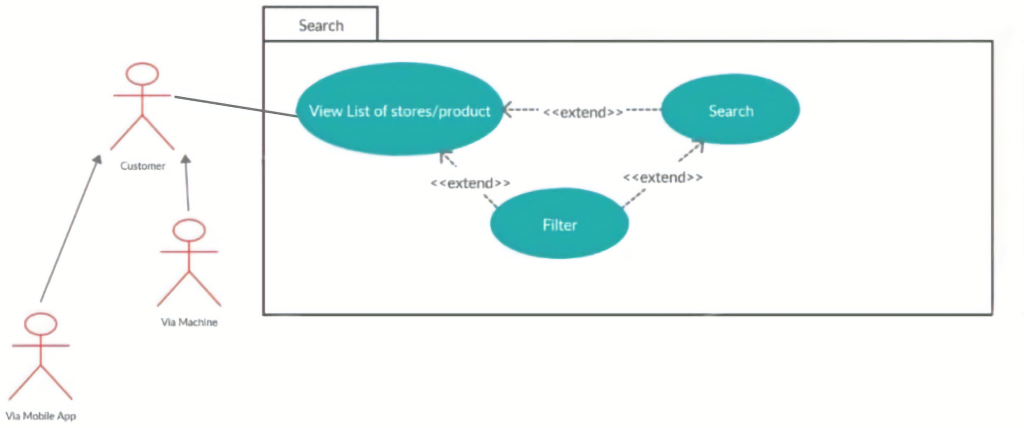


Figure 67: Use case diagram for Search with a filter feature

9.5.2 Class diagram

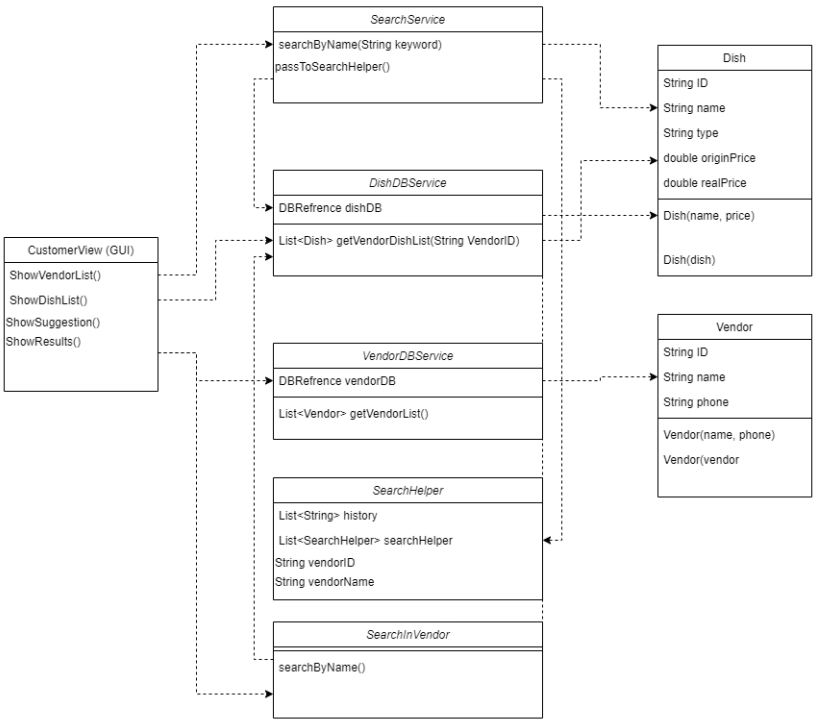


Figure 68: Class diagram for Search with a filter feature

9.5.3 Method description

- Class DishDBService

Method Name	Description
getVendorDishList	- Get all dish of that vendor ID from dishDB - Return as a List<Dish>

- Class VendorDBService

Method Name	Description
getVendorList	- Get all vendor from vendorDB - Return as a List<Dish>

- Class SearchService

Method Name	Description
searchByName	- Get a list of all Dishes in the Food Court whose name contains the keyword - Return as a List<Dish>
passToSearchHelper	Pass the vendorId and vendorName corresponding to a dish to the searchHelper list in order to display the search result correctly

- Class SearchInVendor

Method Name	Description
searchByName	Return a List<Dish> of the vendor whose name contains the keyword

9.5.4 User Story

As a customer, he/she may want to search for a specific dish to make an order, then the app will return the search results based on the keywords that the customer has entered. The customers can also choose the filters such as pricing to narrow the search results. When viewing the menu of a vendor, the customer can also choose a filter category to narrow down the list of dishes that meets his/her needs

