Tran Dinh Quy PROJECT NAME: BUILDING A SYSTEM FOR ORDERING FOODS AND BEVERAGES PROMPTLY 2019

THE UNIVERSITY OF DA NANG
DA NANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY
SPECIALTY: SOFTWARE ENGINEERING

PROJECT TITLE:

BUILDING A SYSTEM FOR ORDERING FOODS AND BEVERAGES PROMPTLY

Instructor	**: M.S. NGUYEN THI LE QUYEN**
Student	**: TRAN DINH QUY**
Student ID	**: 102150065**
Class	**: 15T1**

Da Nang, 12/2019

THE UNIVERSITY OF DA NANG
DA NANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: SOFTWARE ENGINEERING

PROJECT TITLE:

BUILDING A SYSTEM FOR ORDERING FOODS AND BEVERAGES PROMPTLY

Instructor	**: M.S. NGUYEN THI LE QUYEN**
Student	**: TRAN DINH QUY**
Student ID	**: 102150065**
Class	**: 15T1**

Da Nang, 12/2019

INSTRUCTOR'S COMMENTS

REVIEWER'S COMMENTS

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

SUMMARY

Project name: Building a system for ordering foods and beverages promptly.

Student: Tran Dinh Quy

Student ID: 102150065

Class: 15T1

The aim of this project was to create a system that helps everyone can place orders easily. Furthermore, shop owners can use this system to control their sale. Especially, it also creates a shipper community to deliver the order.

I have done sufficiently from the conception to design of this work, as well as the analysis and implementation of the project. This system is divided into 2 parts, one for the website and one for the mobile phone:

About the website, I used ReactJS and Bootstrap framework. I used Android Studio with Kotlin for the mobile application.

Both of them connect together through an API, which is built by Node.js and Express framework, and MySQL for designing the database.

Some other technologies like Firebase for logging with Google Account, AWS S3 for saving pictures, Socket.io for realtime handling.

DA NANG UNIVERSITY

UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom – Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: Tran Dinh Quy Student ID: 102150065

Class: 15T1 Faculty: Information Technology Major: Information Technology

1. *Topic title:* Building a system for ordering foods and beverages promptly
2. *Project topic:* ☐ *has signed intellectual property agreement for the final result*
3. *Initial figure and data:*

   Data were collected from the internet.

Content of the explanations and calculations:

- Place orders and handle orders
- Control dishes, accounts
- Recommendation system

4. *Drawings, charts (specify the types and sizes of drawings):*

- Use case diagram
- Activity diagram
- Class diagram
- Package diagram

5. *Name of instructor:* M.S. Nguyen Thi Le Quyen
6. *Date of assignment:* *28/8/2019*
7. *Date of completion:* *14/12/2019*

Da Nang, 20th December, 2019

Head of Division…………………. **Instructor**

PREFACE

I would first like to thank my instructor, M.S. Nguyen Thi Le Quyen for her continuous support, supervision, motivation, and guidance throughout the tenure of my project in spite of her hectic schedule. She remained a driving spirit in my project and her experience gave me the understanding of handling research projects as well as helping me to clarify the abstruse concepts, requiring knowledge and perception, handling critical situations and understanding the objective of my work.

In addition, I would like to thank my mentor, Mr. Nguyen Luong Hoang from Enclave company for his valuable guidance. He definitely provided me with the tools that I needed to choose the right direction and successfully complete my senior project.

I also want to thank my family and friends, who gave me the strength and confidence during my time of learning and during the implementation phase of this project. They have given a lot of love and encouragement for me which helped pass over the difficulties and fatigues.

Without their generous help, my senior year would not have been successful.

Sincerely,

Tran Dinh Quy

ASSURRANCE

I guarantee:

1. The contents of this senior project are performed by myself following the guidance of instructor M.S. Nguyen Thi Le Quyen.

2. All references used in this senior project thesis, are quoted with the author's name, project name, time and location to publish clearly and faithfully.

3. All invalid copies, educated statute violation or cheating will be borne the full responsibility by myself.

Student,

Tran Dinh Quy

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

No	Items	Description
1	API	Application Programming Interface
2	HTML	Hypertext Markup Language
3	HTTP	Hypertext Transfer Protocol
4	REST	Representational State Transfer
5	UX/UI	User Experience/User Interface
6	AWS	Amazon Web Services

INTRODUCTION

1. Project overview

1.1. Context

Nowadays, the internet appears everywhere, people have a lot of chances to approach 4G and mobile phone. Besides that, I recognize that in my hometown, the number of coffee shops is increasing very fast. But there is no delivery service which is similar to Grab food or Now.

I think current services have some issues that we can resolve to apply to some small place like my home town, such as houses don't have numbers on their addresses, shop cannot confirm whether that order is available or not, waste time that we can reduce when shipper go to the shop.

Because of those problems mentioned above, I decided to choose the project "Building a system for ordering foods and beverages promptly".

1.2. Purpose

The system has been created to assist bring a way to the customer can get dishes, beverages effortlessly and helps the shop can manage orders. Moreover, the system will be implemented with a recommender system for the user.

As an admin, you are a responsible person for managing accounts and shops.

About shop owner, they can access the system for managing their own shop with dishes and orders. They can get notification about orders, confirm if they accept real–timely.

Shipper will receive order and manage that order. The system will show all the information about that order and show routes.

Customer only needs to log in the system by Google account and place order. The system will show some recommended dishes for each user.

2. Theories

2.1. Technologies

- Node.js and Express Framework
- Sequelize
- ReactJS and Bootstrap Framework
- Android with Kotlin
- MySQL

2.2. Algorithm

- Matrix Factorization

2.3. Tools and environment development

- Microsoft Visual Studio Code
- MySQL Workbench
- Android Studio
- Postman

3. Structure of the thesis

INTRODUCTION – This chapter gives information about the context and purpose of the project as well as giving the scope of the problems which will be focused on the thesis.

Chapter 1: THEORIES AND TECHNOLOGIES – This chapter introduces all knowledge theories and technologies used in this project.

Chapter 2: ANALYSIS AND DESIGN – This chapter covers the main features, software requirement specifications and database design of the project.

Chapter 3: IMPLEMENTATION AND RESULT EVALUATION – This chapter shows an implementation of this project, including pictures and a brief explanation for each main function.

CONCLUSION – The concluding section of the project simultaneously emphasizes the problem solved, as well as presenting issues still unresolved and provides recommendations and suggestions.

REFERENCES – Presentation about the detail of the referenced information used in this thesis.

Chapter 1 : THEORIES AND TECHNOLOGIES

This system applies the current popular technologies and platforms that combine to create a system responds to the services that user desire.

To learn more about the technologies, I would like to present the concept of some key technologies that I used in this project.

1.1. Node.js

Node.js is an open–source, cross–platform JavaScript run–time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command–line tools and for server–side scripting running scripts server–side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server–side and client–side scripts.

Though ".js" is the standard filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event–driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real–time Web applications (e.g., real–time communication programs and browser games).

The Node.js distributed development project, governed by the Node.js Foundation, is facilitated by the Linux Foundation 's Collaborative Projects program.

– **Features:**

Extremely fast: Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.

I/O is Asynchronous and Event–Driven: All APIs of Node.js library are asynchronous i.e. non–blocking. So a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.

Single threaded: Node.js follows a single–threaded model with event looping.

Highly Scalable: Node.js is highly scalable because the event mechanism helps the server to respond in a non–blocking way.

No buffering: Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.

Open source: Node.js has an open–source community that has produced many excellent modules to add additional capabilities to Node.js applications.

1.2. ReactJS

ReactJS is a JavaScript–based open–source front–end web application platform led by Facebook and by a community of individuals and corporations to address all of the parts of the developer's workflow while building complex web applications.

– **Features and benefits:**

Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep the state out of the DOM.

Learn Once, Write Anywhere: You can develop new features in React without rewriting existing code.

Productivity: ReactJS helps us quickly create UI views with simple and powerful template syntax as well as supports command–line tools and intelligent code completion for man IDEs.

1.3. MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

– MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

1.4. Android Studio and Kotlin

1.4.1. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

1.4.2. Kotlin

Kotlin is officially supported by Google for mobile development on Android. Since the release of Android Studio 3.0 in October 2017, Kotlin is included as an alternative to the standard Java compiler. The Android Kotlin compiler lets the user choose between targeting Java 6 or Java 8 compatible bytecode.

– **Features and benefits:**

Drastically reduce the amount of boilerplate code.

Avoid entire classes of errors such as null pointer exceptions.

Leverage existing libraries for the JVM, Android, and the browser.

1.5. Socket.io

Socket.IO is a JavaScript library for realtime web applications. It enables realtime, bi–directional communication between web clients and servers. It has two parts: a client–side library that runs in the browser, and a server–side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event–driven.

Socket.IO primarily uses the WebSocket protocol with polling as a fallback option, while providing the same interface. Although it can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.

Socket.IO is not a WebSocket library with fallback options to other real–time protocols. It is a custom real–time transport protocol implementation on top of other real–time protocols. A Socket.IO implementing server cannot connect to a non–Socket.IO WebSocket client. A Socket.IO implementing client cannot talk to a non–

Socket.IO WebSocket or Long Polling Comet server. Socket.IO requires using the Socket.IO libraries on both client and server–side.

– **Features and benefits:**

Socket.IO provides the ability to implement real–time analytics, binary streaming, instant messaging, and document collaboration. Notable users include Microsoft Office, Yammer, and Zendesk.

Socket.IO handles the connection transparently. It will automatically upgrade to WebSocket if possible. This requires the programmer to only have Socket.IO knowledge.

As of version 2.0, Socket.IO makes use of µWebSockets as the underlying WebSocket library.

1.6. Amazon Web Services

The AWS technology is implemented at server farms throughout the world and maintained by the Amazon subsidiary. Fees are based on a combination of usage, the hardware/OS/software/networking features chosen by the subscriber, required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either. As part of the subscription agreement, Amazon provides security for the subscriber's system. AWS operates from many global geographical regions including 6 in North America.

– Features and benefits:

Amazon Elastic Compute Cloud or AWS EC2 provides scalable computing capacity in the Amazon Web Services (AWS) cloud.

Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e–commerce network. Amazon S3 can be employed to store any type of object which allows for uses like storage for Internet applications, backup and recovery, disaster recovery, data archives, data lakes for analytics, and hybrid cloud storage. In its service–level agreement, Amazon S3 guarantees 99.9% uptime, which works out to less than 43 minutes of downtime per month. AWS launched Amazon S3 in the United States on March 14, 2006, then in Europe in November 2007.

1.7. Firebase

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps.

– Features and benefits:

Firebase has a lot of features but in my project, I just use Firebase Auth.

Firebase Auth is a service that can authenticate users using only client–side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

1.8. Matrix factorization algorithm

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user–item interaction matrix into the product of two lower dimensionality rectangular matrices. This family of methods became widely known during the Netflix prize challenge due to its effectiveness as reported by Simon Funk in his 2006 blog post, where he shared his findings with the research community.

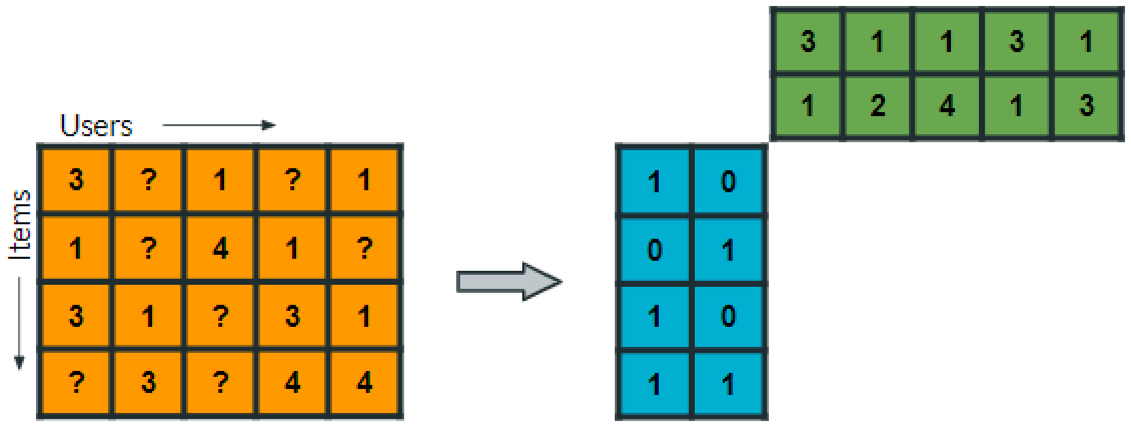


Figure 1.1: Matrix factorization algorithm

Algorithm:

Rules:

i: number of items

u: number of users

Step 1: Build a score matrix between user and item. Missing cells are filled "0". Init 2 matrixes $P(K\ x\ i)$ and $Q(u\ x\ K)$, fill out random numbers. In my system, because the rating score is from 1 to 5 so I choose random numbers in that range.

Step 2: Record coordinates of available cells and their scores.

Step 3: Get one available cell to know the real value (r_{ij}). Multiple P and Q values to get predict score ($\widehat{r_{ij}}$) of that coordinates. Calculate error from real value. Modify P and Q values

Step 4: Repeat step 3 until the result is acceptable

The difference usually called the error between the estimated rating and the real rating can be calculated by the following equation for each user–item pair. Here we consider the squared error because the estimated rating can be either higher or lower than the real rating:

$$e^2 = (r_{ij} - \widehat{r_{ij}})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{ki})^2 \quad (2.1)$$

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately. Derivate 2 sides with p_{ik} we have:

$$\frac{\delta}{\delta_{p_{ik}}} e_{ij}^2 = -2\left(r_{ij} - \widehat{r_{ij}}\right)q_{kj} = -2e_{ij}q_{kj} \quad (2.2)$$

Derivate 2 sides with q_{kj} we have:

$$\frac{\delta}{\delta_{q_{kj}}} e_{ij}^2 = -2\left(r_{ij} - \widehat{r_{ij}}\right)p_{ik} = -2e_{ij}p_{ik} \quad (2.3)$$

Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj}:

Next p_{ik} is calculated with this formula:

$$p'_{ik} = p_{ik} + \alpha \frac{\delta}{\delta_{p_{ik}}} e_{ij}^2 = p_{ik} + -2\alpha e_{ij}q_{kj} \quad (2.4)$$

And similar to q_{kj}:

$$q'_{kj} = q_{kj} + \alpha \frac{\delta}{\delta_{q_{kj}}} e_{ij}^2 = q_{kj} + -2\alpha e_{ij} p_{ik} \quad (2.5)$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually, we will choose a small value for α, say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

Adding Biases:

When predicting the ratings of users given to items, it is useful to consider how ratings are generated. In the above discussion, we have assumed that ratings are generated based on matching the user’s preferences on some latent factors and the items $'$ characteristics on the latent factors. Actually, it may also be helpful to consider additional factors here.

For example, we can assume that when a rating is generated, some biases may also contribute to the ratings. In particular, every user may have his or her own bias, meaning that he or she may tend to rate items higher or lower than the others. In movie ratings, if a user is a serious movie watcher, he or she may tend to give lower ratings, when compared to another user who generally enjoys movies as long as they are not too boring. A similar idea can also be applied to the items being rated.

Hence, in the equal of predicting a rating, we can also add these biases in order to better model how a rating is generated:

$$\widehat{r_{ij}} = b + bu_i + bd_j + \sum_{k=1}^{K} p_{ik} q_{kj}$$

where bb is the global bias (which can be easily estimated by using the mean of all ratings), bu_i is the bias of user i, and bd_j is the bias of item j.

Using the same steps mentioned above, we can derive the update rules for the user biases and item biases easily:

$$bu'_i = bu_i + \alpha \times (e_{ij} - \beta bu_i)$$

$$bd'_j = bd_j + \alpha \times (e_{ij} - \beta bd_j)$$

The process of factorization will converge faster if biases are included in the model

Chapter 2 : ANALYSIS AND DESIGN

This chapter will go into detail the requirements, describing nonfunctional requirements, design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the application. This consists of a package containing Requirements Specification, Use–Cases of the use–case model, Use Case Specifications and Activity Diagram. Shows an overview of what functions the system can satisfy. In addition, it defines the architecture, modules, and data for a system to satisfy specified requirements. System design is intended to be the link between the system architecture and the implementation of technological system elements that compose the physical architecture model of the system. It could be seen as the application of systems theory to product development.

The System Design process is to provide sufficiently detailed data and information about the system and it is a system element to enable the implementation consistent with architectural entities as defined in models and views of the system architecture. It shows the components of the web application, the structure of data tables, the relationship the elements that make up the system.

2.1. Requirement analysis

2.1.1. General requirements

Customer needs to place order promptly. The system will show some recommended dishes for each user.

As an admin, you are a responsible person for managing accounts and shops.

About shop owner, they can access the system for managing their own shop with dishes and orders. They can get notification about orders, confirm if they accept real–timely.

Shipper will receive order and manage that order. The system will show all the information about that order and show routes.

2.1.2. Main actors

This system has four main actors:

- Administrator
- Shop Owner
- Customer
- Shipper

2.2. System analysis

2.2.1. System architecture

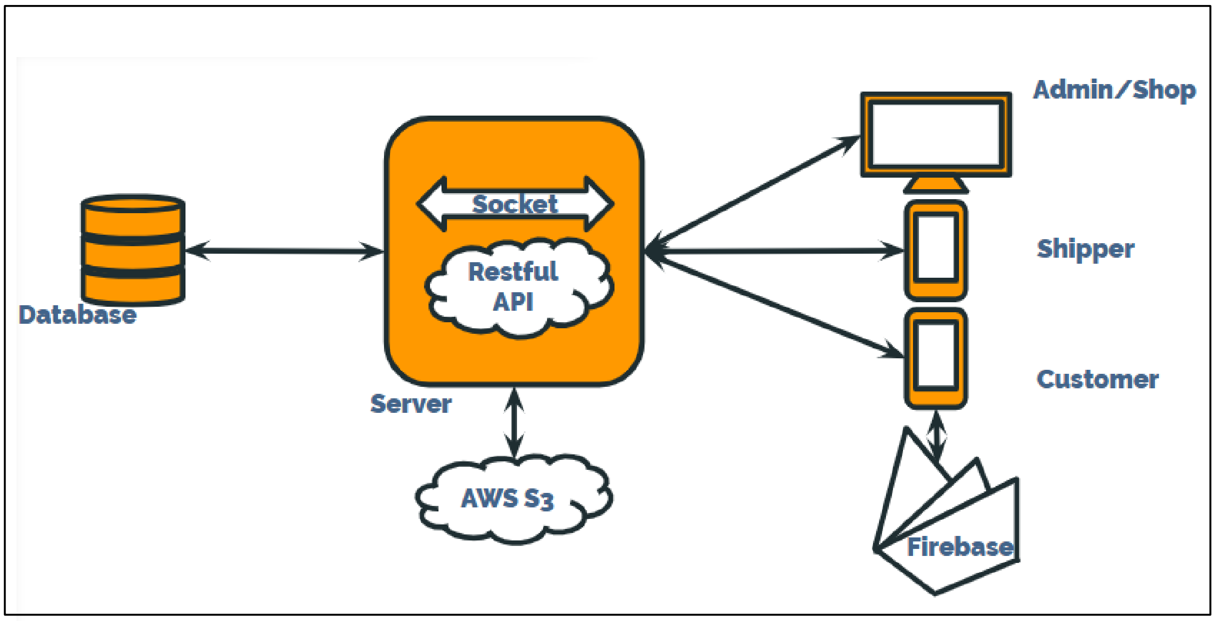


Figure 2.1: System architecture

This system includes:

API Service: I take Node.js / Express Framework as the main basic, combining with MySQL under representational state transfer (REST) technology.