9.5.5 Main-flow

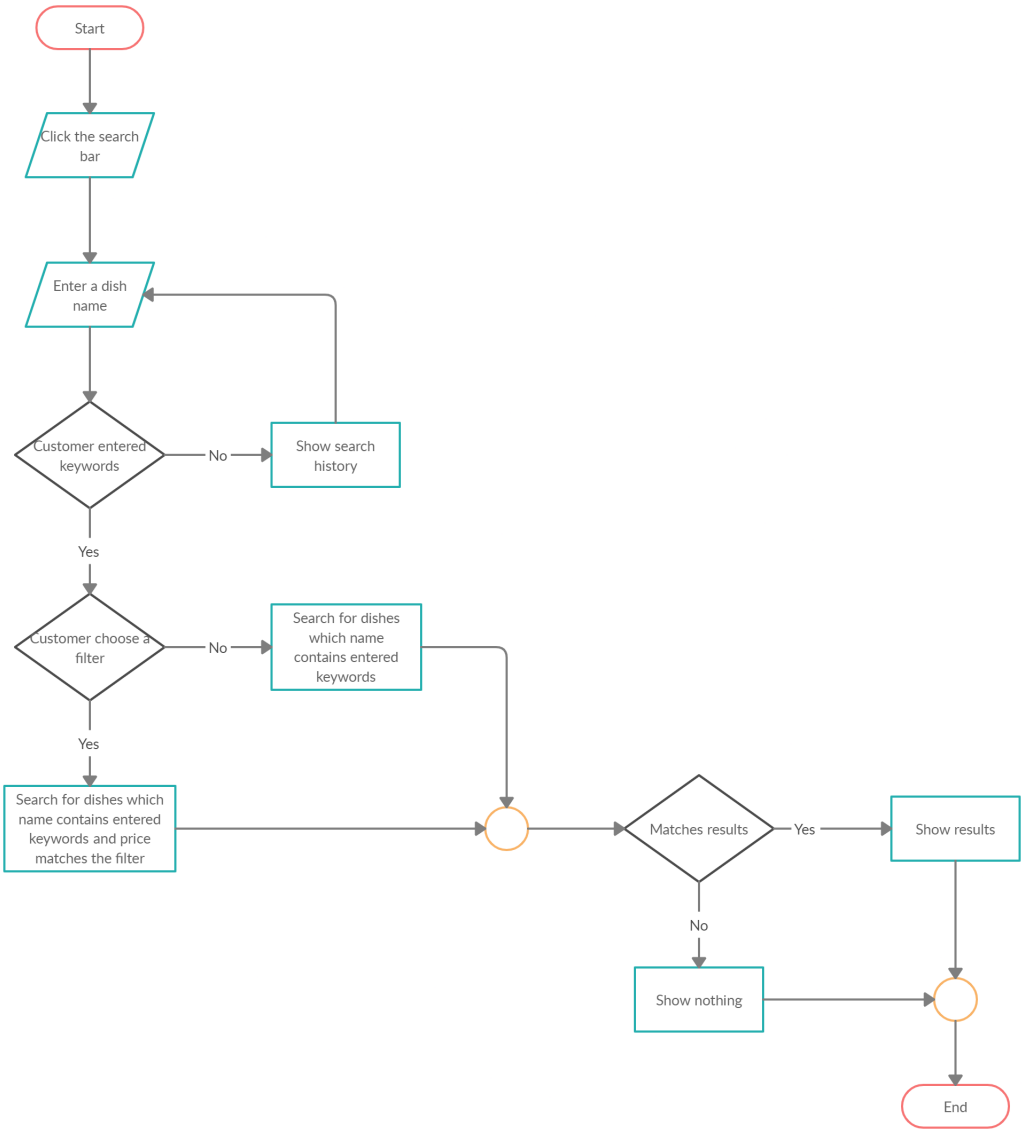


Figure 69: Flow chart for Search in Home screen

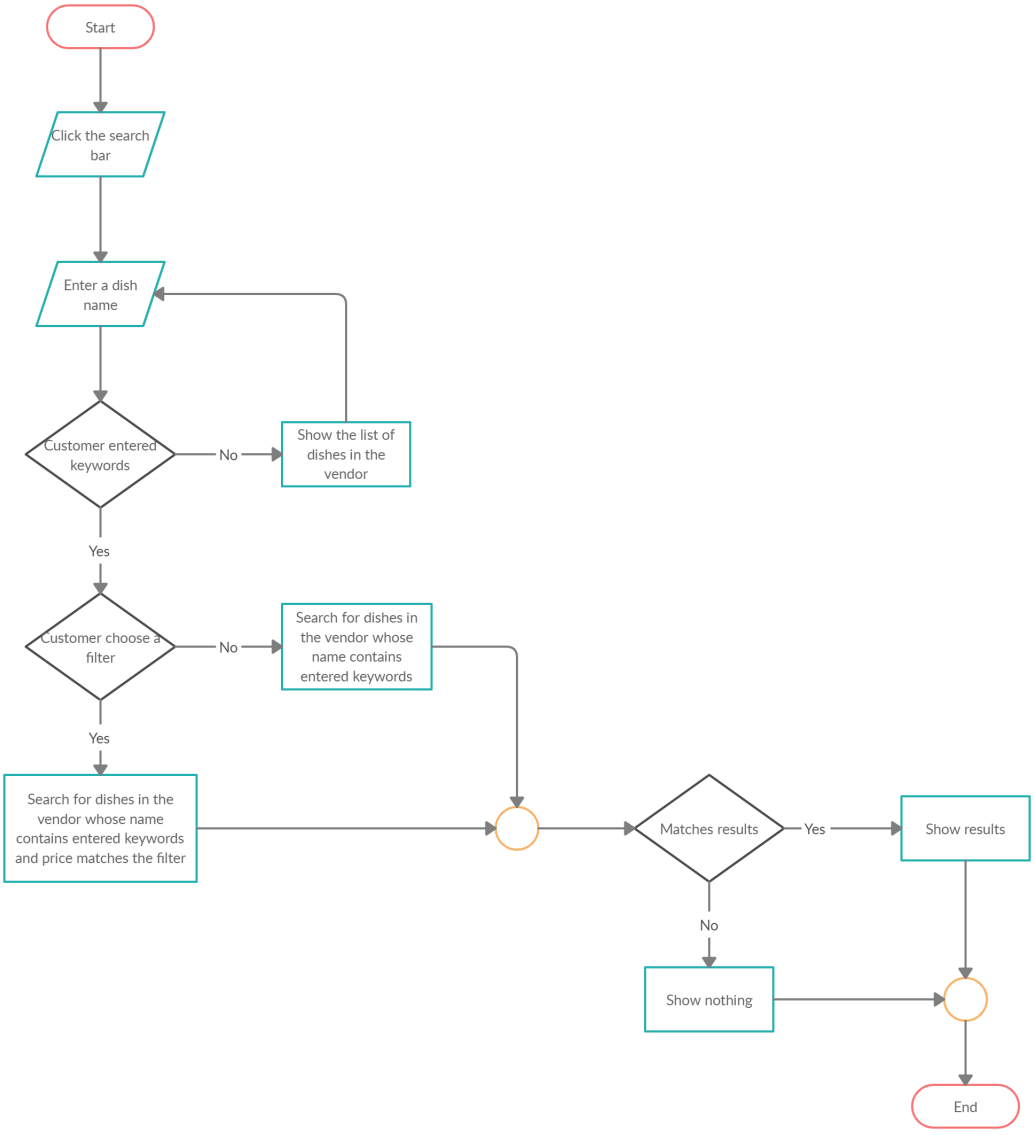


Figure 70: Flow chart for Search in a Vendor

9.5.6 Use-case detail/scenario

Use case ID	2
Use case name	Search with a filter function
Actor	Customer
Description	Customers can search for a specific dish and they can use filters to narrow down the search results

Trigger	Customers click on the search bar
Preconditions	Customers have to log in to the app
Normal flow	1. Customers click on the search bar 2. Customers enter search keywords 3. The app shows a list of dishes whose name contains search keywords 4. Customers choose a dish they want to order from 5. The app shows the menu of the vendor contains that dish 6. Customers add the dish to cart and make an order
Exceptions	Exception in step 2: a. The customer doesn't enter a name in the search bar b. The app shows nothing Exception in step 3: a. If there is no result matches the keywords and the filters, the app will display a not found message
Alternative flow	Alternative in step 3: a. The customer uses the filters to narrow the search results b. The app will show the search results based on the keywords and the filters Alternative in step 4: a. The customer doesn't choose a dish b. The customer can add a dish straight to cart without going through the vendor menu Alternative in step 2: If the customer is viewing a vendor menu a. The app shows a list of dishes of that vendor b. The customer enter search keyword c. The app shows a list of dishes whose name contains the search keyword d. The customer can click the "Add to cart" button to add a dish to cart