2.2.2. Use case diagram

1) Use case diagram for Administrator

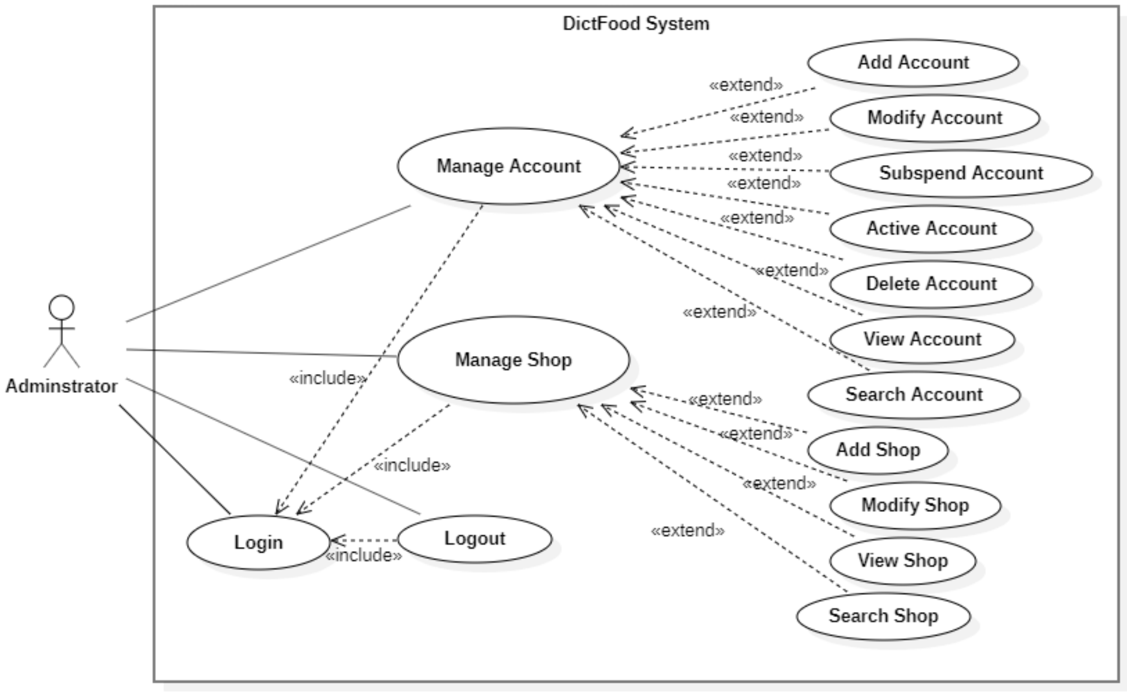


Figure 2.2: Use case diagram – Administrator

2) Use case diagram for Shop Owner

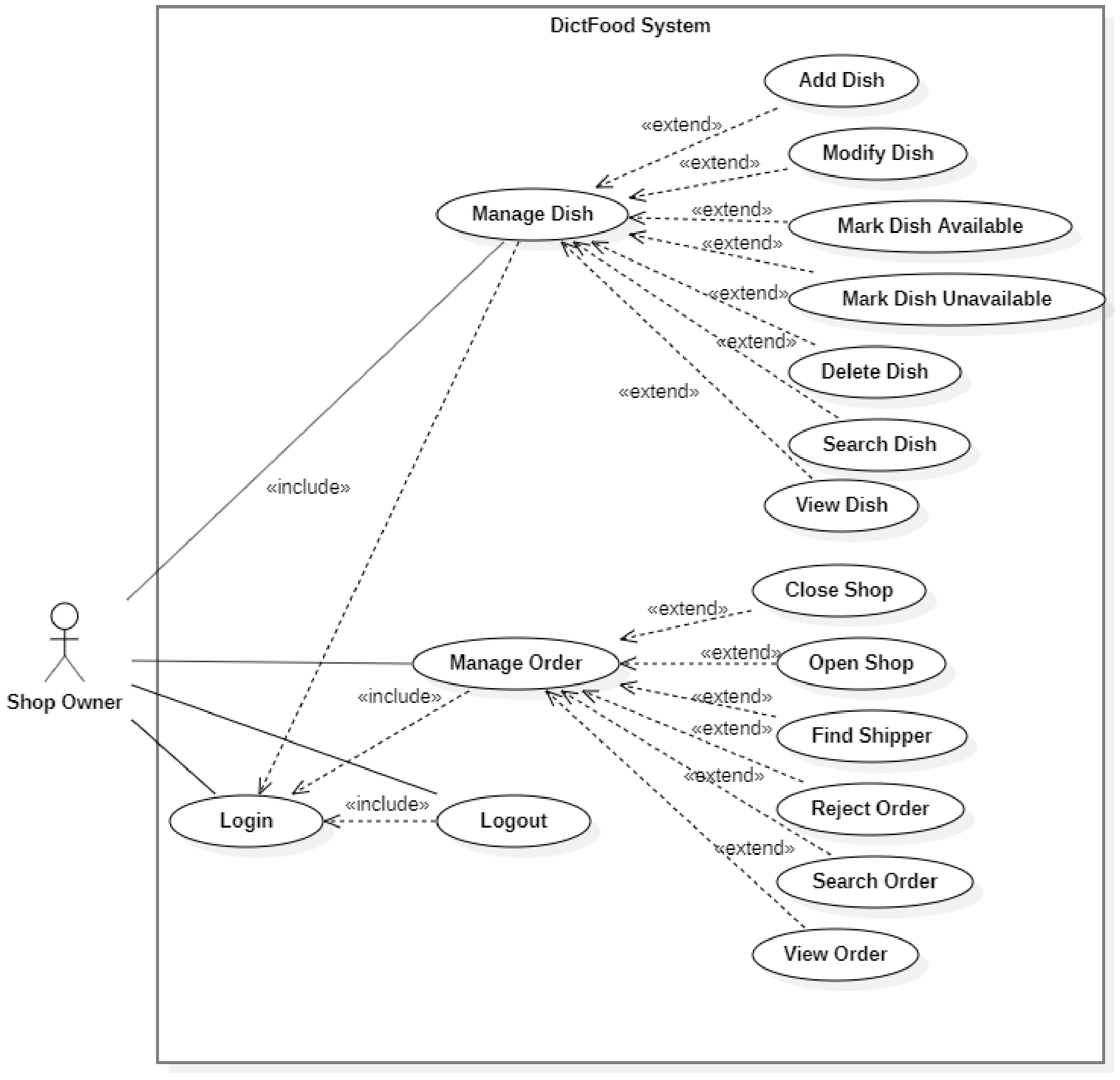


Figure 2.3: Use case diagram – Shop Owner

3) Use case diagram for Customer

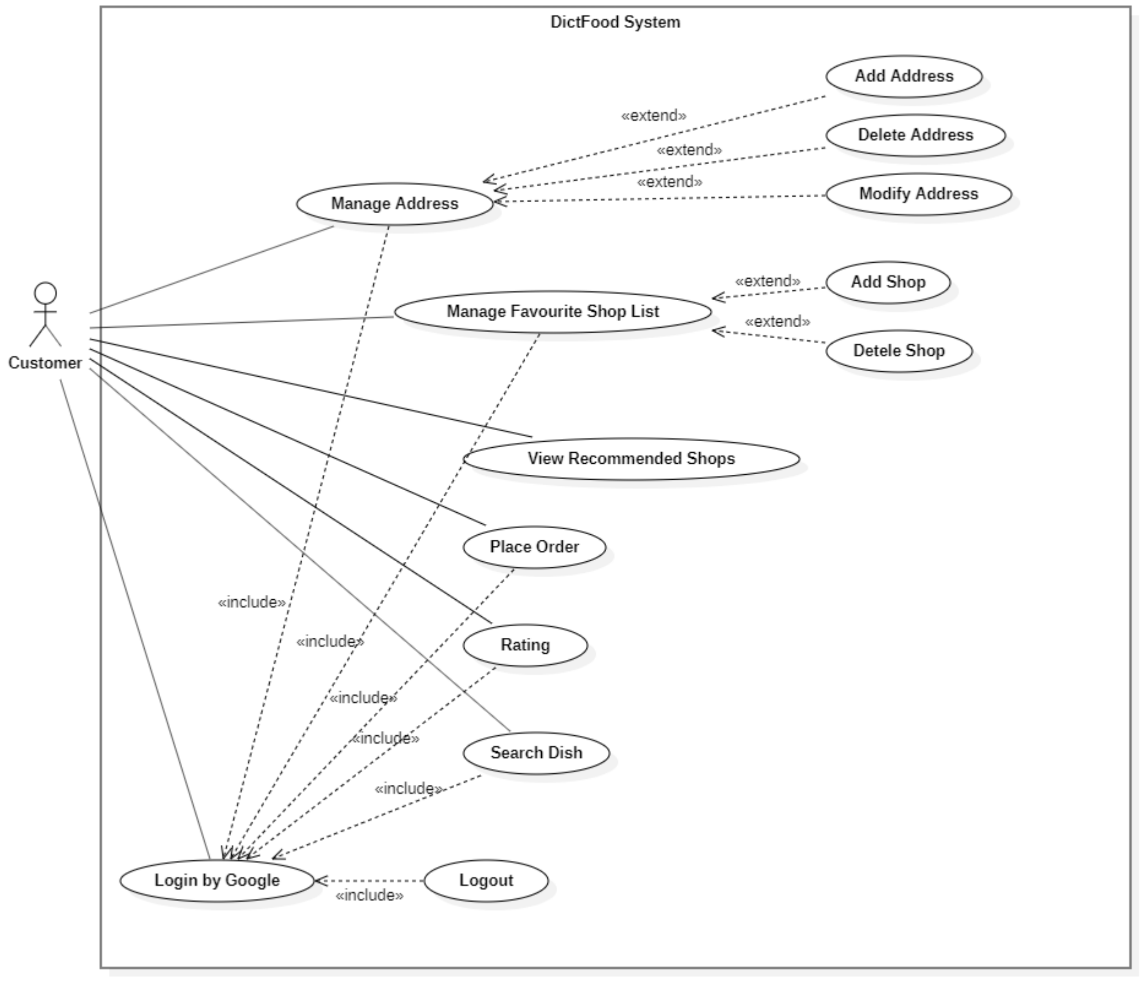


Figure 2.4: Use case diagram – Customer

4) Use case diagram for Shipper

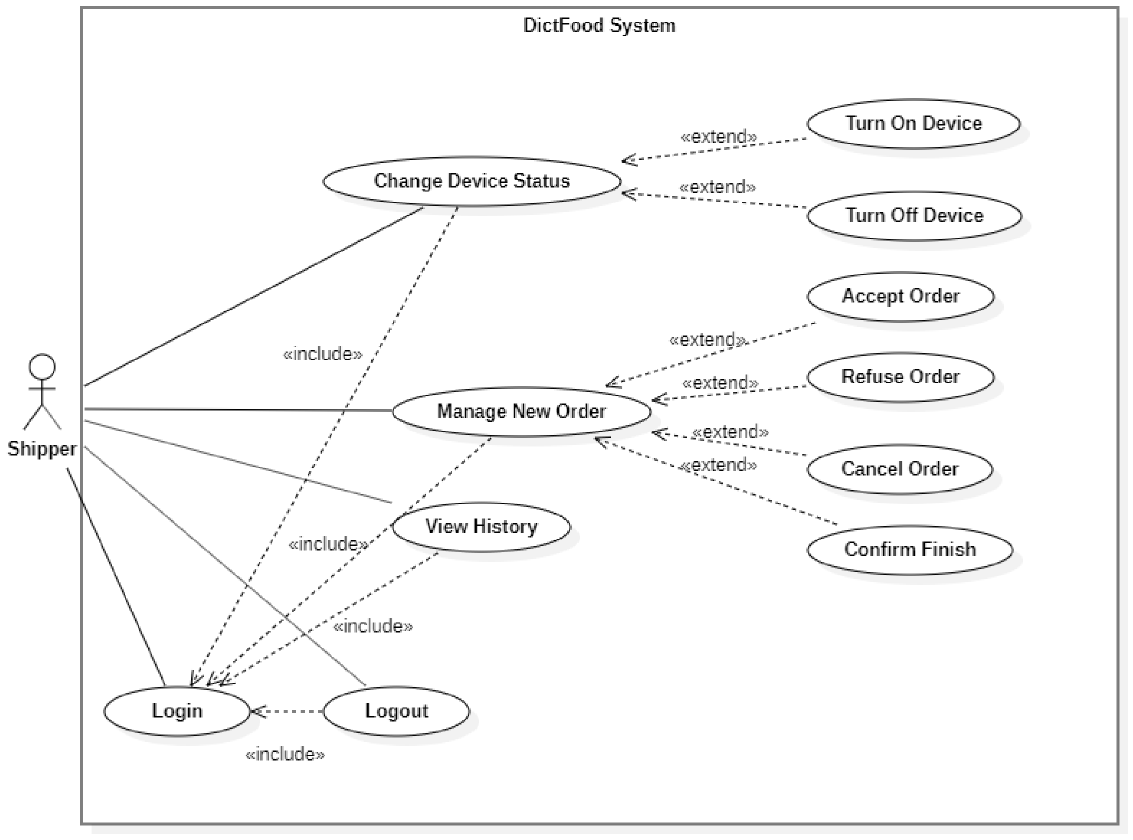


Figure 2.5: Use case diagram – Shipper

2.2.3. Description of the main use case

Table 2.1: Use case description – Login

Use case ID	UC–01		
Actor	Administrator, Shop Owner		
Brief description	This use case for logging in		
Pre–conditions	User who has account to access to system		
Post–conditions	User logged in successfully		
Flow of events		Actor Input	System Response
	1	Open the system	
	2		Show login form
	3	Fill the email and password. Click “Login” button	
	4		System validates account that user filled and submit. Show home screen if user’s account is correct.

Table 2.2: Use case description – Logout

Use case ID	UC–02		
Actor	Administrator, Shop Owner		
Brief description	This use case for logging out		
Pre–conditions	User who has account to access to system and logged in		
Post–conditions	User logged in successfully		
Flow of events		Actor Input	System Response
	1	Open the system	

	2	Click button “v”, click button “Logout”	
	3		Show login page

Table 2.3: Use case description – Administrator – Create user account

Use case ID	UC–03		
Actor	Administrator		
Brief description	This use case for creating employee or administrator account		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Server received the request from Administrator then adds data into database		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show user list
	3	Click “Add” button	
	4		Show create user form
	5	Fill all information including username, password, email and so on. Click Create button	
	6		System validates the information that the administrator filled and submit. If the information hasn't been used by anyone before, it will show successful alert

Table 2.4: Use case description – Administrator – Modify user

Use case ID	UC–04		
Actor	Administrator		
Brief description	This use case for managing employee account		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Modify personal information		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show user list
	3	Click “Edit” button on any employee account	
	4		Show user information
	5	Change any fields in there.	
	6	Press “OK” button	
	7		Show successful alert

Table 2.5: Use case description – Administrator – View user list

Use case ID	UC–05		
Actor	Administrator		
Brief description	This use case for viewing user list		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Show user list		
Flow of events		Actor Input	System Response

	1	Select "User Management"	
	2		Show all the user list

Table 2.6: Use case description – Administrator – Search user

Use case ID	UC–06		
Actor	Administrator		
Brief description	This use case for searching user		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Show list of users is being searched		
Flow of events		Actor Input	System Response
	1	Select "User Management"	
	2		Show all the user list
	3	Input anything in "Search" box	
	4		Show the user list matching "Search" box

Table 2.7: Use case description – Administrator – Suspend user

Use case ID	UC–07
Actor	Administrator
Brief description	This use case for deleting user
Pre–conditions	Administrator already logged into the system

Post–conditions	Block chosen user		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show all the user list
	3	Click “Status” button on any employee account	

Table 2.8: Use case description – Administrator – Active user

Use case ID	UC–08		
Actor	Administrator		
Brief description	This use case for deleting user		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Active chosen user		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show all the user list
	3	Click “Status” button on any employee account	

Table 2.9: Use case description – Administrator – Delete user

Use case ID	UC–09		
Actor	Administrator		
Brief description	This use case for deleting user		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Deleted chosen user		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show all the user list
	3	Click “Delete” button on any employee account	

Table 2.10: Use case description – Administrator – Delete user

Use case ID	UC–10		
Actor	Administrator		
Brief description	This use case for adding shop		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Added shop		
Flow of events		Actor Input	System Response
	1	Select “User Management”	
	2		Show user list
	3	Click “Add” button	

	4	Fill all information including username, password, email and so on. Click “OK” button	
	5		System validates the information that the administrator filled and submit. If the information hasn't been used by anyone before, it will show successful alert

Table 2.11: Use case description – Administrator – View shop list

Use case ID	UC–11		
Actor	Administrator		
Brief description	This use case for viewing shop list		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Show shop list		
Flow of events		Actor Input	System Response
	1	Select “Shop Management”	
	2		Show all the shop list

Table 2.12: Use case description – Administrator – Modify shop

Use case ID	UC–12
Actor	Administrator
Brief description	This use case for modify shop infomation
Pre–conditions	Administrator already logged into the system
Post–conditions	Modify shop information

Flow of events		Actor Input	System Response
	1	Select "Shop Management"	
	2		Show shop list
	3	Click on any shop name	
	4		Show shop information
	5	Change any fields in there.	
	6	Press "OK" button	
	7		Show successful alert

Table 2.13: Use case description – Administrator – Search shop

Use case ID	UC–13		
Actor	Administrator		
Brief description	This use case for searching user		
Pre–conditions	Administrator already logged into the system		
Post–conditions	Show list of users is being searched		
Flow of events		Actor Input	System Response
	1	Select "Shop Management"	
	2		Show all the shop list
	3	Input anything in "Search" box	
	4		Show the shop list matching "Search" box

Table 2.14: Use case description – Shop Owner – Add user

Use case ID	UC–14		
Actor	Shop Owner		
Brief description	This use case for adding dish		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Added dish		
Flow of events		Actor Input	System Response
	1	Select "Dish Management"	
	2		Show dish list
	3	Click "Add" button	
	4	Fill all information including name, price, picture and so on. Click "OK" button	
	5		System validates that information that and submit. It will show successful alert

Table 2.15: Use case description – Shop Owner – View dish list

Use case ID	UC–15		
Actor	Shop Owner		
Brief description	This use case for viewing dish list		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Show dish list		
Flow of events		Actor Input	System Response

	1	Select "Dish Management"	
	2		Show all the shop list

Table 2.16: Use case description – Shop Owner – Modify dish

Use case ID	UC–16		
Actor	Shop Owner		
Brief description	This use case for modifying dish		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Modified dish information		
Flow of events		Actor Input	System Response
	1	Select "Dish Management"	
	2		Show dish list
	3	Click "Edit" button on any dish	
	4		Show dish information
	5	Change any fields in there.	
	6	Press "OK" button	
	7		Show successful alert

Table 2.17: Use case description – Shop Owner – Delete dish

Use case ID	UC–17		
Actor	Shop Owner		
Brief description	This use case for deleting dish		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Deleted chosen dish		
Flow of events		Actor Input	System Response
	1	Select "Dish Management"	
	2		Show all the dish list
	3	Click "x" button on any dish	

Table 2.18: Use case description – Shop Owner – Search dish

Use case ID	UC–18		
Actor	Shop Owner		
Brief description	This use case for searching dish		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Show list of dishes is being searched		
Flow of events		Actor Input	System Response
	1	Select "Dish Management"	
	2		Show all the dish list
	3	Input in "Search" box	
	4		Show the user list matching "Search" box

Table 2.19: Use case description – Shop Owner – Change dish status

Use case ID	UC–19		
Actor	Shop Owner		
Brief description	This use case for changing dish status		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Changed shop status		
Flow of events		Actor Input	System Response
	1	Select "Order Management"	
	2		Show order list
	3	Click "Close Shop" button to close shop or click "Open Shop" button to open shop	
	4		Open or close shop depend on button

Table 2.20: Use case description – Shop Owner – Change dish status

Use case ID	UC–20		
Actor	Shop Owner		
Brief description	This use case for starting to search shipper for an order		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Changed shop status		
Flow of events		Actor Input	System Response
	1	Select "Order Management"	
	2		Show order list

	3	Click “>” button on any order which has “Waiting” status	
	4		Searching for that order and show notification

Table 2.21: Use case description – Shop Owner – Reject order

Use case ID	UC–21		
Actor	Shop Owner		
Brief description	This use case for rejecting an order		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Changed shop status		
Flow of events		Actor Input	System Response
	1	Select “Order Management”	
	2		Show order list
	3	Click “x” button on any order which has “Waiting” status	
	4		Show confirm dialog
	5	Click “OK” button	
	6		That item status change into “REJECTED”

Table 2.22: Use case description – Shop Owner – Search order

Use case ID	UC–22
Actor	Shop Owner
Brief description	This use case for searching order

Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Show list of users is being searched		
Flow of events		Actor Input	System Response
	1	Select “Order Management”	
	2		Show all the order list
	3	Input id in “Search” box	
	4		Show the shop list with id in “Search” box

Table 2.23: Use case description – Shop Owner – View order list

Use case ID	UC–23		
Actor	Shop Owner		
Brief description	This use case for viewing order list		
Pre–conditions	Shop Owner already logged into the system		
Post–conditions	Show order list		
Flow of events		Actor Input	System Response
	1	Select “Order Management”	
	2		Show all the order list

Table 2.24: Use case description – Customer – Login

Use case ID	UC–24		
Actor	Customer		
Brief description	This use case for logging in		
Pre–conditions	User who has Google account to access to system		
Post–conditions	User logged in successfully		
Flow of events		Actor Input	System Response
	1	Open the system	
	2		Show login form
	3	Click “Login with Google” button	
	4		If email was exist on the system, show home screen. If email wasn’t exist on system, System will create automatically. After that, show home screen.

Table 2.25: Use case description – Customer – Logout

Use case ID	UC–25		
Actor	Customer		
Brief description	This use case for logging out		
Pre–conditions	User who has Google account to access to system		
Post–conditions	User logged out successfully		
Flow of events		Actor Input	System Response
	1	Open home screen	
	2		Show home screen

	3	Click “User” tab	
	4		Show user tab
	5	Click “Logout” button	
	6		Logout, move to login screen

Table 2.26: Use case description – Customer – View recommended shop list

Use case ID	UC–26		
Actor	Customer		
Brief description	This use case for viewing recommended shop list		
Pre–conditions	Customer already logged into the system		
Post–conditions	Show recommended shop list		
Flow of events		Actor Input	System Response
	1	Open home screen	
	2		Show all recommended shop list

Table 2.27: Use case description – Customer – Search dish

Use case ID	UC–27		
Actor	Customer		
Brief description	This use case for searching order		
Pre–conditions	Customer already logged into the system		
Post–conditions	Show list of users is being searched		
Flow of events		Actor Input	System Response
	1	Open home screen	
	2	Select search icon	

	3		Show search screen
	4	Input dish name in “Search” box, and click search	
	5		Show all shops found which have that dish

Table 2.28: Use case description – Customer – Add address

Use case ID	UC–28		
Actor	Customer		
Brief description	This use case for adding address		
Pre–conditions	Customer already logged into the system		
Post–conditions	Added address		
Flow of events		Actor Input	System Response
	1	Open main screen	
		Click “User” tab	
			Show user’s functions
		Click “Address” button	
	2		Show address list
	3	Click “Add” button	
	4	Fill all information including name, phone and so on, pick up your location. Click “OK” button	
	5		System validates that information that and submit. It will show successful alert

Table 2.29: Use case description – Customer – Delete address

Use case ID	UC–29		
Actor	Customer		
Brief description	This use case for deleting address		
Pre–conditions	Customer already logged into the system		
Post–conditions	Deleted address		
Flow of events		Actor Input	System Response
	1	Open main screen	
		Click “User” tab	
			Show user functions
		Click “Address” button	
	2		Show address list
	3	Swipe to left or right	
	4		That address disappeared

Table 2.30: Use case description – Customer – Modify address

Use case ID	UC–30		
Actor	Customer		
Brief description	This use case for modify address		
Pre–conditions	Customer already logged into the system		
Post–conditions	Modified address		
Flow of events		Actor Input	System Response
	1	Open main screen	

	2	Click “User” tab	
	3		Show user’s functions
	4	Click “Address” button	
	5		Show address list
	6	Click on any address	
	7	Fill all information including name, phone and so on, pick up your location. Click “OK” button	
	8		System validates that information that and submits. It will show successful alert

Table 2.31: Use case description – Customer – Add favorite shop

Use case ID	UC–31		
Actor	Customer		
Brief description	This use case for modifing address		
Pre–conditions	Customer already logged into the system		
Post–conditions	Modified address		
Flow of events		Actor Input	System Response
	1	Click on any shop at Main screen or Search screen	
	3		Show shop’s information
	4	Click “Save” button	
	5		Button becomes pink. That shop is on favorite list

Table 2.32: Use case description – Customer – Delete favorite shop

Use case ID	UC–32		
Actor	Customer		
Brief description	This use case for deleting favorite shop		
Pre–conditions	Customer already logged into the system		
Post–conditions	Deleted favorite shop		
Flow of events		Actor Input	System Response
	1	Open main screen. Click "Saved" tab	
	2		Show favorite list
	3	Swipe to left or right	
	4		That shop disappeared

Table 2.33: Use case description – Customer – Rate for shop

Use case ID	UC–33		
Actor	Customer		
Brief description	This use case for rating shop		
Pre–conditions	Customer already logged into the system and an order finished		
Post–conditions	Deleted favorite shop		
Flow of events		Actor Input	System Response
	1	Open main screen	
	2	Click "User" tab	
	3		Show User's functions
	4	Click "History"	
	5		Show history order

	6	Click on any order that is finished	
	7		Show order information
	8	Click on rating bar	
	9		Show rating on rating bar

Table 2.34: Use case description – Customer – Place order

Use case ID	UC–34		
Actor	Customer		
Brief description	This use case for rating shop		
Pre–conditions	Customer already logged into the system and an order finished		
Post–conditions	Deleted favorite shop		
Flow of events		Actor Input	System Response
	1	Open main screen or favorite tab	
	2		Show shops
	3	Click on any shop	
	4		Show shop’s information
	5	Click on any dish that you want	
	6		Show quantity form
	7	Click “Add” button	
	8		Show order detail on bottom sheet
	9	Click “Next” button	
	10		Show checkout information
	11	Click on address to change and input note. Click place order	
	12		Show successful notification

Table 2.35: Use case description – Shipper – Login

Use case ID	UC–35		
Actor	Shipper		
Brief description	This use case for logging in		
Pre–conditions	User who has an account to access to the system		
Post–conditions	User logged in successfully		
Flow of events		Actor Input	System Response
	1	Open the system	
	2		Show login form
	3	Fill in email and password. Click button login	
	4		Show Working Screen if login successfully

Table 2.36: Use case description – Shipper – Logout

Use case ID	UC–36		
Actor	Customer		
Brief description	This use case for logging out		
Pre–conditions	User who has Google account to access to the system		
Post–conditions	User logged out successfully		
Flow of events		Actor Input	System Response
	1	Open Working screen	
	2		Show working screen
	3	Click on “ ” icon	
	4		Logout, move to login screen

Table 2.37: Use case description – Shipper – View today history

Use case ID	UC–37		
Actor	Shipper		
Brief description	This use case for viewing history		
Pre–conditions	Shipper already logged into the system		
Post–conditions	User logged out successfully		
Flow of events		Actor Input	System Response
	1	Open Working screen	
	2		Show history today

Table 2.38: Use case description – Shipper – change device status

Use case ID	UC–38		
Actor	Shipper		
Brief description	This use case for changing device status		
Pre–conditions	Shipper already logged into the system		
Post–conditions	Changed device status		
Flow of events		Actor Input	System Response
	1	Open main screen	
	2		Show main screen
	3	Click “Power” button	
	4		Make device online/offline

Table 2.39: Use case description – Shipper – Accept order

Use case ID	UC–39		
Actor	Shipper		
Brief description	This use case for accepting order		
Pre–conditions	Shipper already logged into the system and online		
Post–conditions	Accepted order		
Flow of events		Actor Input	System Response
	1	Open Working screen, click “Power” button to make device online, wait for system send order to device	
	2		Send order information to the device. Show information of that order on a dialog
	3	Click “Accept” button	
	4		Accept and move to map screen, close dialog

Table 2.40: Use case description – Shipper – Refuse order

Use case ID	UC–40		
Actor	Shipper		
Brief description	This use case for refusing order		
Pre–conditions	Shipper already logged into the system and online		
Post–conditions	Accepted order		
Flow of events		Actor Input	System Response
	1	Click “Cancel” button	
	2		Close dialog

Table 2.41: Use case description – Shipper – Confirm finish

Use case ID	UC–41		
Actor	Shipper		
Brief description	This use case for confirming finish order		
Pre–conditions	Shipper already logged into the system and accept an order		
Post–conditions	Finished order		
Flow of events		Actor Input	System Response
	1	On bottom sheet, click “Finish” button	
	2		Show confirm dialog
	3	Click “OK” button	
	4		Move to waiting screen, close dialog

Table 2.42: Use case description – Shipper – Cancel order

Use case ID	UC–42		
Actor	Shipper		
Brief description	This use case for canceling order if any problem happen		
Pre–conditions	Shipper already logged into the system and accept an order		
Post–conditions	Canceled order		
Flow of events		Actor Input	System Response
	1	On the bottom sheet, click “Cancel” button	
	2		Show confirm dialog
	3	Click “OK” button	
	4		Move to waiting screen, close dialog

2.2.4. Activity diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. Activity diagram is used as a flowchart that consists of activities performed by the system. I must clearly understand about the elements used in activity diagram. The main of an activity diagram is the activity itself. After identifying the activities, I need to understand how they associated with constraints and conditions.