9.5.7 Mock-up

Home Screen

Figure 71: Default screen when customers login to the app

Search

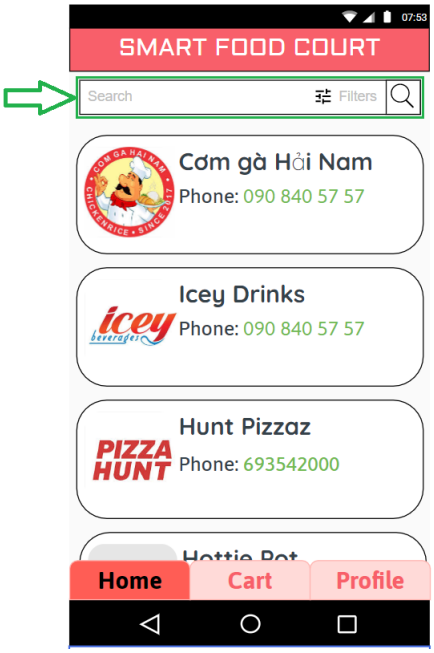


Figure 72: Click the search bar to search

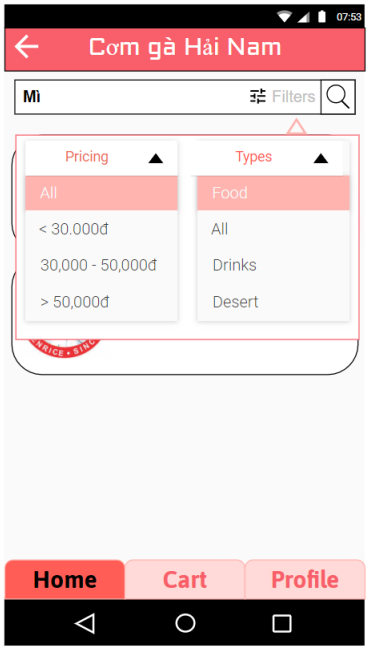


Figure 73: Choose filters to narrow down the search results

9.6 Order Management

This is the use-case set of Order Management composed of:

6.A. Show order queue

6.B. Inform ready order

6.C. Finish order

6.D. Notify product out of stock

Use-case diagram:

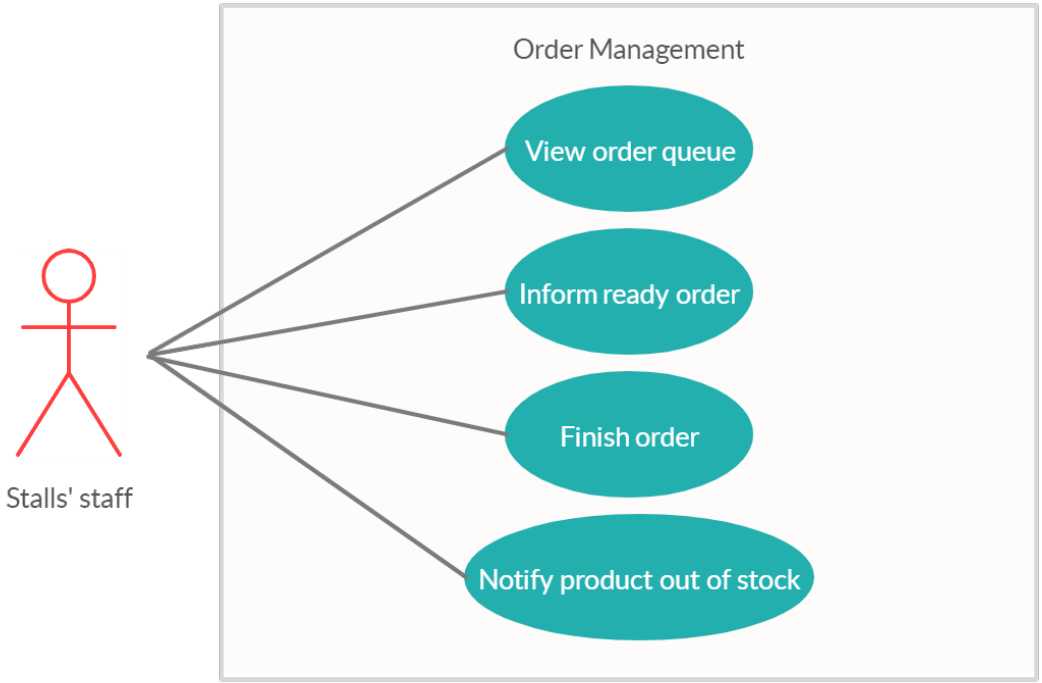


Figure 74: Use case diagram for Order Management

Class diagram:

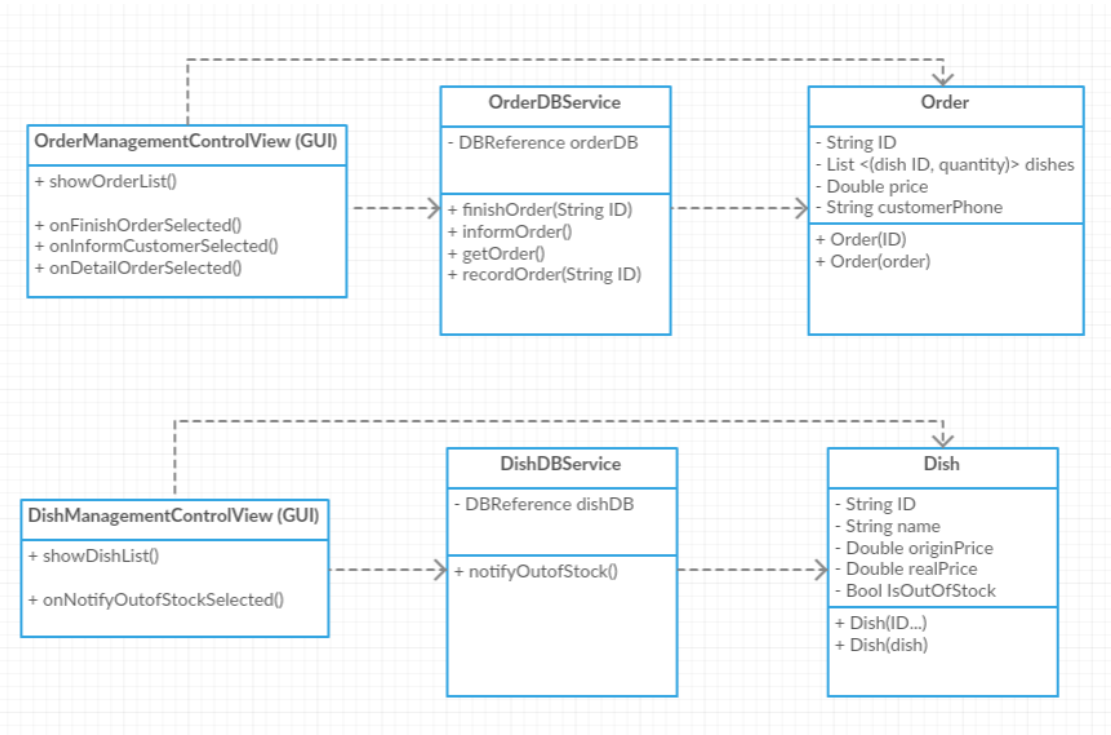


Figure 75: Class diagram for Order Management

9.6.1 Show order queue

9.6.1.1 User story - As a vendors' staff, I want to view customer's order so that I can know what food I should make to serve customer.

9.6.1.2 Main flow

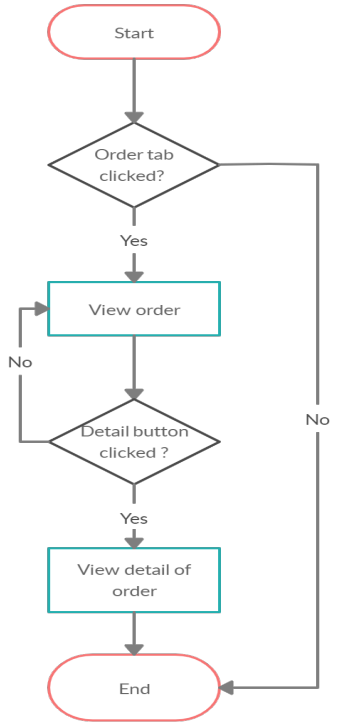
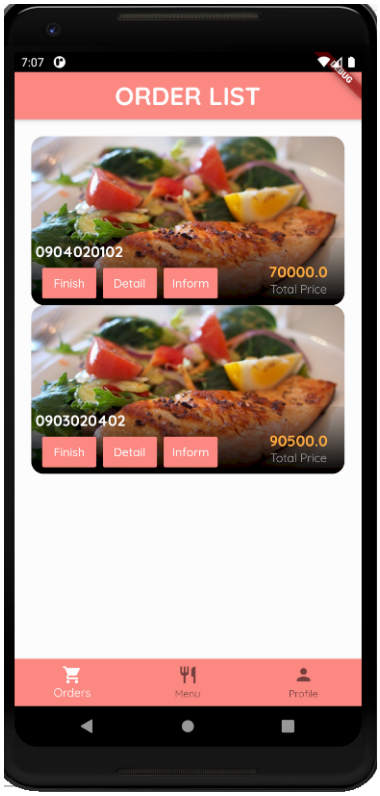