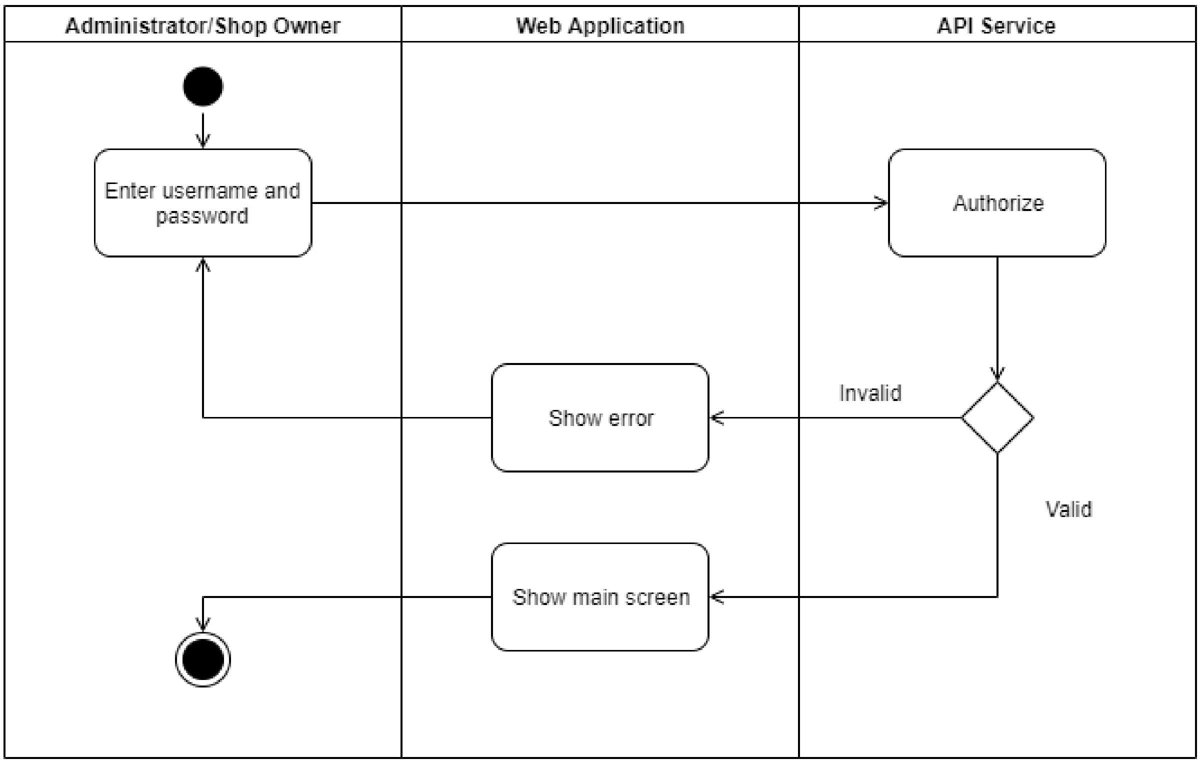


Figure 2.6: Activity diagram – Administrator/Shop Owner – Login

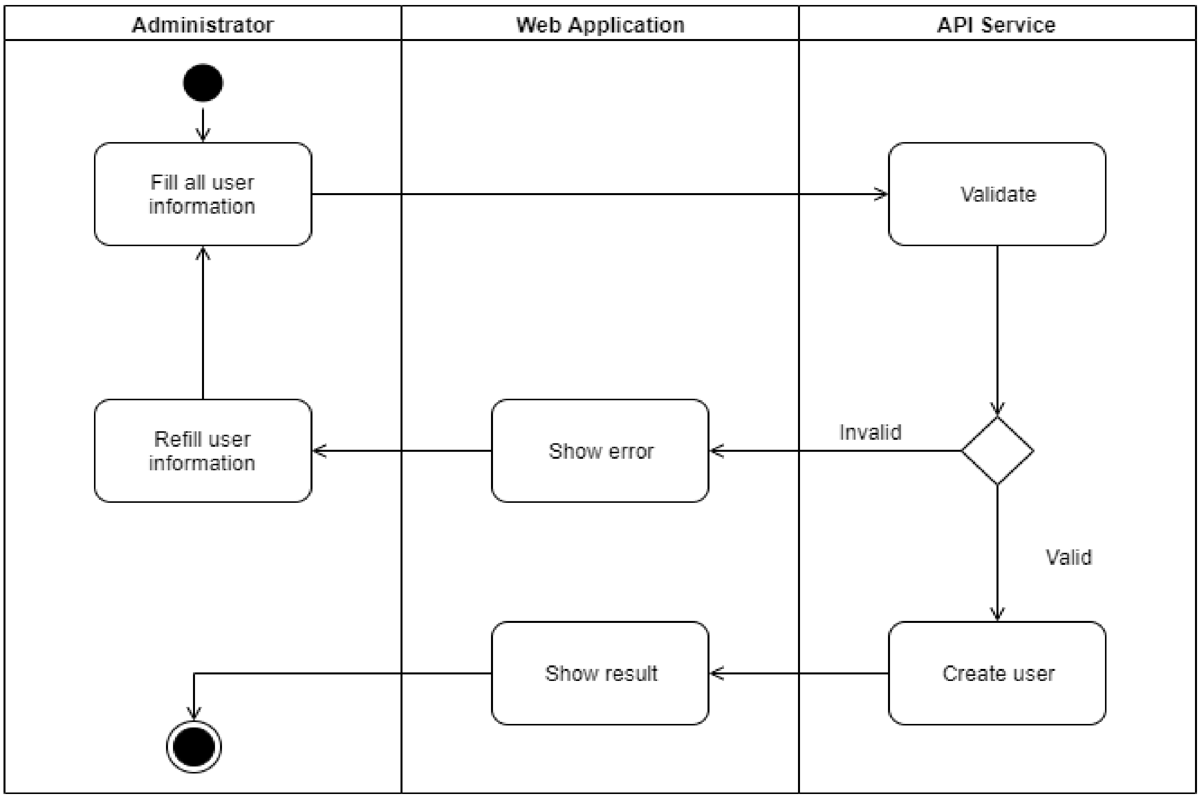


Figure 2.7: Activity diagram – Administrator – Add user

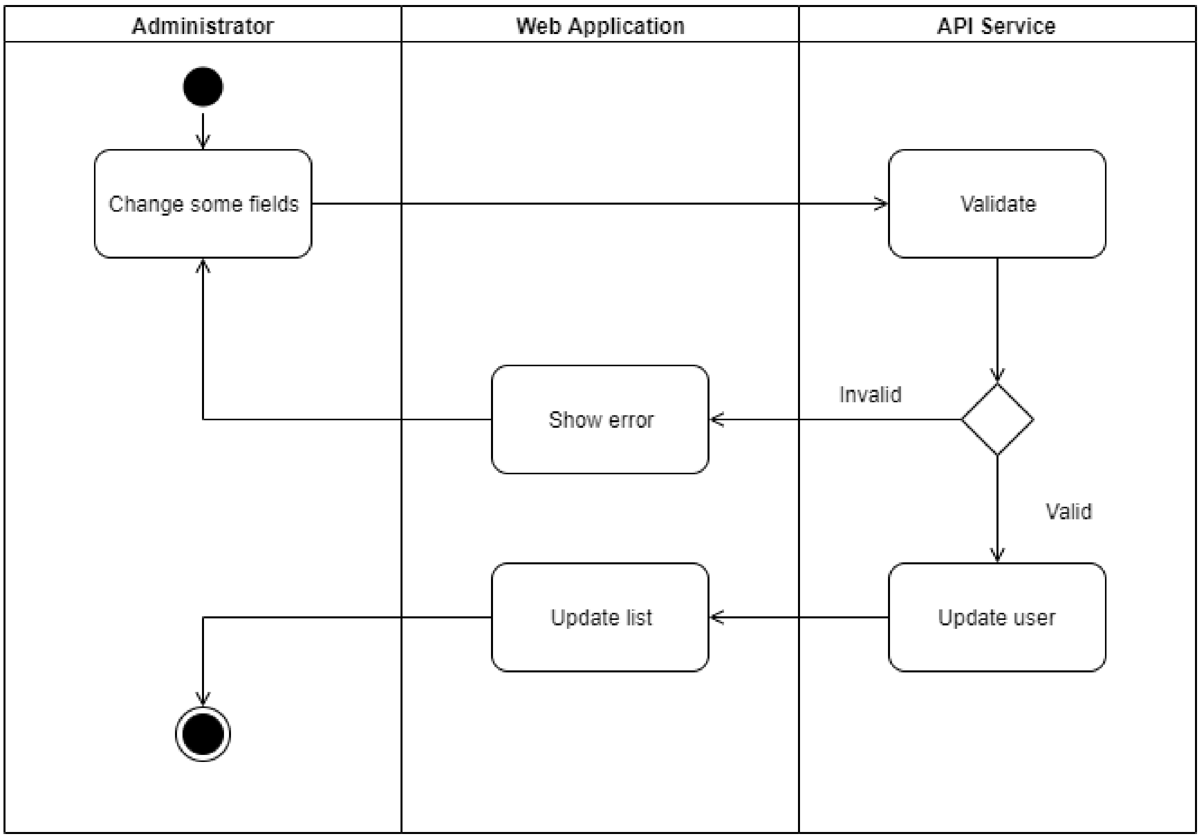


Figure 2.8: Activity diagram – Administrator – Modify user

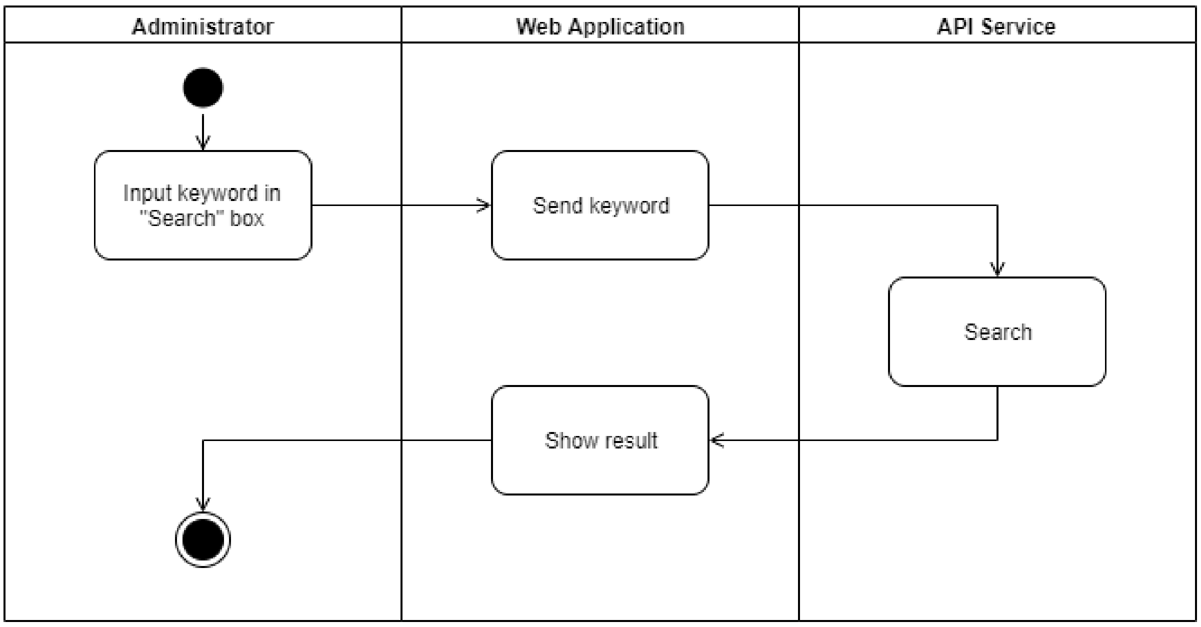


Figure 2.9: Activity diagram – Administrator – Search user

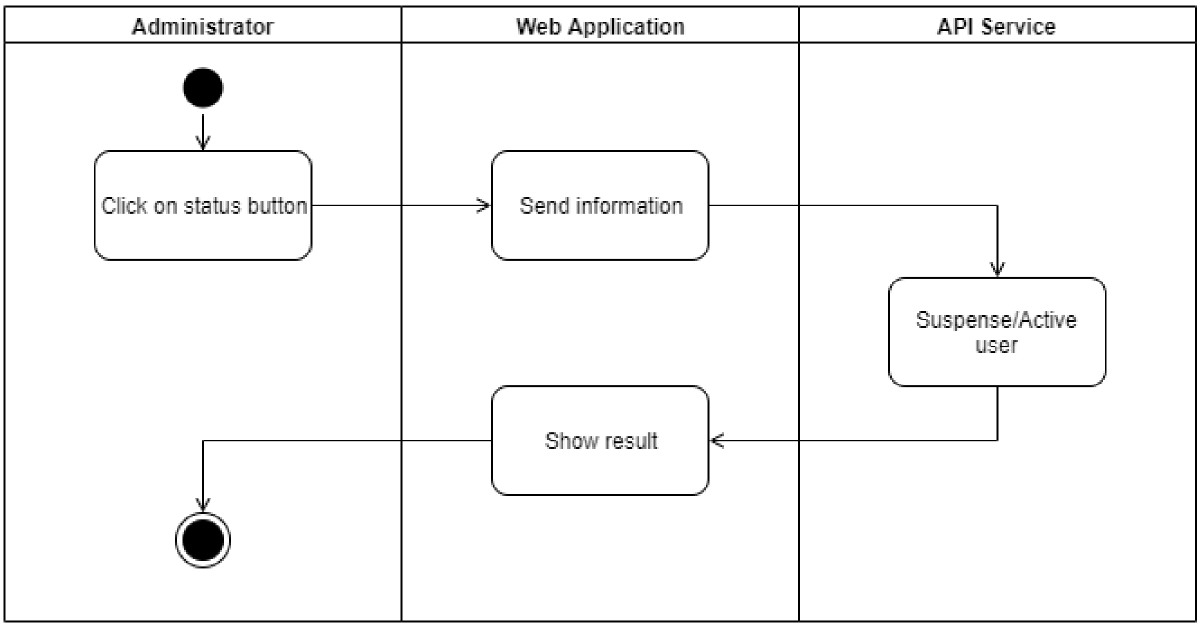


Figure 2.10: Activity diagram – Administrator – Modify user status

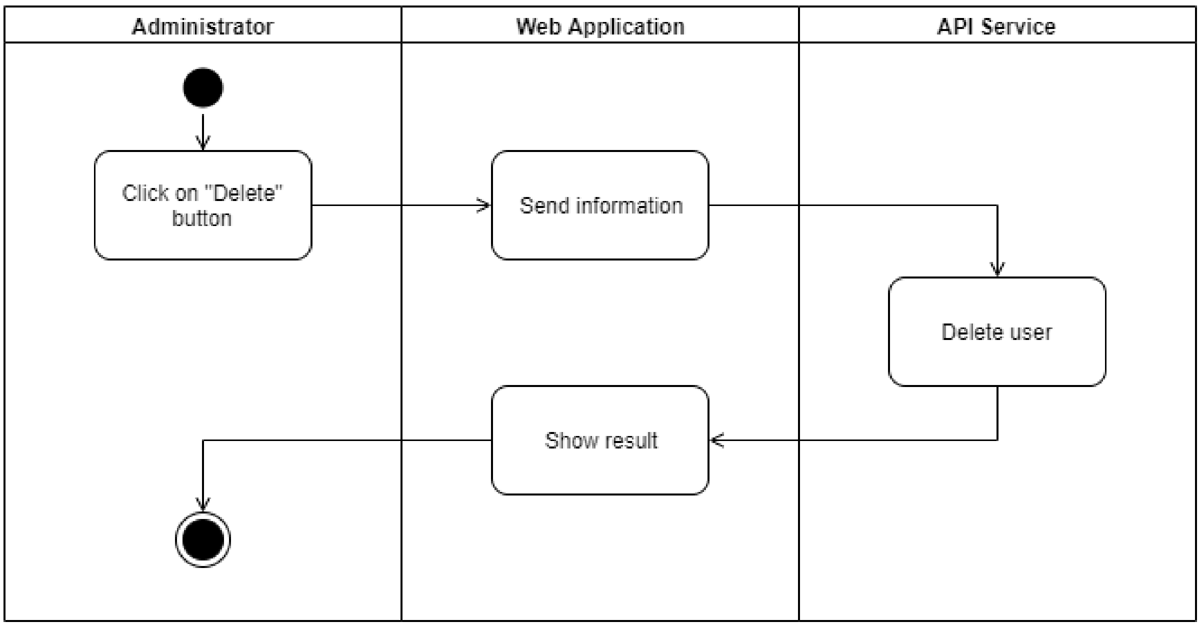


Figure 2.11: Activity diagram – Administrator – Modify user status

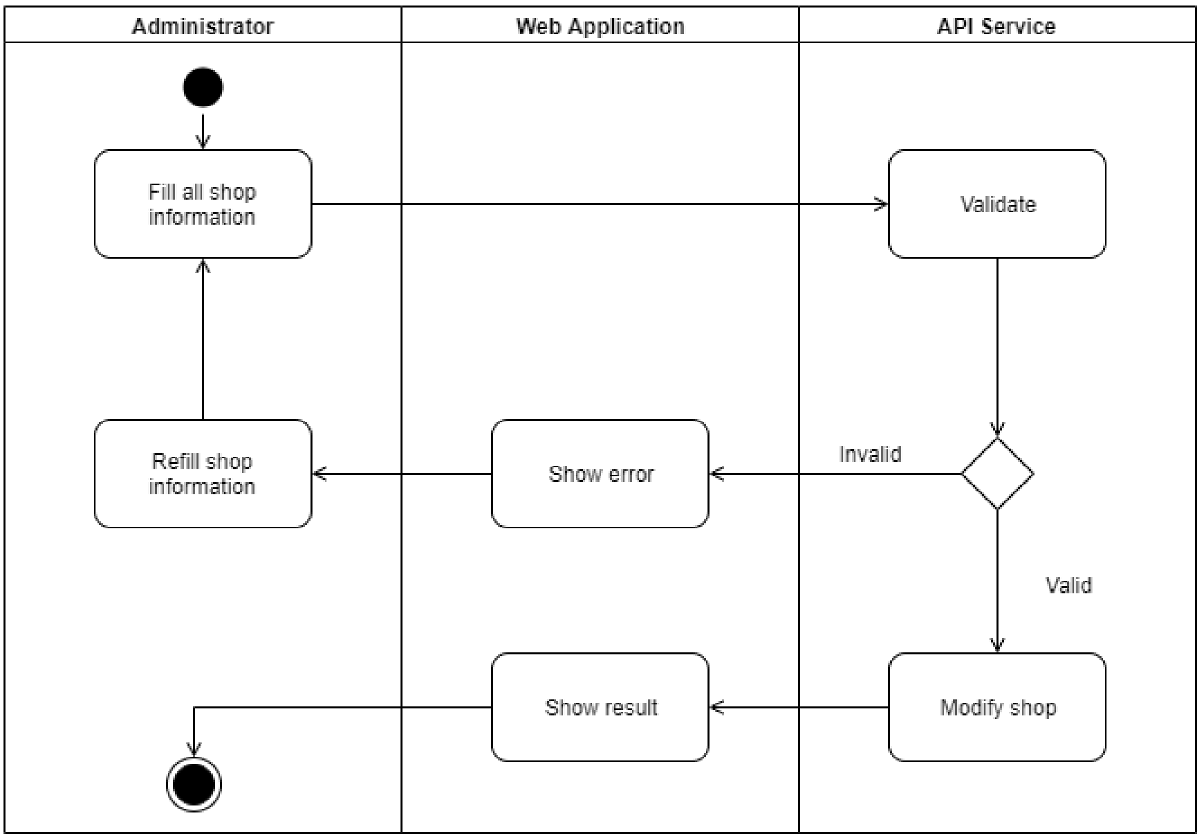


Figure 2.12: Activity diagram – Administrator – Add shop

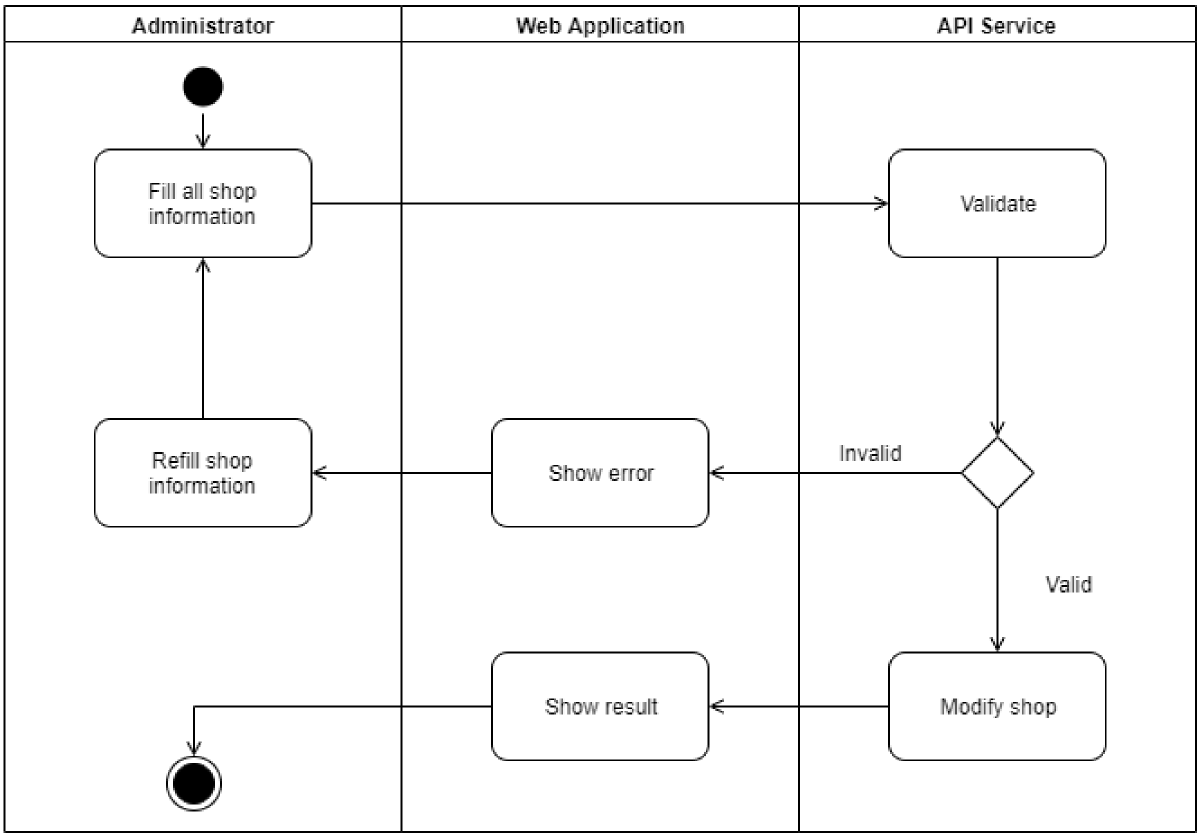


Figure 2.13: Activity diagram – Administrator – Modify shop

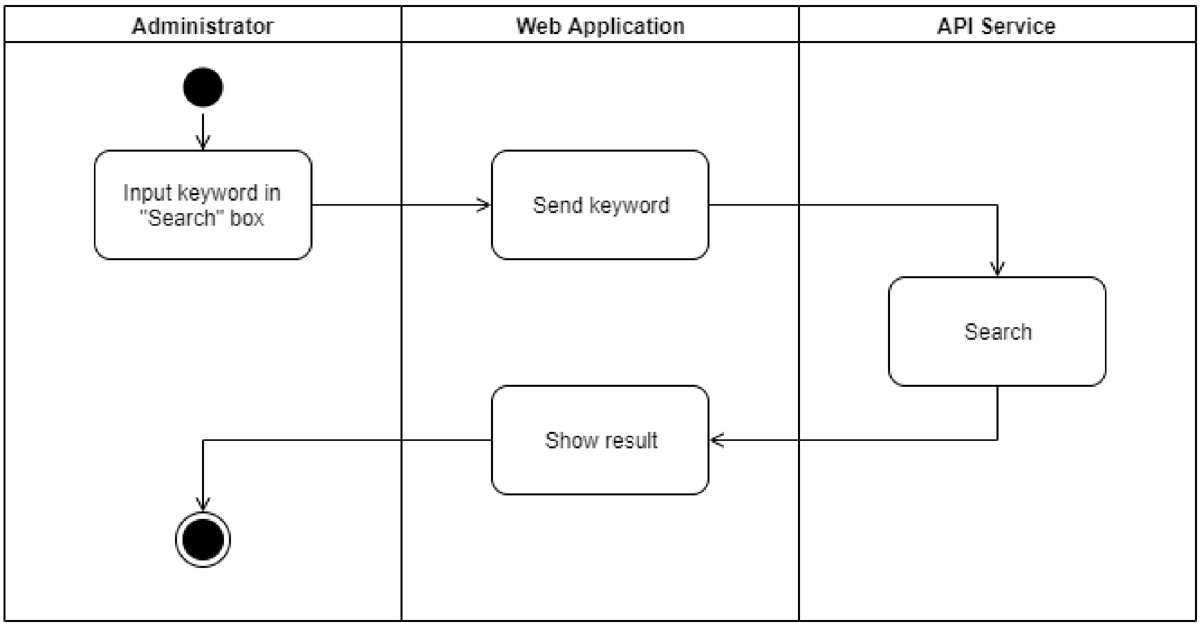


Figure 2.14: Activity diagram – Administrator – Search shop

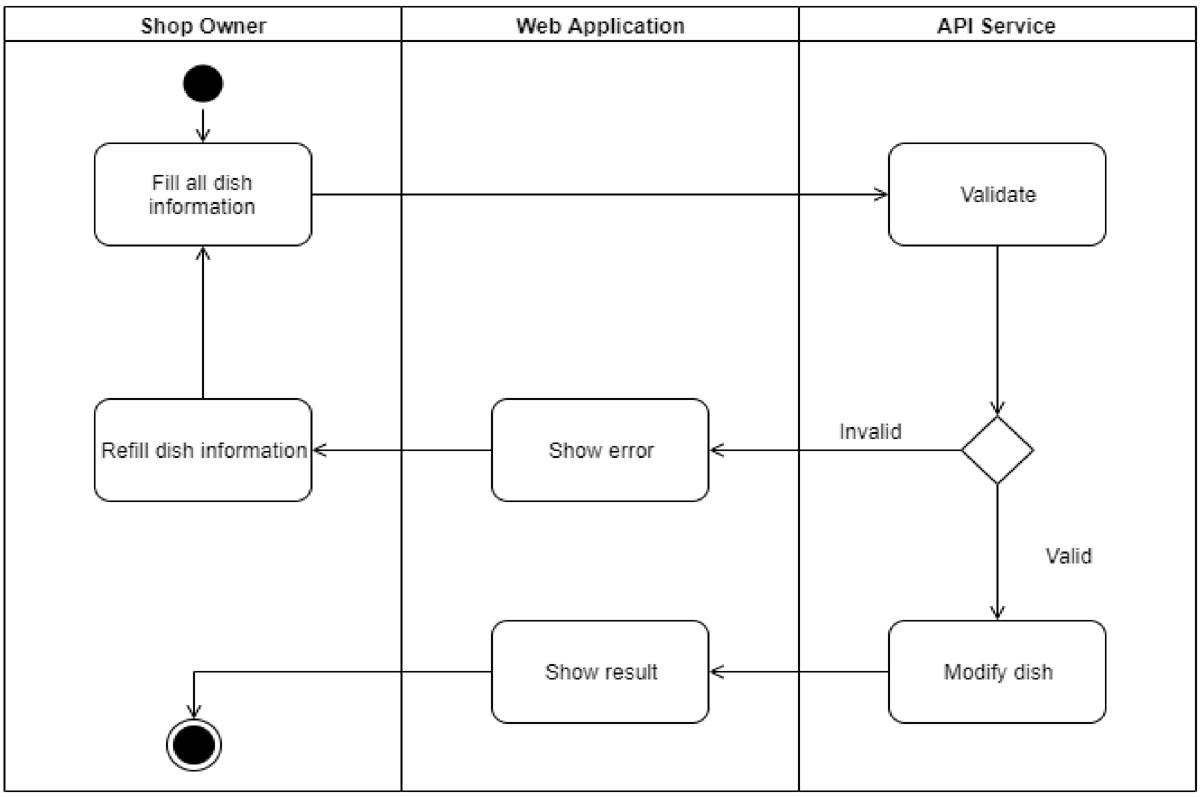


Figure 2.15: Activity diagram – Shop Owner – Add dish

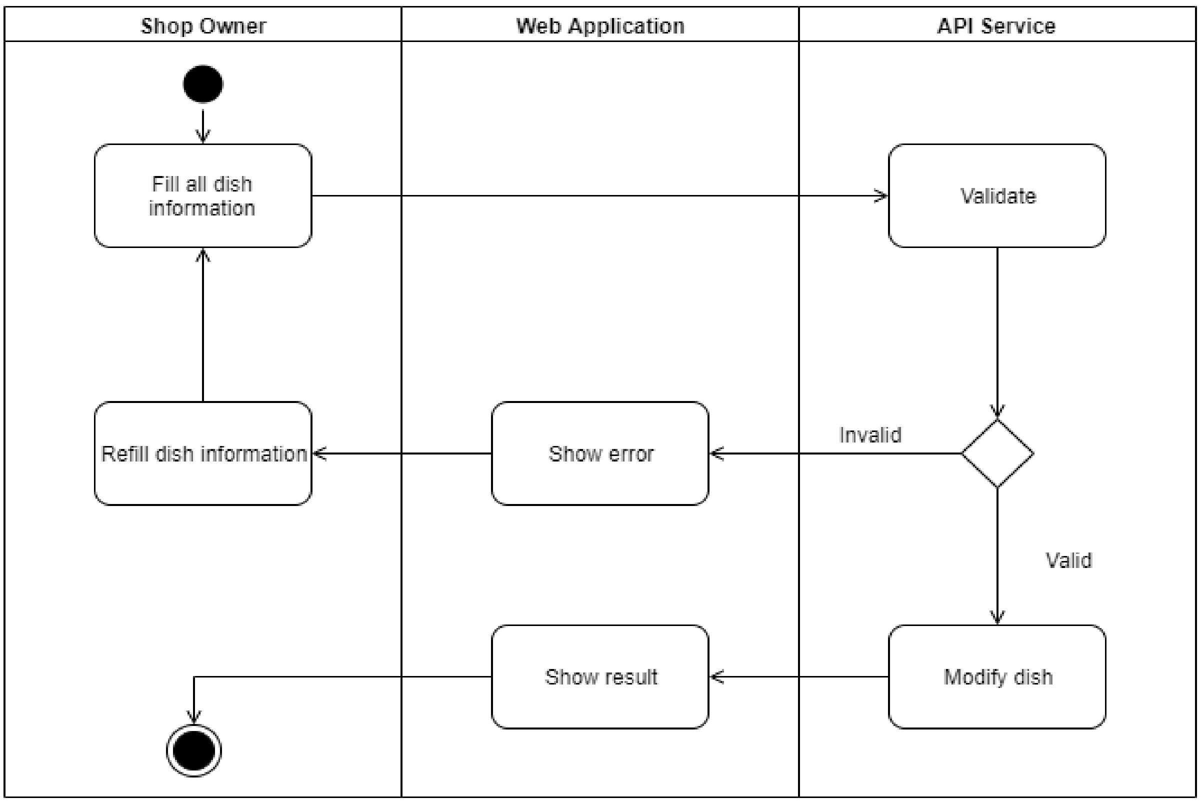


Figure 2.16: Activity diagram – Shop Owner – Modify dish

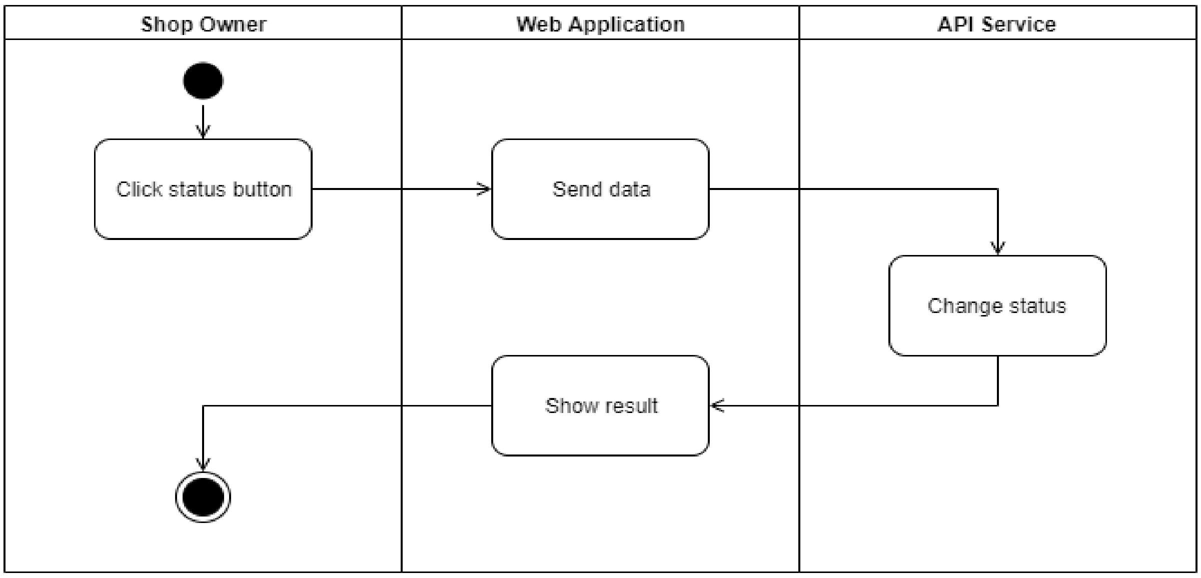


Figure 2.17: Activity diagram – Shop Owner – Change dish status

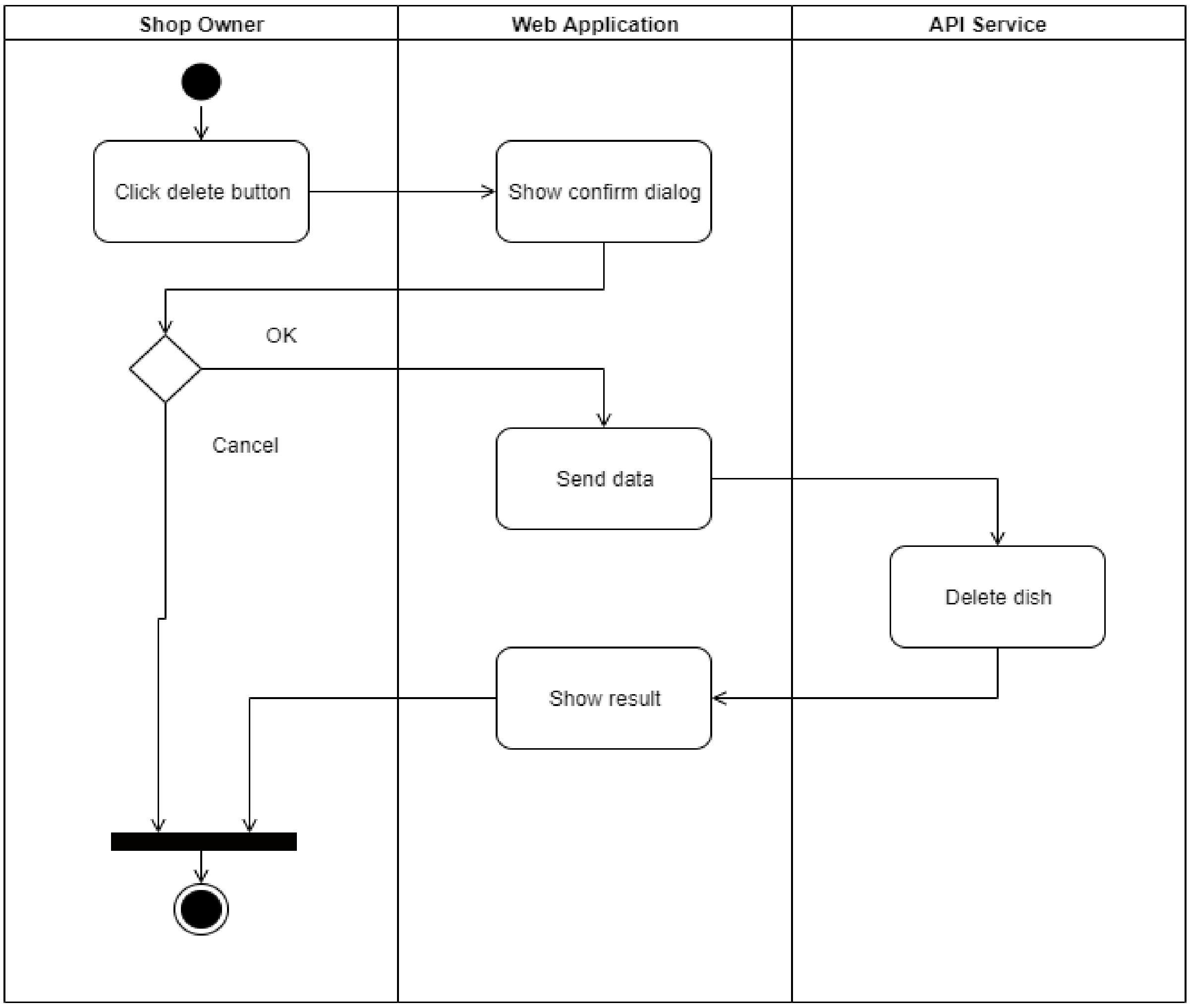


Figure 2.18: Activity diagram – Shop Owner – Delete dish

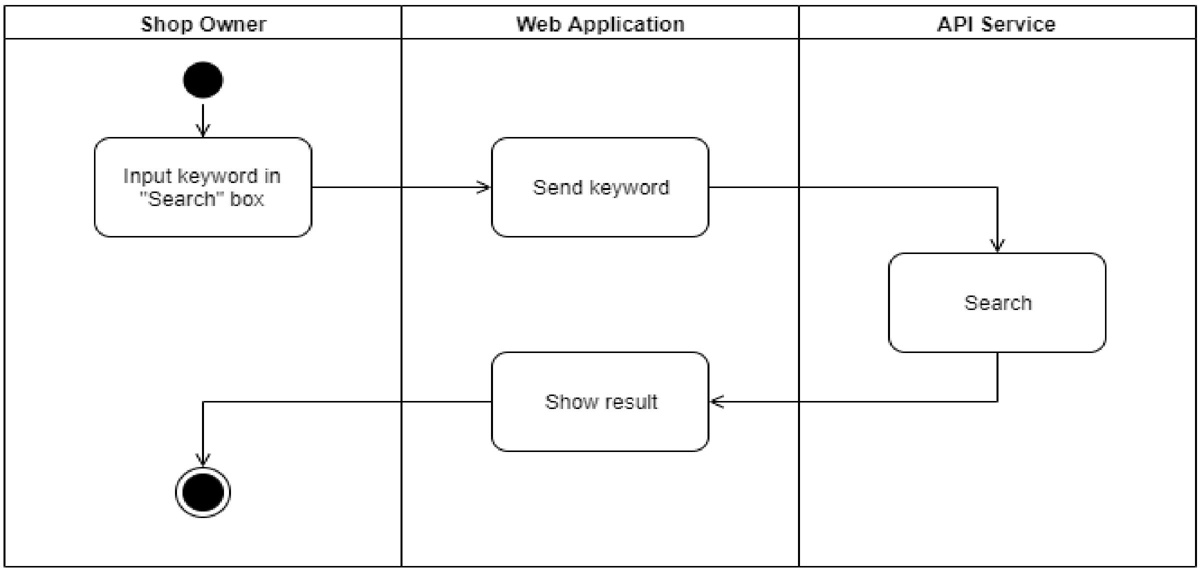


Figure 2.19: Activity diagram – Shop Owner – Search dish

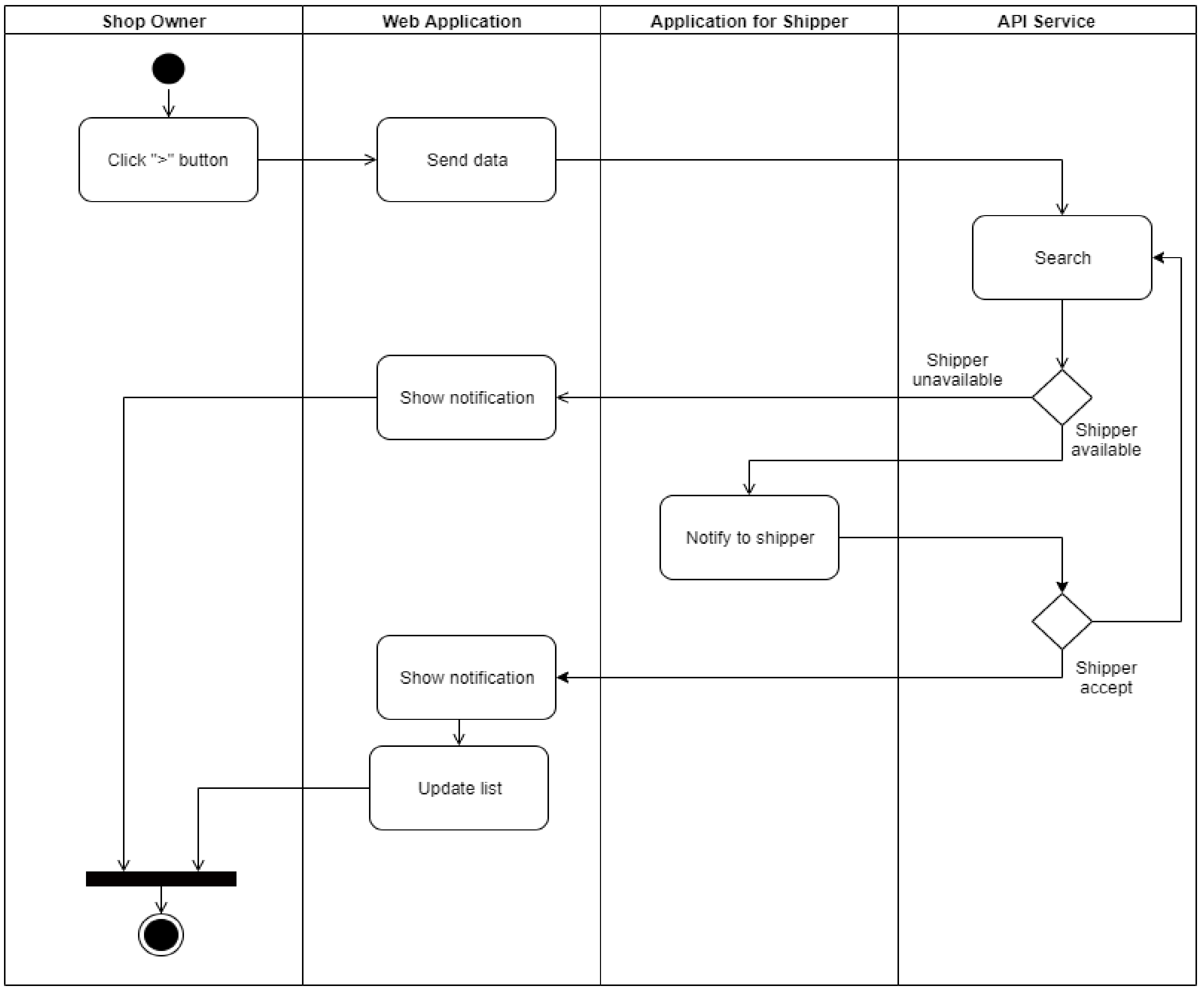


Figure 2.20: Activity diagram – Shop Owner – Search shipper

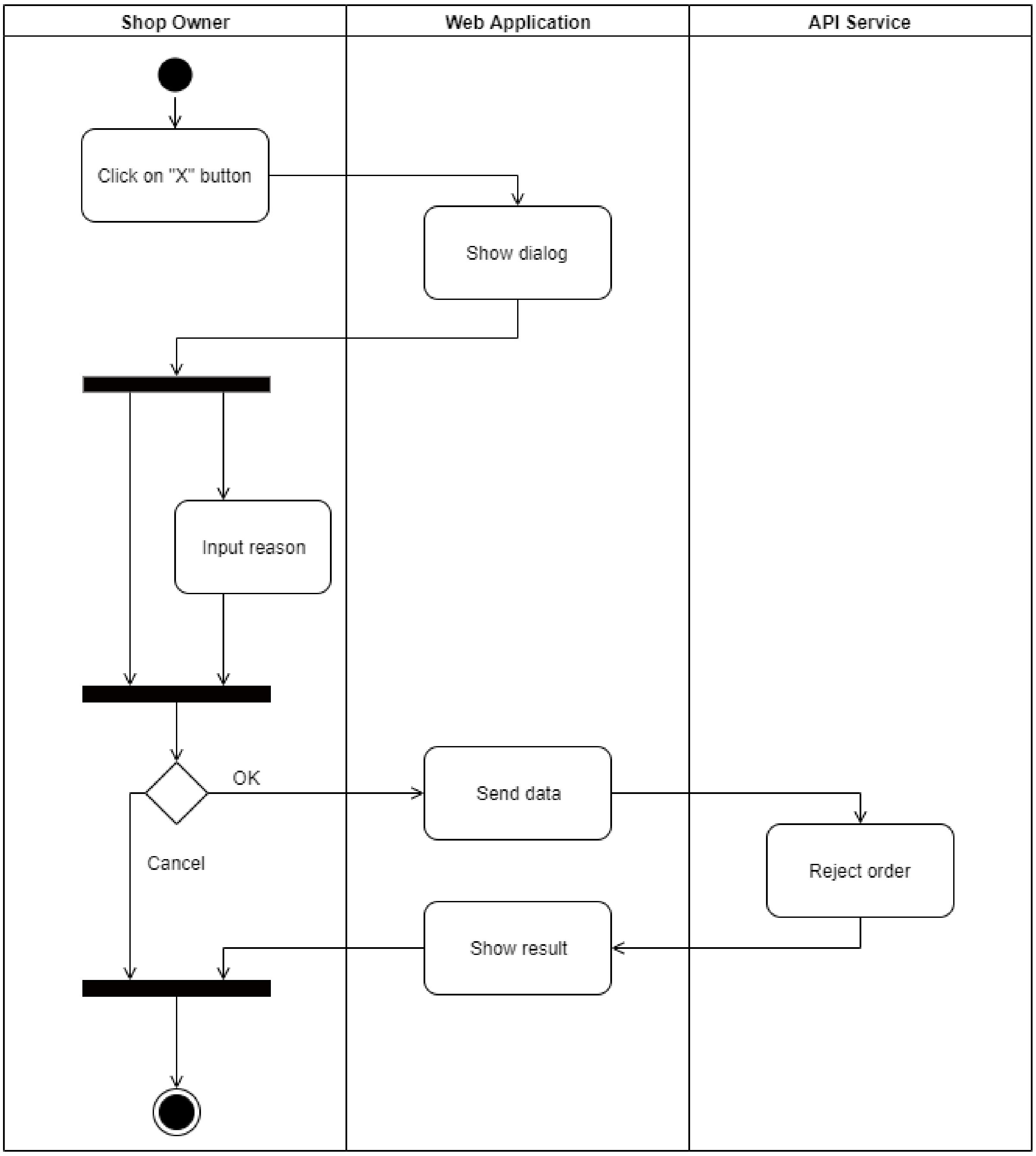


Figure 2.21: Activity diagram – Shop Owner – Reject order

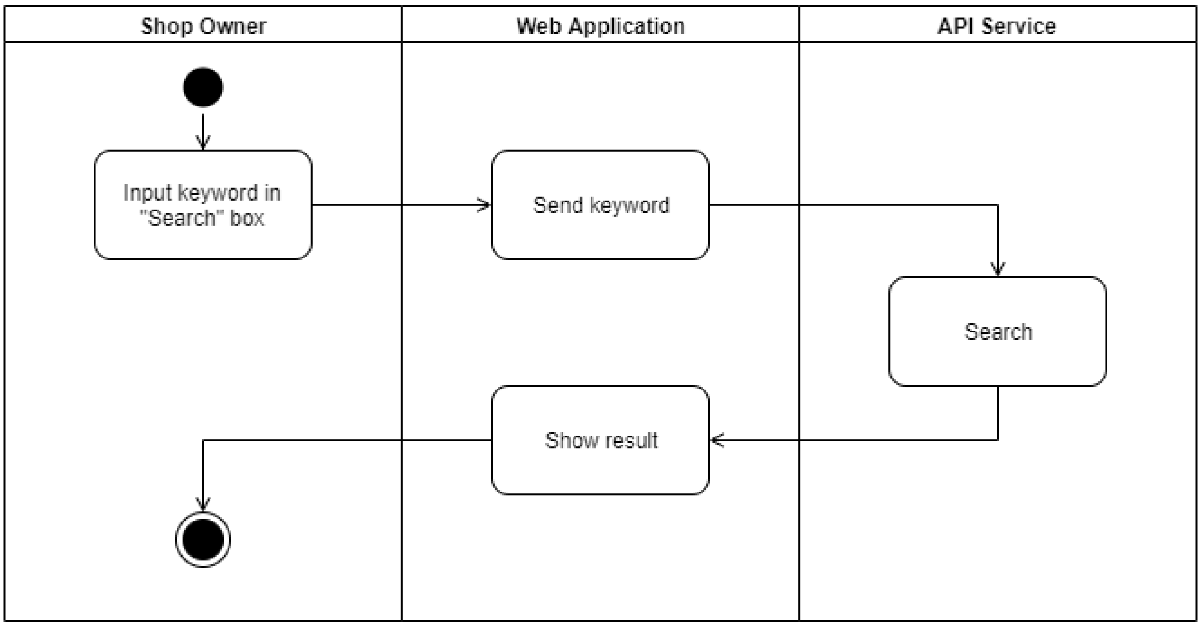


Figure 2.22: Activity diagram – Shop Owner – Search order

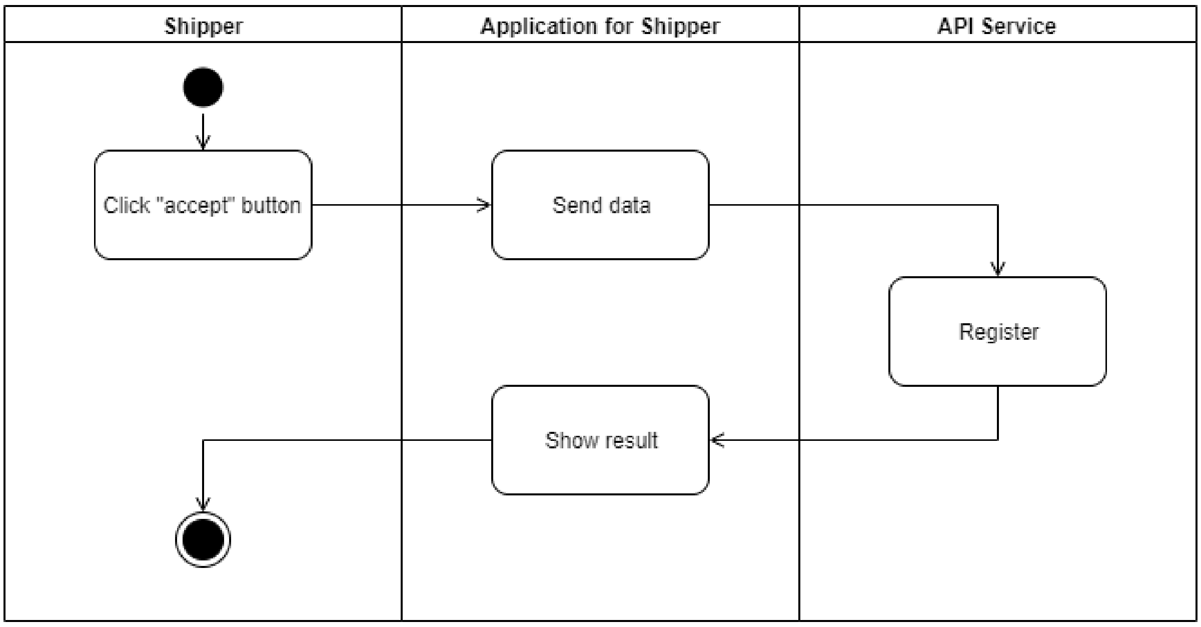


Figure 2.23: Activity diagram – Shipper – Accept order

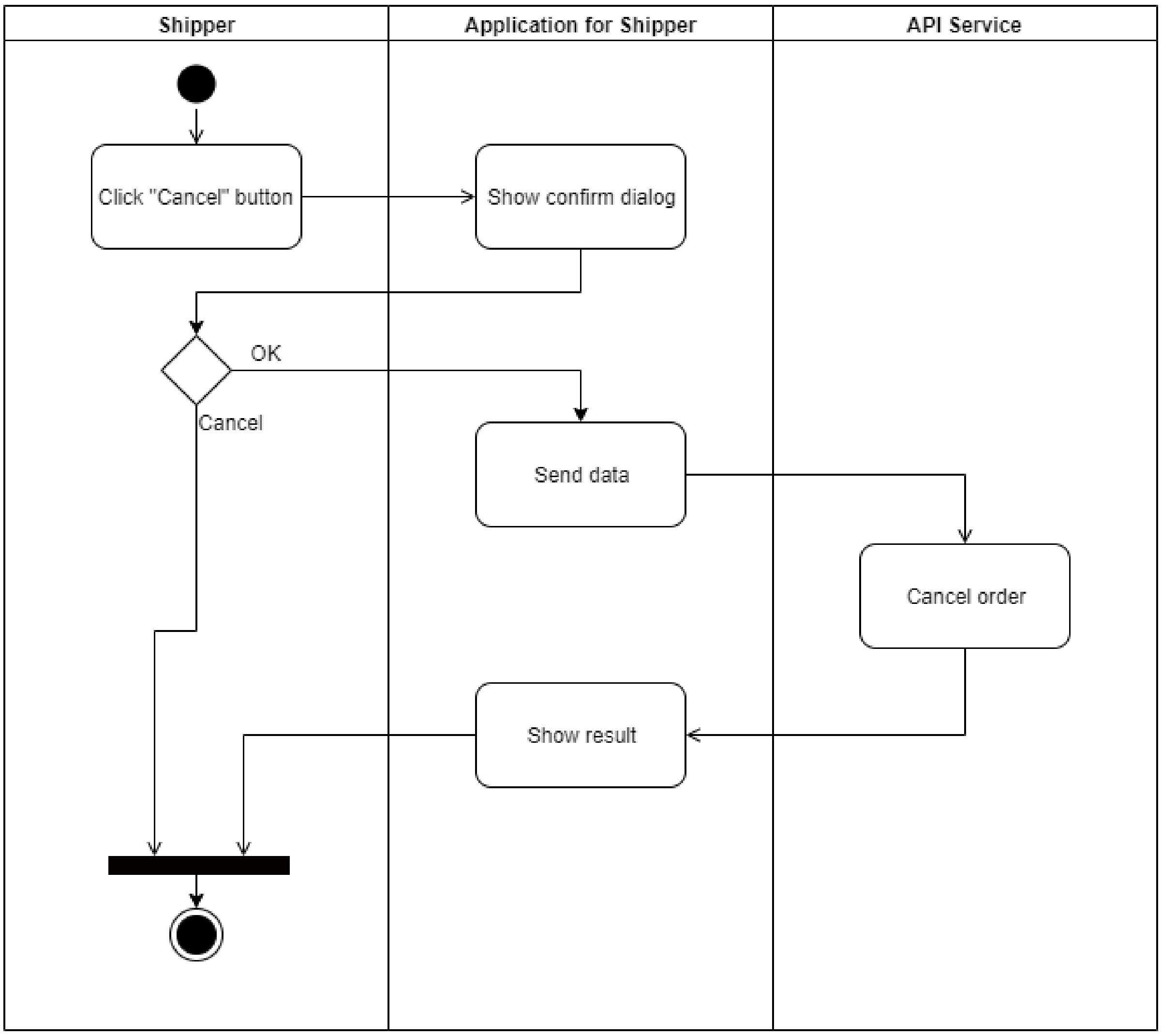


Figure 2.24: Activity diagram – Shipper – Cancel order

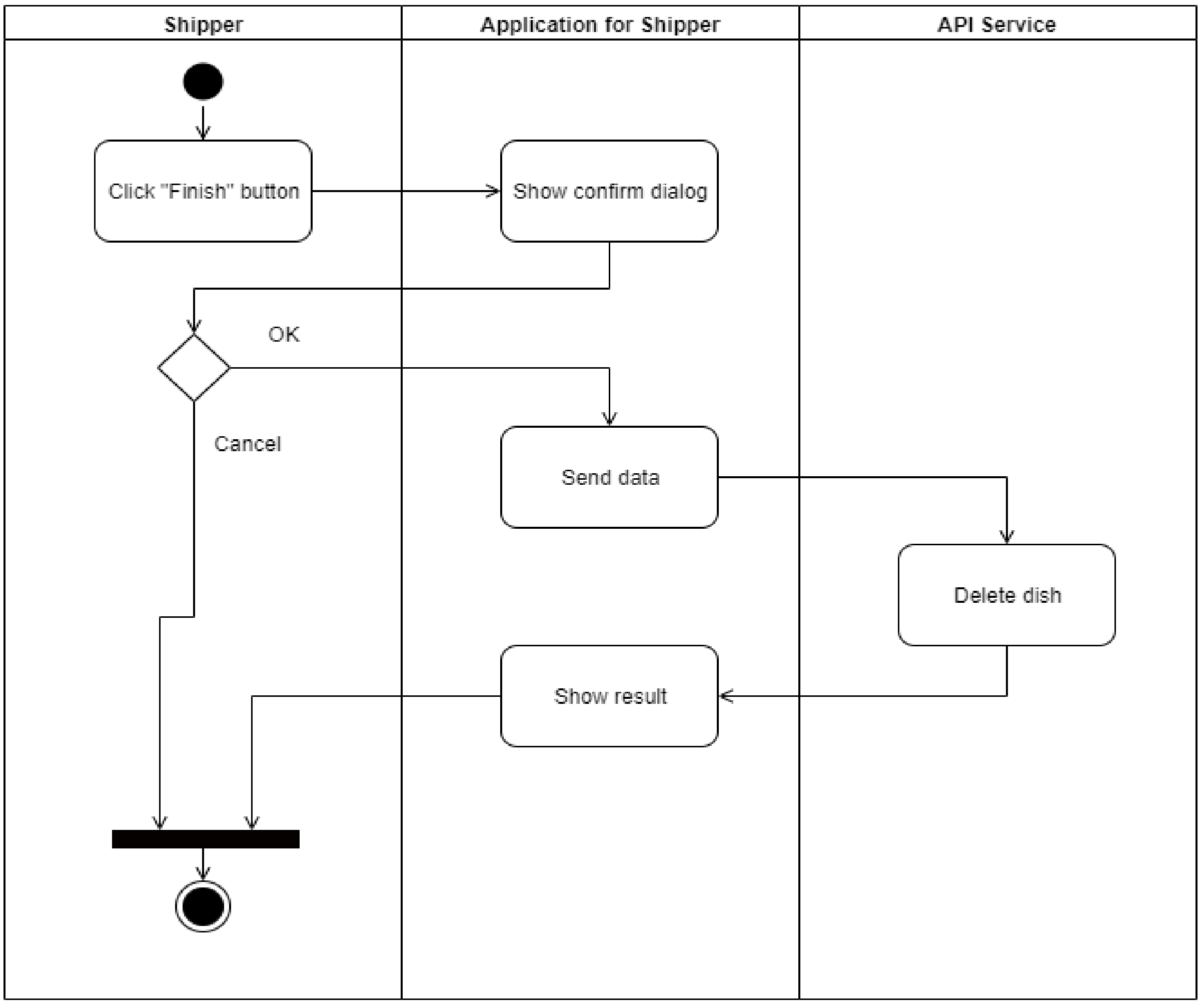


Figure 2.25: Activity diagram – Shipper – Confirm finished order

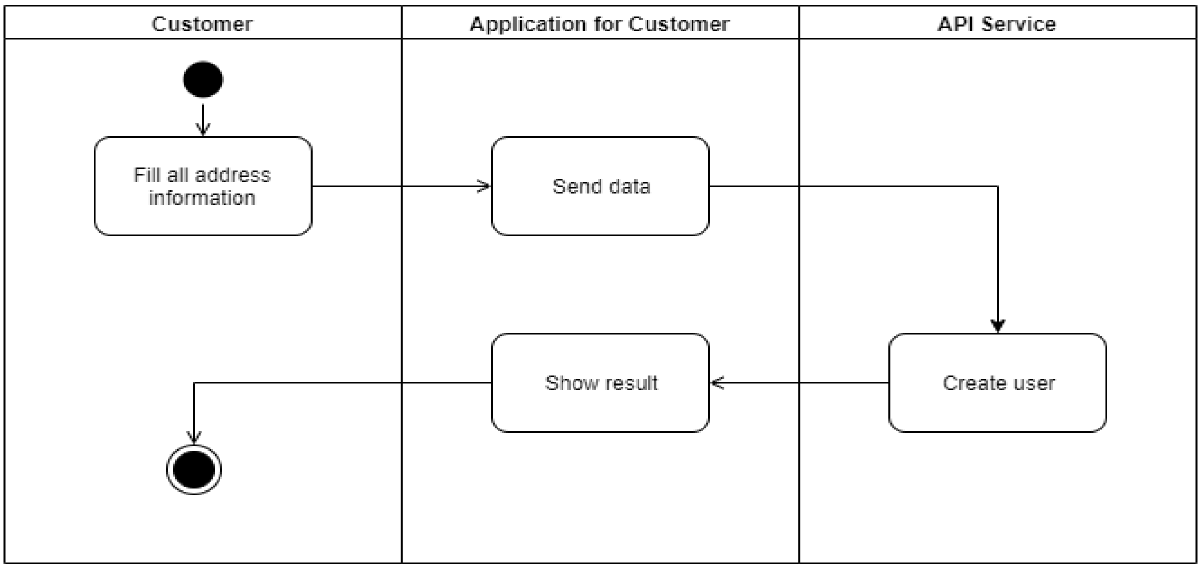


Figure 2.26: Activity diagram – Customer – Add address

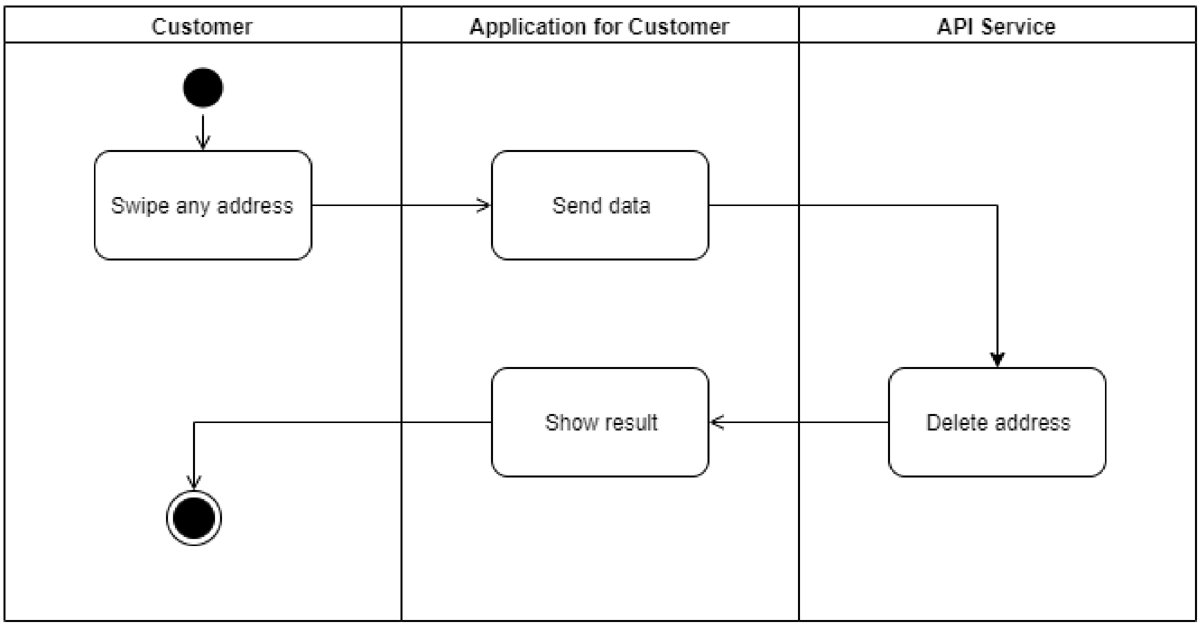


Figure 2.27: Activity diagram – Customer – Delete address

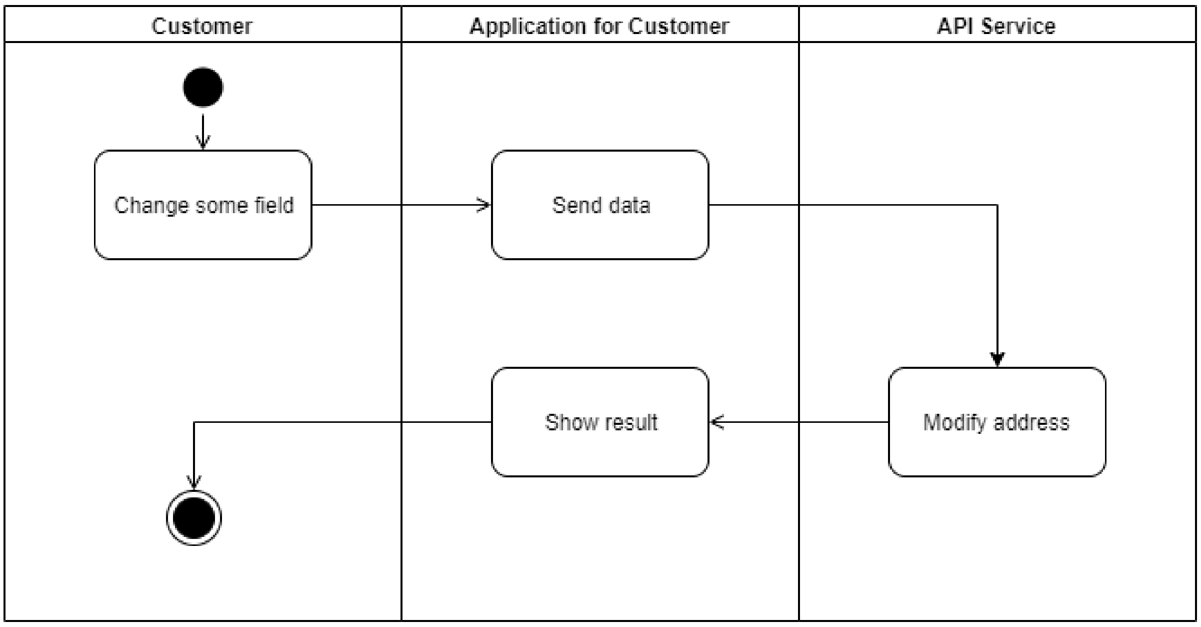


Figure 2.28: Activity diagram – Customer – Modify address

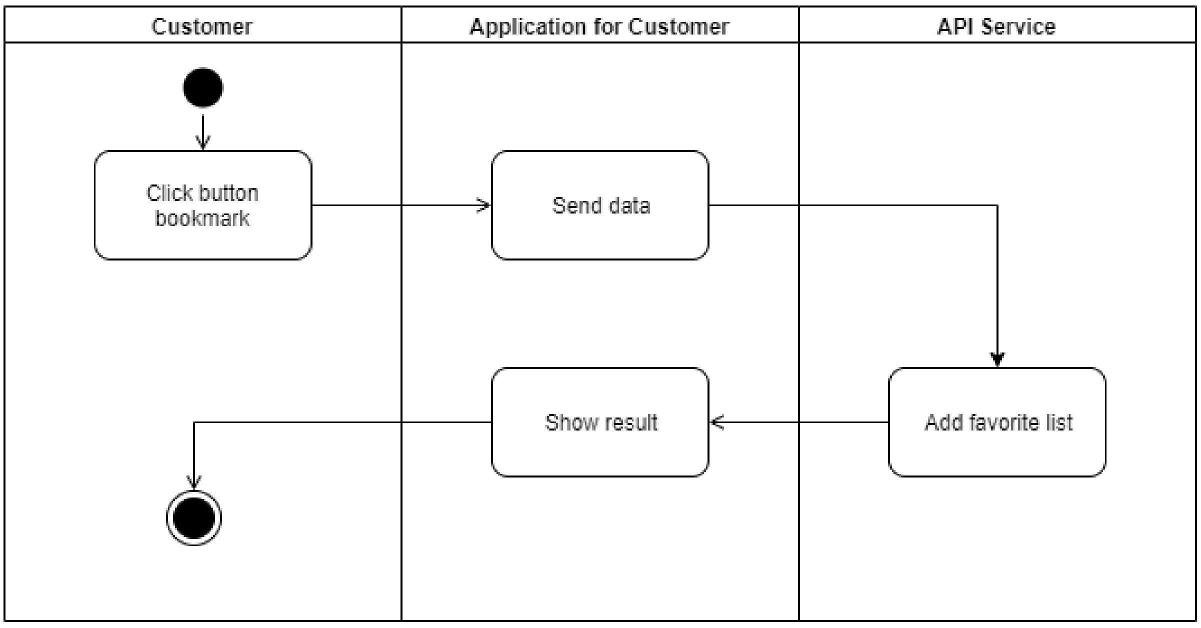


Figure 2.29: Activity diagram – Customer – Add favourite shop

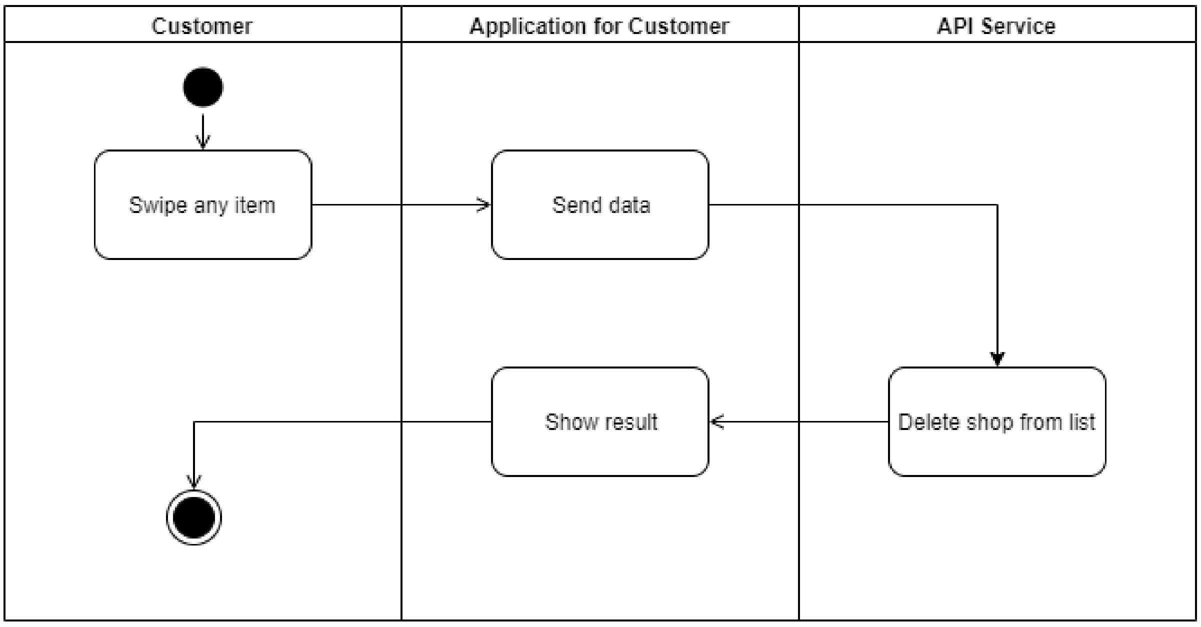


Figure 2.30: Activity diagram – Customer – Delete favourite shop

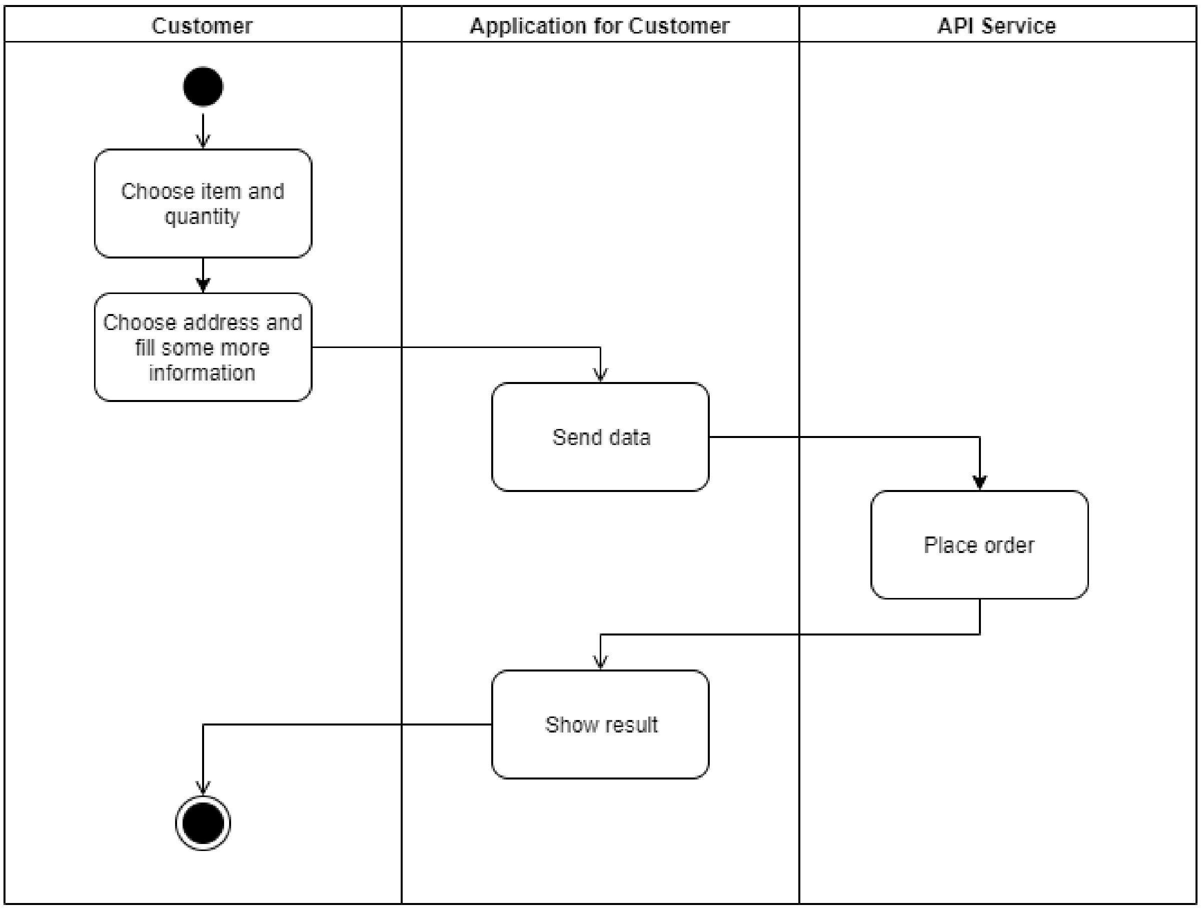


Figure 2.31: Activity diagram – Customer – Place order

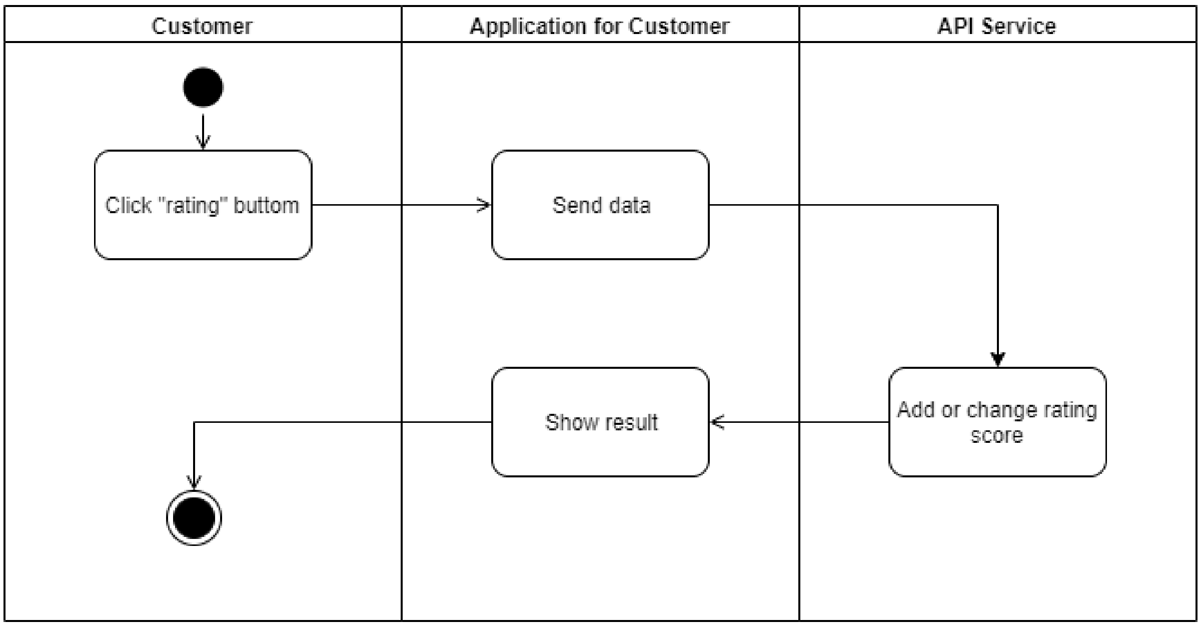


Figure 2.32: Activity diagram – Customer – Rate for shop

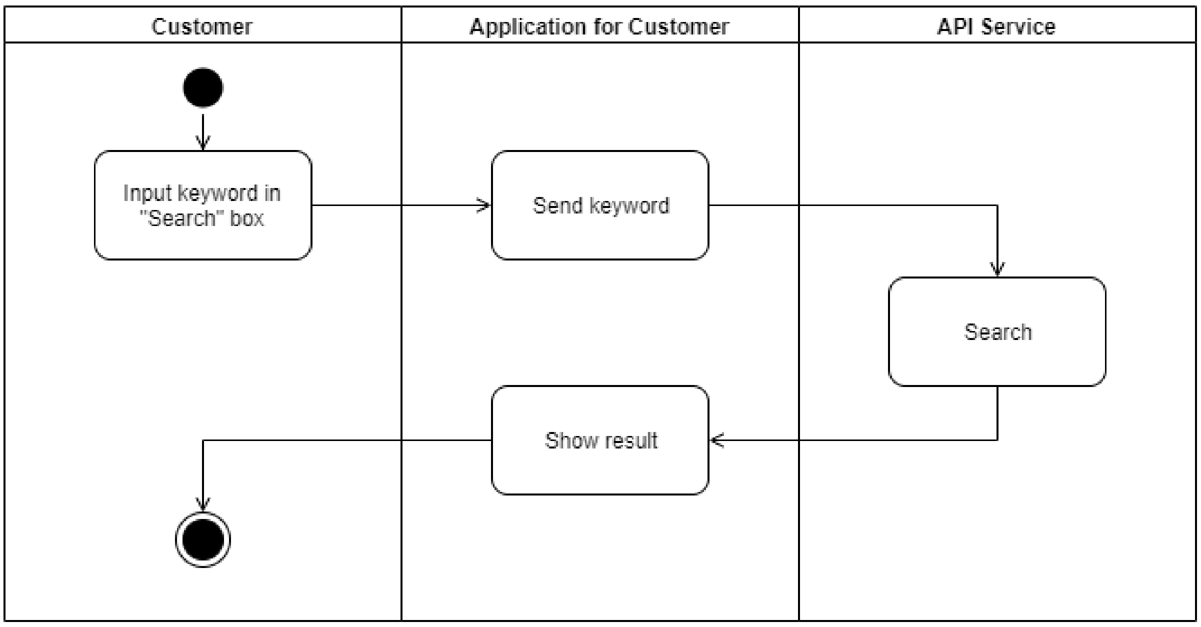


Figure 2.33: Activity diagram – Customer – Search dish

2.3. System design

2.3.1. Class diagram

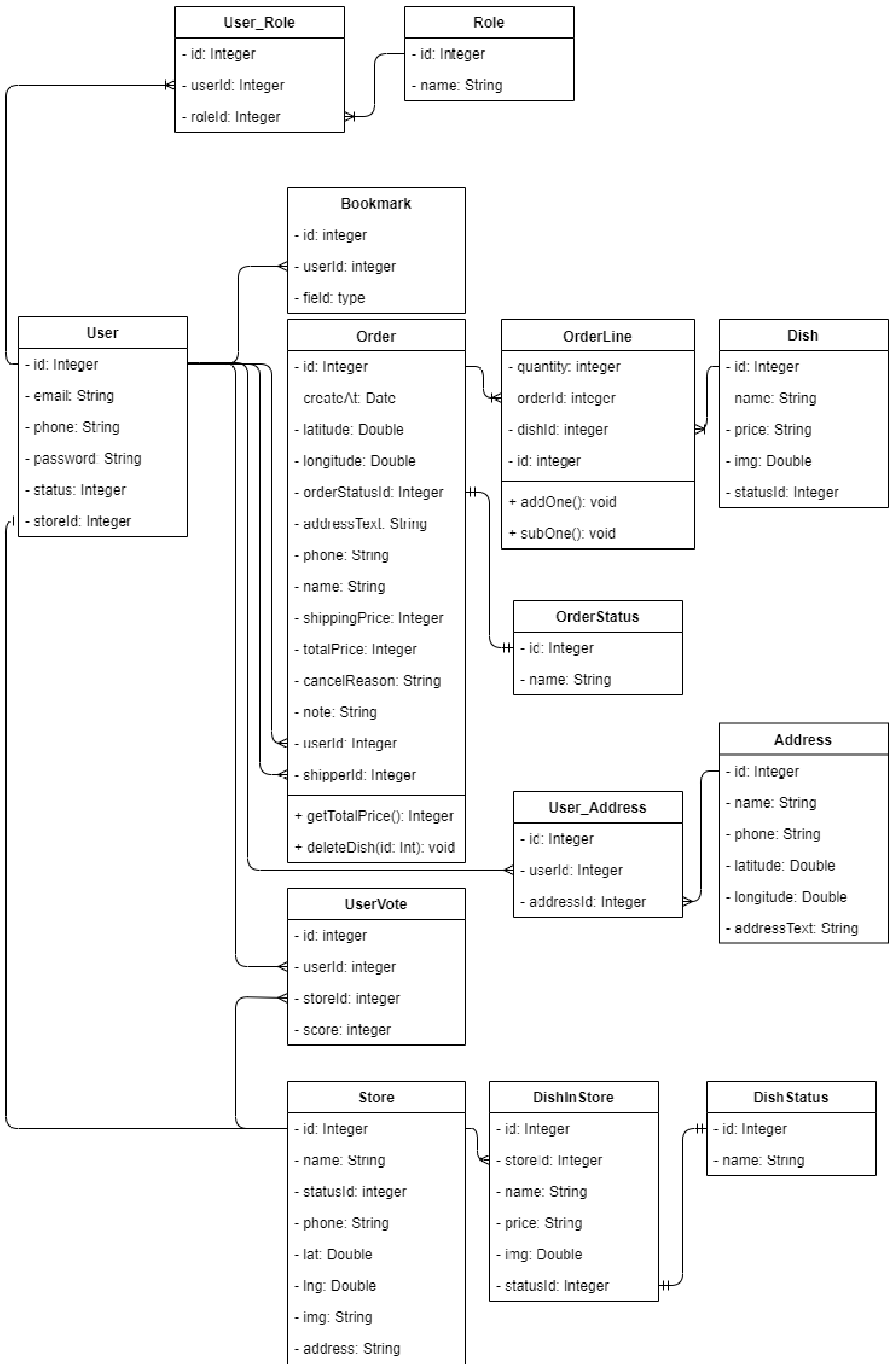


Figure 2.34: Class diagram

2.3.2. Package diagram

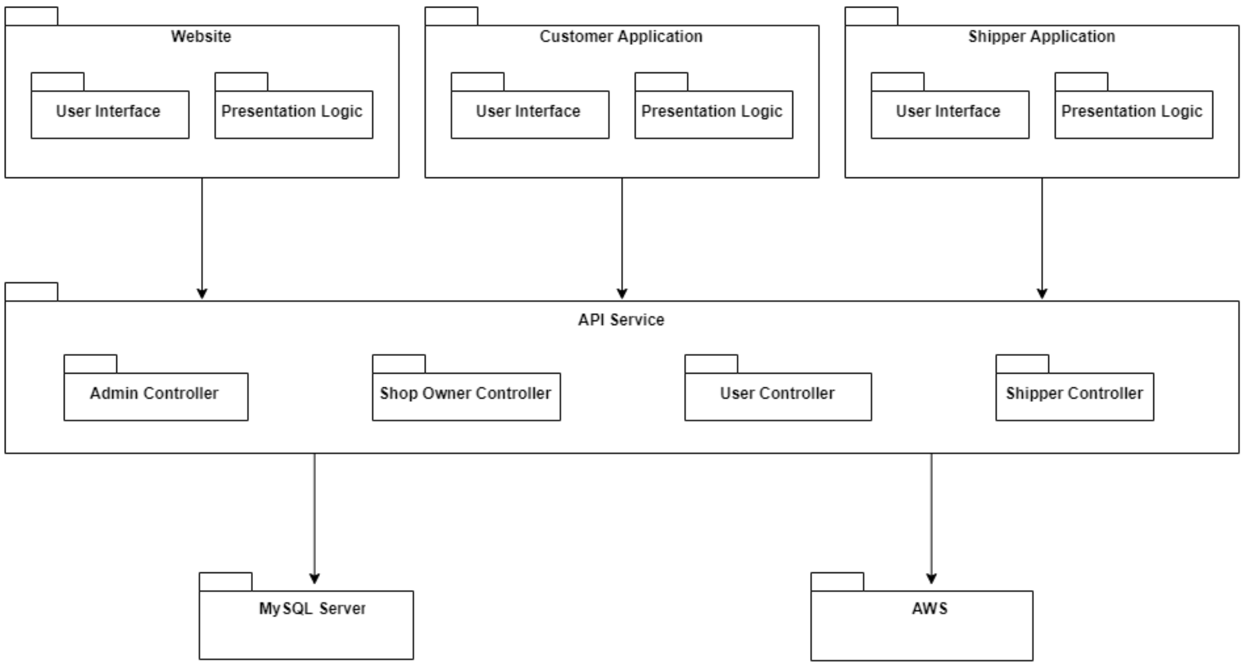


Figure 2.35: Package diagram

This package diagram describes my system architecture. My system uses API service to execute the actions of users with the assistance of MySQL Server to save data, and AWS services.

2.3.3. Database design

1) Description of tables

Table 2.43: Description of table – users

No.	Name	Type	Length	Allow Null	Key
1	id	Integer	11	FALSE	PK
2	email	Varchar	255	FALSE	
3	phone	Varchar	255	TRUE	
4	password	Varchar	255	FALSE	
5	status	Integer	11	FALSE	
6	storeId	Varchar	255	FALSE	FK

Table 2.44: Description of table – bookmarks

No.	**Name**	**Type**	**Length**	**Allow Null**	**Key**
1	id	Integer	11	FALSE	PK