Figure 76: Flow chart for View order queue of vendors' staff

9.6.1.3 Use-case detail/scenario

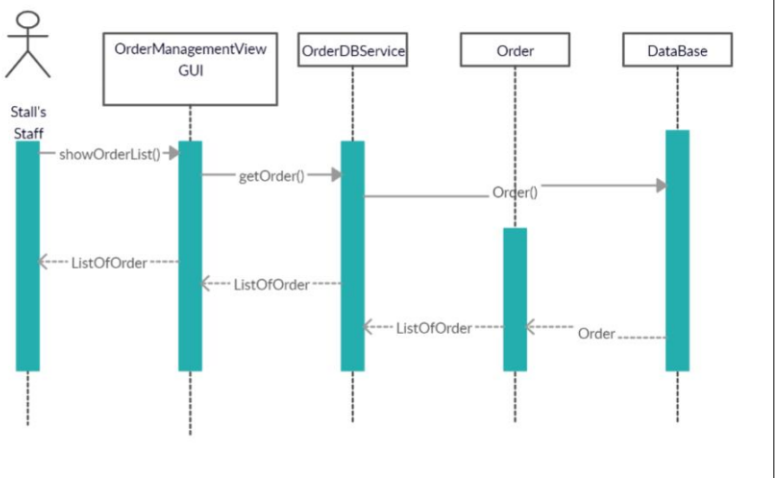
Use case ID	6.A
Use case name	View order queue.
Actor	vendors' staff.
Description	Show staff the order queue so that they can manage and accept the order.
Precondition	Staff is using their specified interface and click on the Order tab.
Normal flow	1. From the main menu, staff click on the "Order" tab. 2. Then the list of order will appear. 3. Staff click on order to view detail.
Exceptions	1. Exception 1: at step 1: If staff does not click on the Order tab staff can not view order. 2. Exception 2: If staff does not click on the Detail button staff can not view detail of order.
Alternative flow	Alternative 1: at step 2: If there is no order. Nothing will be displayed.

9.6.1.4 Mock-up



9.6.1.5 Sequence diagram

The sequence diagram for view order



9.6.2 Inform ready order

9.6.2.1 User story

As a vendor's staff, I want to inform customer when the food they ordered is ready so they can come to pick it up.

9.6.2.2 Main flow

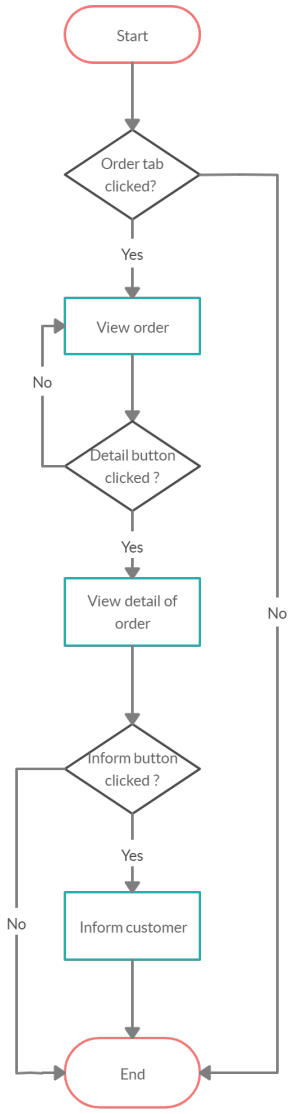
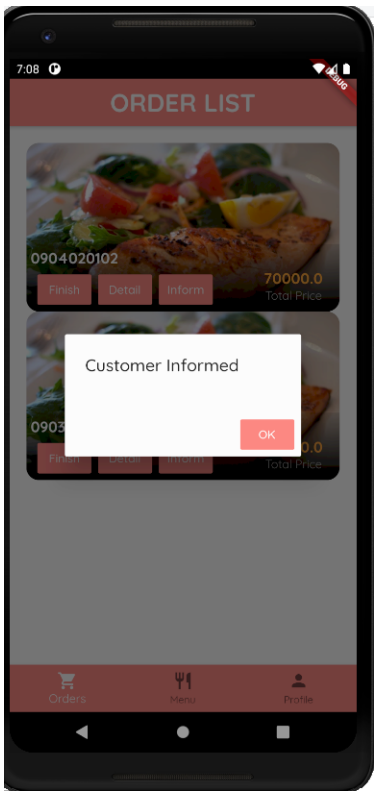


Figure 77: Flow chart for Inform ready order of vendors' staff

9.6.2.3 Use-case detail/scenario

Use case ID	6.B
Use case name	Inform an order is ready.
Actor	vendors' staff.
Description	Inform the customer that the dishes is ready.
Preconditions	The order is in the queue above and ready to be served.
Normal flow	1. From the main menu, staff click on the "Order" tab. 2. In the "Order" tab, staff click on the "Detail" button of order which is ready to be served. 3. Click on "Inform" to send a notification to the customer.
Exceptions	Exception 1: Staff can use the phone number provided to inform customer by message or call. .
Alternative flow	Alternative 1: at step 3: The notification will be sent automatically again after an amount of time if the customer has not picked up.

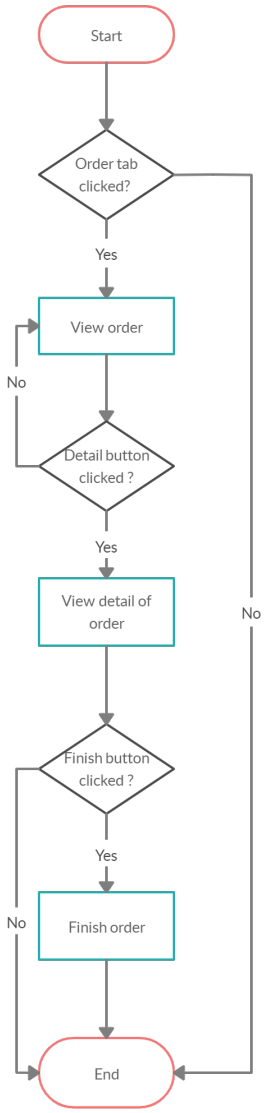
9.6.2.4 Mock-up



9.6.3 Finish order

9.6.3.1 User story - As a vendor's staff, I want to clear any order that has been picked up by the customer out of the order list.