2	userId	Integer	11	FALSE	FK
3	storeId	Integer	11	FALSE	FK

Table 2.45: Description of table – stores

No.	**Name**	**Type**	**Length**	**Allow Null**	**Key**
1	id	Integer	11	FALSE	PK
2	name	Varchar	255	FALSE	FK
3	status	Integer	11	FALSE	FK
4	phone	Varchar	11	FALSE	
5	lat	Float		FALSE	
6	lng	Float		FALSE	
7	img	Varchar	255	FALSE	
8	address	Varchar	255	FALSE	

Table 2.46: Description of table – userVotes

No.	**Name**	**Type**	**Length**	**Allow Null**	**Key**
1	id	Integer	11	FALSE	PK
2	userId	Integer	11	FALSE	FK
3	storeId	Integer	11	FALSE	FK
4	score	Integer	11	FALSE	

Table 2.47: Description of table – productInStores

No.	Name	Type	Length	Allow Null	Key
1	id	Integer	11	FALSE	PK
2	name	Varchar	255	FALSE	
3	price	Integer	11	FALSE	
4	img	Varchar	255	FALSE	
5	statusProductId	Integer	11	FALSE	

Table 2.48: Description of table – addresses

No.	Name	Type	Length	Allow Null	Key
1	id	Integer	11	FALSE	PK
2	name	Integer	11	FALSE	
3	phone	Varchar	255	FALSE	
4	latitude	Double		FALSE	
5	longitude	Double		FALSE	
6	addressText	Varchar	255	FALSE	

Table 2.49: Description of table – orders

No.	Name	Type	Length	Allow Null	Key
1	id	Integer	11	FALSE	PK
2	createAt	Datetime		FALSE	
3	latitude	Double		FALSE	
4	longitude	Double		FALSE	

5	note	Varchar	255	TRUE	
6	cancelReason	Varchar	255	TRUE	
7	totalPrice	Integer	11	FALSE	
8	shippingPrice	Integer	11	FALSE	
9	name	Varchar	255	FALSE	
10	phone	Varchar	255	FALSE	
11	addressText	Varchar	255	FALSE	
12	userId	Integer	11	FALSE	FK
13	shipperId	Integer	11	TRUE	FK
14	statusOrderId	Integer	11	FALSE	FK

Table 2.50: Description of table – orderLines

No.	**Name**	**Type**	**Length**	**Allow Null**	**Key**
1	id	Integer	11	FALSE	PK
2	quantity	Varchar	255	FALSE	
3	orderId	Integer	11	FALSE	FK
4	productId	Integer	11	FALSE	FK

Table 2.51: Description of table – products

No.	**Name**	**Type**	**Length**	**Allow Null**	**Key**