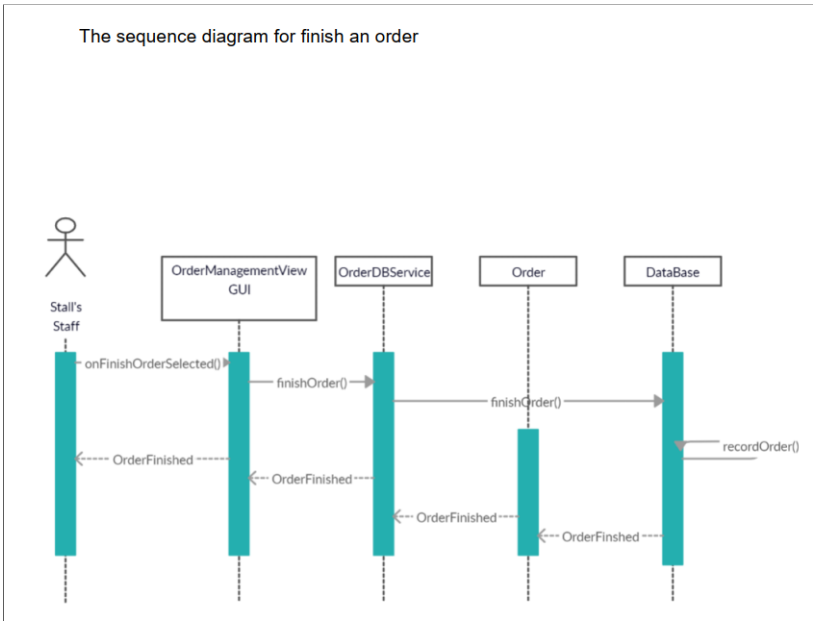
9.6.3.2 Main flow



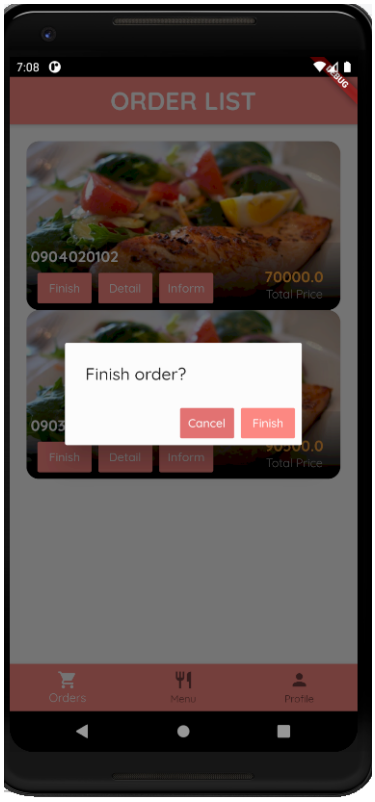
9.6.3.3 Use-case detail/scenario

Use case ID	6.C
Use case name	Finish an order
Actor	vendors' staff
Description	Finish a delivered order and remove it from the order's list.
Preconditions	The customer has taken the food.
Normal flow	1. From the main menu, staff click on the "Order" tab. 2. In the "Order" tab, staff click on the order when the customer come to pick up the food. 3. Staff check customer information. Then deliver food to customer and tap "Finish". 4. Finished order will disappear from the order queue.
Exceptions	Exception 1: at step 2: If staff does not tap the "Detail" button, staff can not finish the order
Alternative flow	Alternative 1: Staff can click finish to destroy an order which the customer does not come to pick up after a long time

9.6.3.4 Sequence diagram



9.6.3.5 Mock-up



9.6.4 Notify product out-of-stock

9.6.4.1 User story - As a vendor's staff, I want to inform customer that some product is currently out-of-stock.

9.6.4.2 Main flow

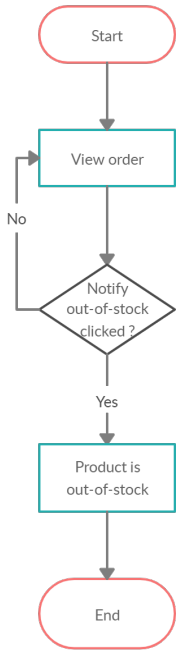


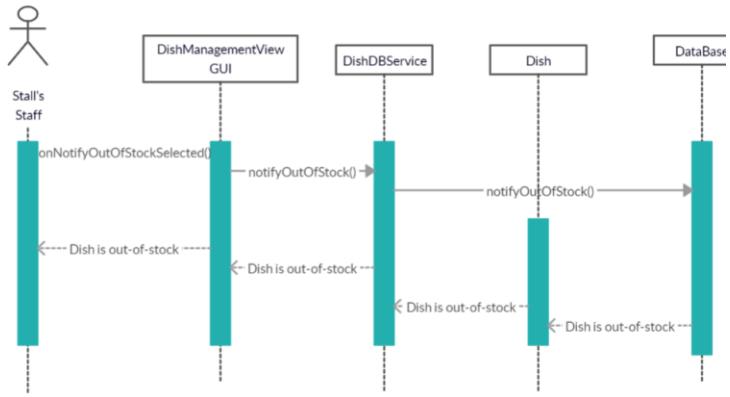
Figure 78: Flow chart for notify product is out-of-stock of vendors' staff

9.6.4.3 Use-case detail/scenario

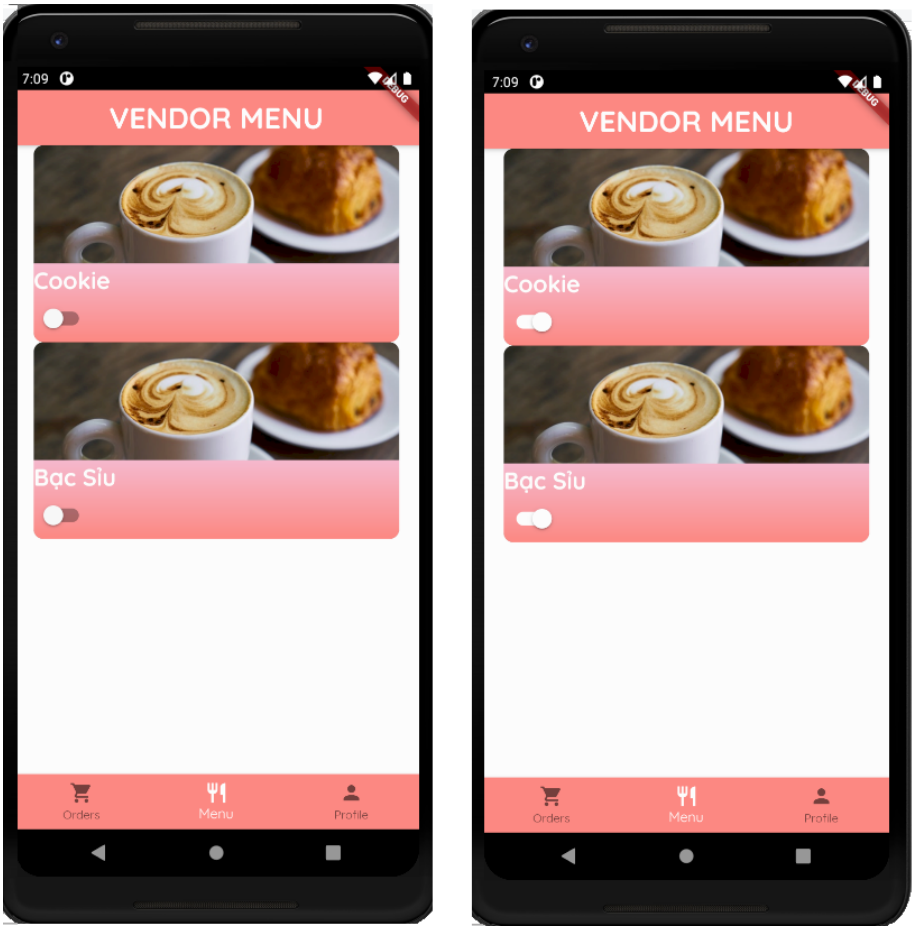
Use case ID	6.D
Use case name	Out of order
Actor	vendors' staff
Description	Inform the system that a product is currently out of order.
Preconditions	Staff needs to enter the product interface.
Normal flow	1. In the product interface, staff looks for the product that is out of stock. 2. He then clicks on the switch on the product status to change it to "out of order" state. 3. The system will inform the customer whenever they order an "out of order" product.
Exceptions	Exception 1: at step 2: If staff does not tap the button, product is still can be ordered by the customer
Alternative flow	Alternative 1: at step 2: The product will stay in this state until the staff switch it back to normal.

9.6.4.4 Sequence diagram

The sequence diagram for notify food is out-of-stock



9.6.4.5 Mock-up



10 Conclusion

The system now can serve the basic uses that needed for a simple Smart Food Court. For future prospects, we are looking for tracking things such as history orders of customer to provide suggestion feature. The feature that notify customer ready order also need to be improved.