1	id	Integer	11	FALSE	PK
2	name	Varchar	255	FALSE	
3	price	Integer	11	FALSE	
4	img	Varchar	255	FALSE	
5	statusProductId	Integer	11	FALSE	

2) Relationships

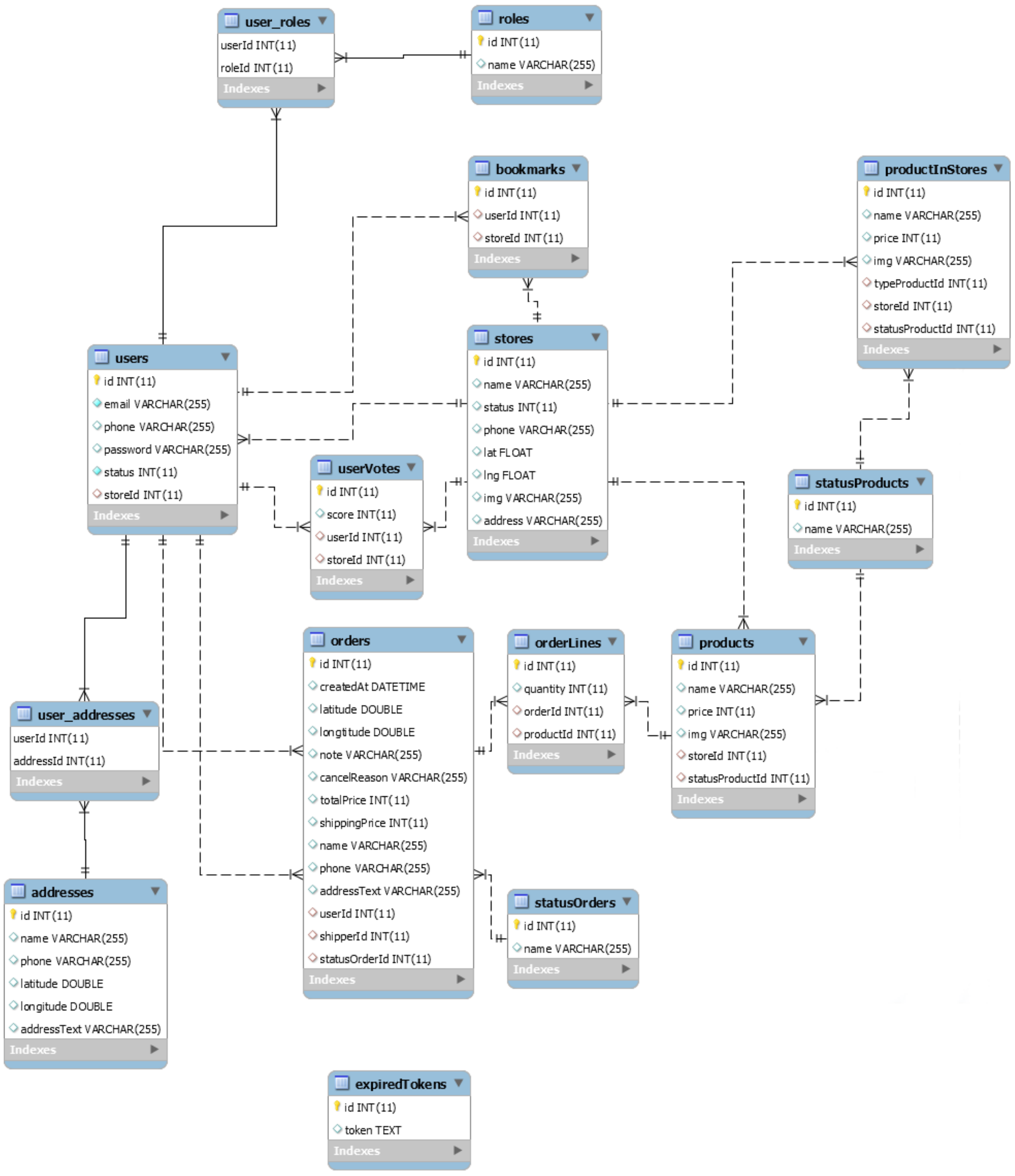


Figure 2.36: Relationship of tables

2.4. Conclusion

This chapter presented the requirements specification that the system could meet the user’s demands. Follow the requirements, the use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. The activity diagrams, draw the activity flow of a system, show the steps and action sequences as well as the interactions between user and user, user and system. Thereby, the overview and the activity streams of the system are fully presented.

Besides, this chapter also describes the system structure, as well as the action sequences for each function. By that, it facilitates the testing phase, the tester can go back to the sequence diagrams to follow the action sequences and create the function tests and the input data as well. Furthermore, it shows the database to help the reader have clearer views of the system. By that, I can consider and evaluate the complexity of this system.

Chapter 3 : IMPLEMENTATION AND RESULT EVALUATION

3.1. Development environment

- **Web service**

The web service is a RESTful API. It is built by Node.js language version 10.15.3 on Express Framework version 4.17, basing on representational state transfer (REST) technology. I use MySQL to store.

- **Web Client**

I used ReactJS to make presentation logic and HTML, CSS / Bootstrap Framework version 4.3.1 to build the user interface.

- **Application**

  I used Android Studio 3.0.1 with Kotlin language to build an application for shipper and an application for customer.

Software Development Tools

I used below tools: Visual Studio Code, Android Studio, MySQL Workbench and Postman, GitHub.

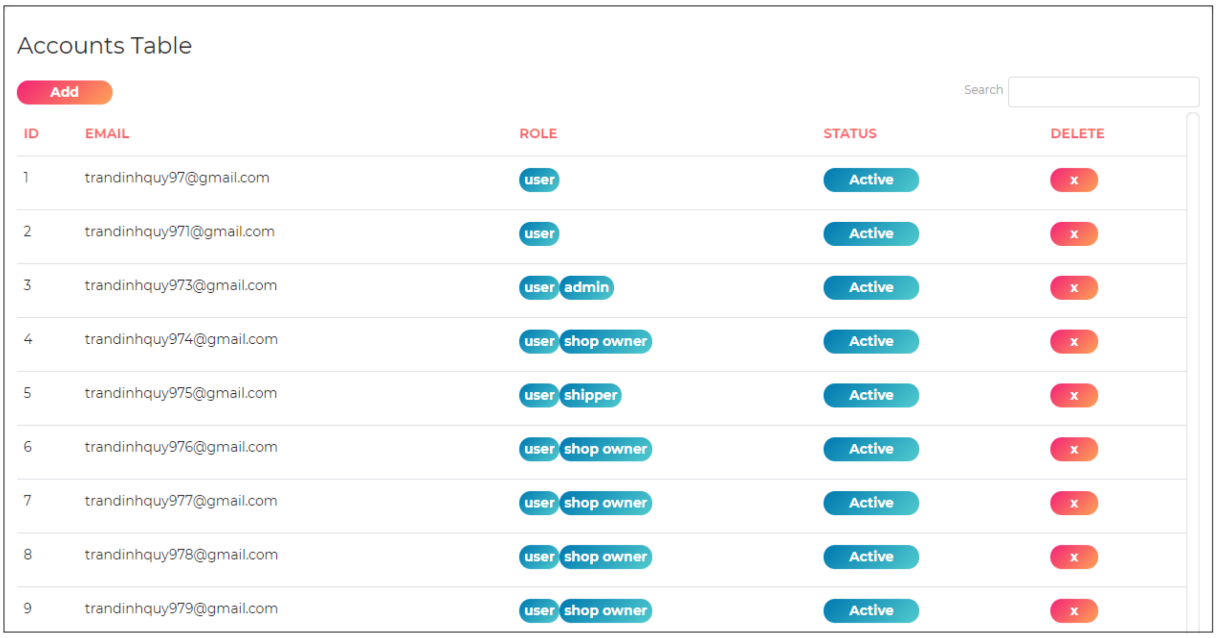
3.2. Demo system's main features

3.2.1. Administrator

The administrator can manage accounts and shops. With each kind of item, the website will give the appropriate managing features such as searching, seeing detail, adding, updating, resetting password and deleting

1) Manage user

The administrator can create a new user, reset password and even delete the user.



Accounts Table

Add

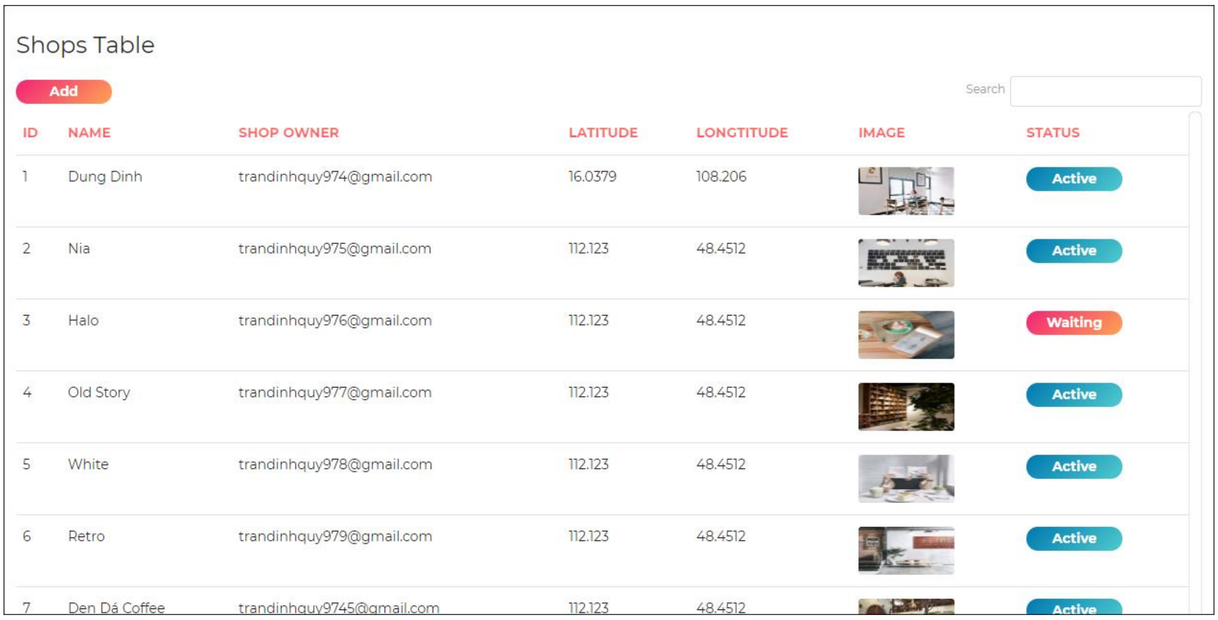
Search

ID	EMAIL	ROLE	STATUS	DELETE
1	trandinhquy97@gmail.com	user	Active	x
2	trandinhquy971@gmail.com	user	Active	x
3	trandinhquy973@gmail.com	user admin	Active	x
4	trandinhquy974@gmail.com	user shop owner	Active	x
5	trandinhquy975@gmail.com	user shipper	Active	x
6	trandinhquy976@gmail.com	user shop owner	Active	x
7	trandinhquy977@gmail.com	user shop owner	Active	x
8	trandinhquy978@gmail.com	user shop owner	Active	x
9	trandinhquy979@gmail.com	user shop owner	Active	x

Figure 3.1: Demonstration – Administrator – Manage user

2) Manage shop

The administrator can create a new shop, change shop information, and even delete the shop.



Shops Table

Add

Search

ID	NAME	SHOP OWNER	LATITUDE	LONGTITUDE	IMAGE	STATUS
1	Dung Dinh	trandinhquy974@gmail.com	16.0379	108.206		Active
2	Nia	trandinhquy975@gmail.com	112.123	48.4512		Active
3	Halo	trandinhquy976@gmail.com	112.123	48.4512		Waiting
4	Old Story	trandinhquy977@gmail.com	112.123	48.4512		Active
5	White	trandinhquy978@gmail.com	112.123	48.4512		Active
6	Retro	trandinhquy979@gmail.com	112.123	48.4512		Active
7	Den Dá Coffee	trandinhquy9745@gmail.com	112.123	48.4512		Active

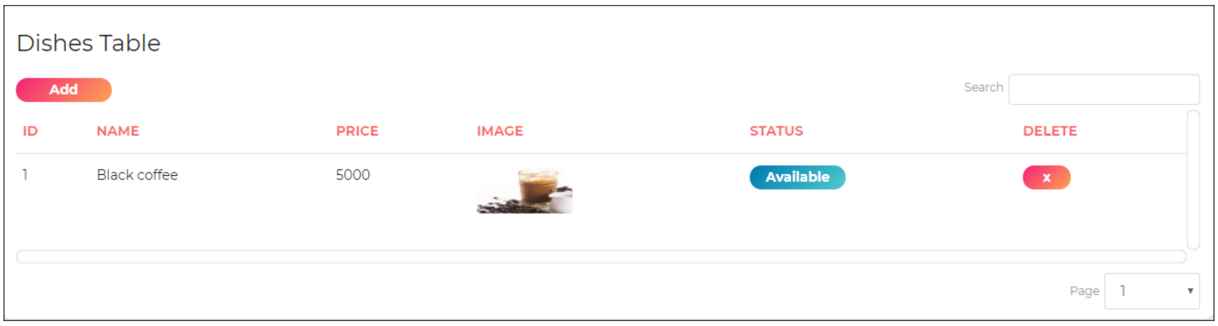
Figure 3.2: Demonstration – Administrator – Manage shop

3.2.2. Shop Owner

The shop owner can manage orders and dishes. With each kind of item, the website will give the appropriate managing features such as searching, seeing detail, adding, updating and deleting…

1) Manage dish

The shop owner can create a new dish, search, edit and even delete the dish.


Dishes Table

Add

Search

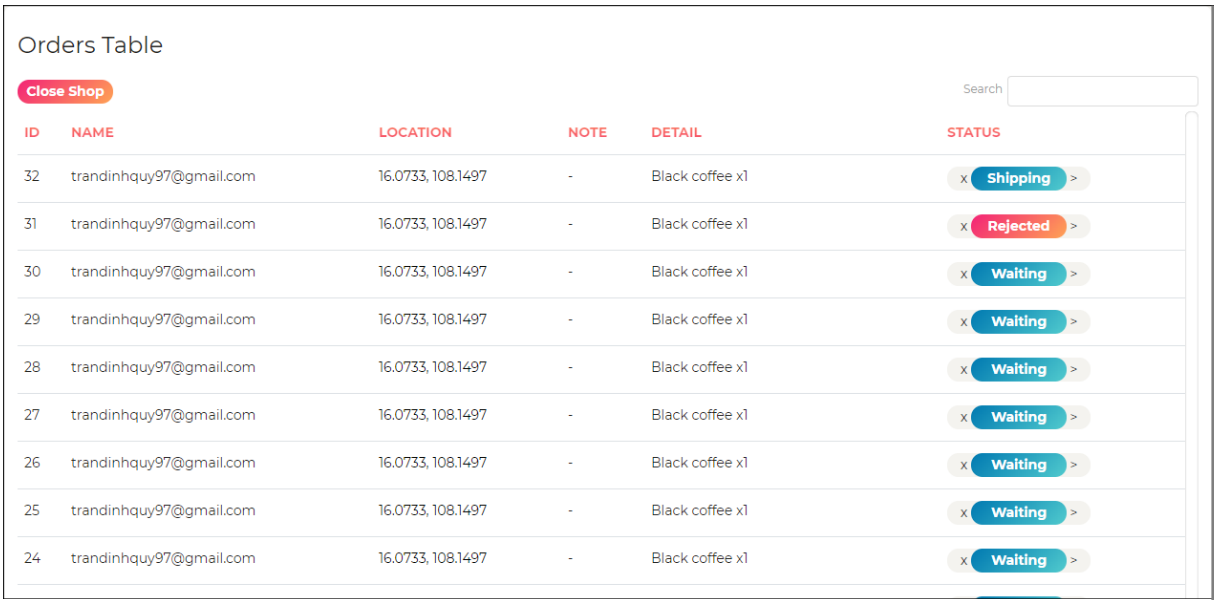
ID	NAME	PRICE	IMAGE	STATUS	DELETE
1	Black coffee	5000		Available	x

Page 1

Figure 3.3: Demonstration – Shop Owner – Manage dish

2) Manage order

Shop owner can find shipper for that order, reject order and close/open shop


Orders Table

Close Shop

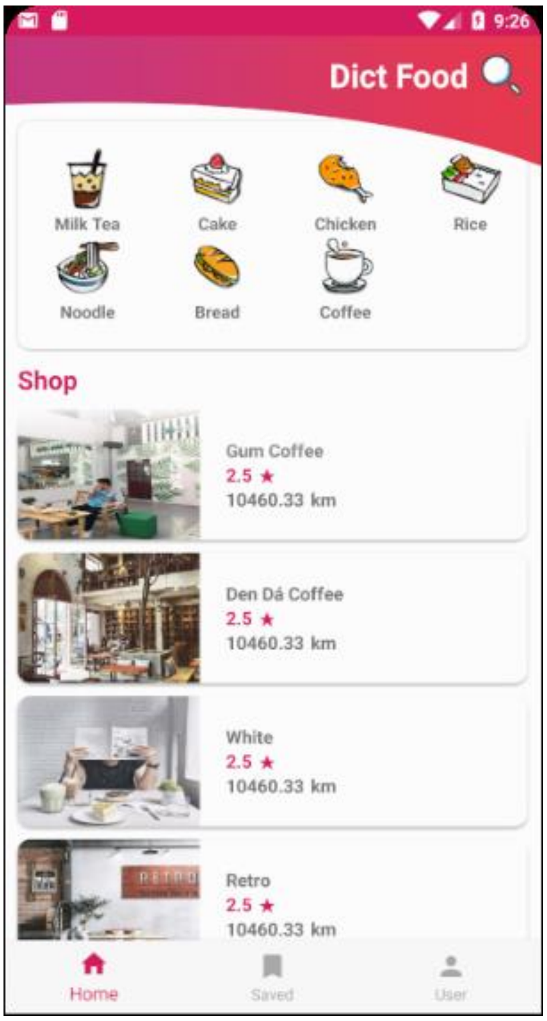
Search

ID	NAME	LOCATION	NOTE	DETAIL	STATUS
32	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Shipping >
31	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Rejected >
30	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
29	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
28	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
27	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
26	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
25	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >
24	trandinhquy97@gmail.com	16.0733, 108.1497	-	Black coffee x1	x Waiting >

Figure 3.4: Demonstration – Shop Owner – Manage order

3.2.3. Customer

Customer can place order, manage address, manage invidual favourite shop …

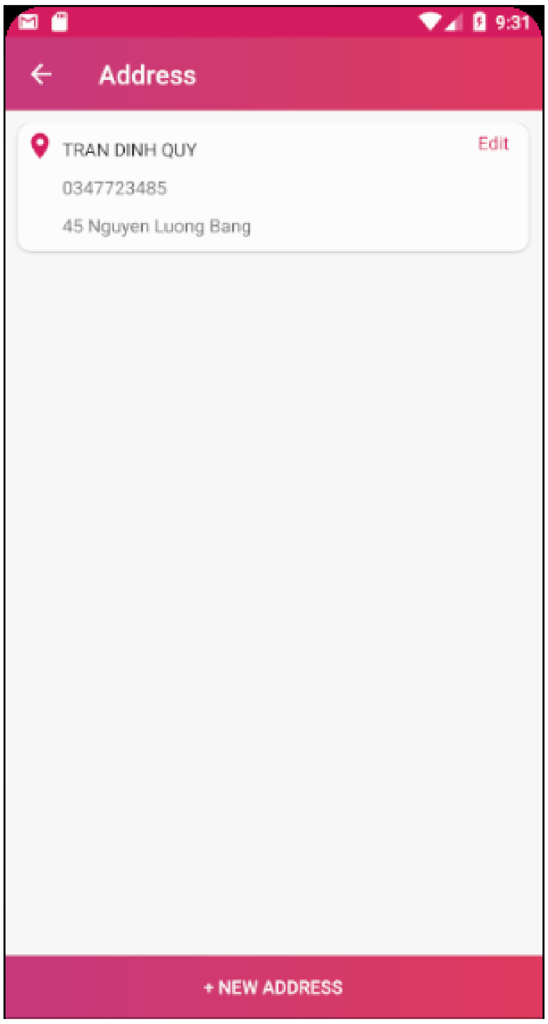


Gum Coffee
2.5 ★
10460.33 km

Den Đá Coffee
2.5 ★
10460.33 km

White
2.5 ★
10460.33 km

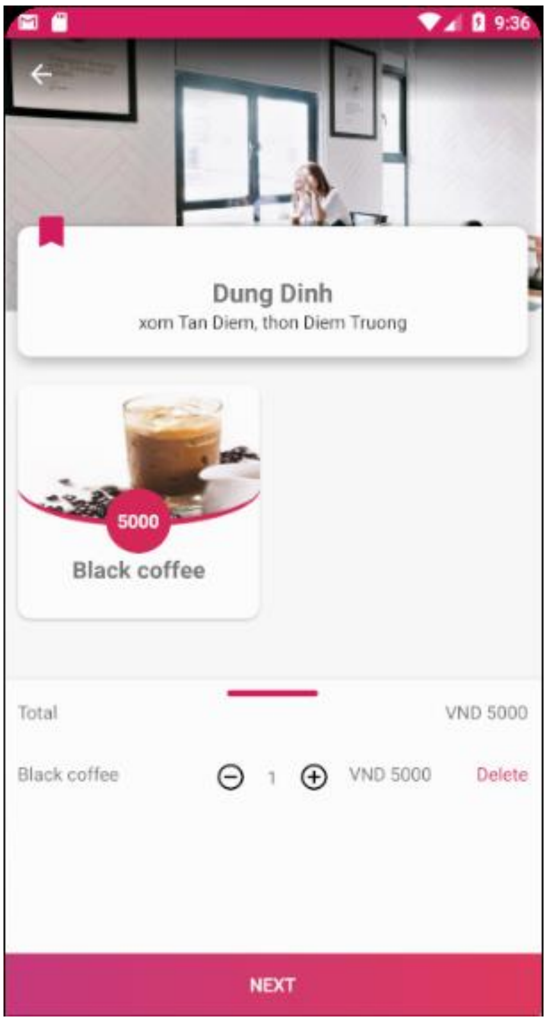
Retro
2.5 ★
10460.33 km



TRAN DINH QUY
0347723485
45 Nguyen Luong Bang

Figure 3.5: Demonstration – Customer – View recommended shop

Figure 3.6: Demonstration – Customer – Manage invidual address



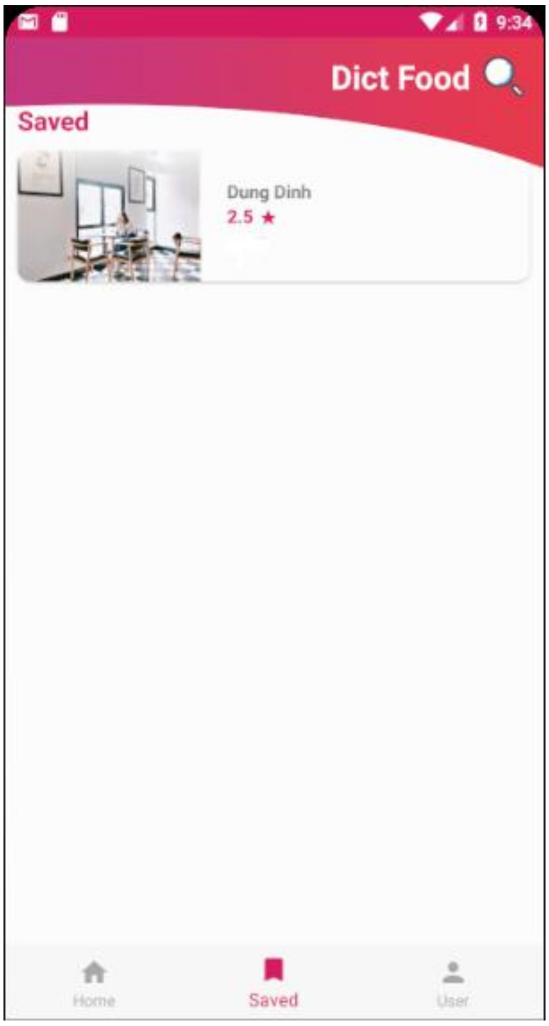
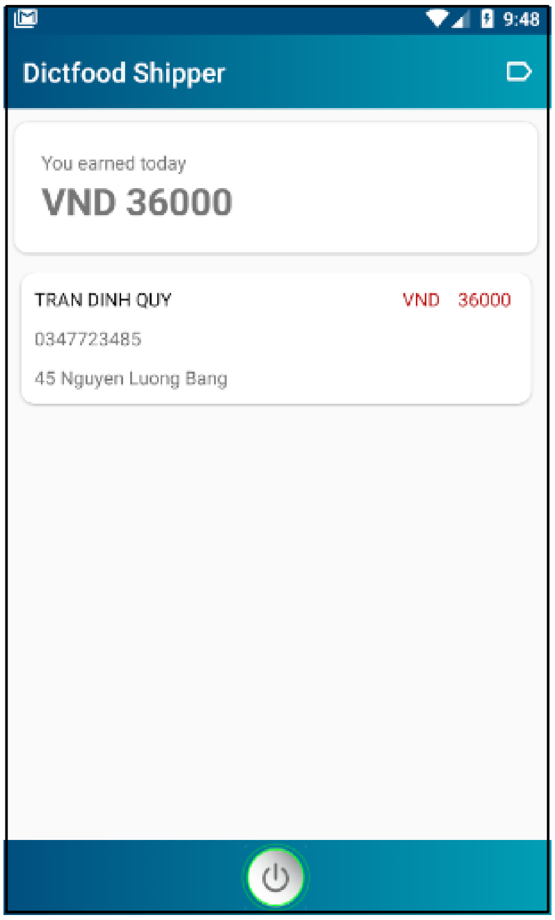


Figure 3.7: Demonstration – Customer – Manage favorite shop

Figure 3.8: Demonstration – Customer – Place order

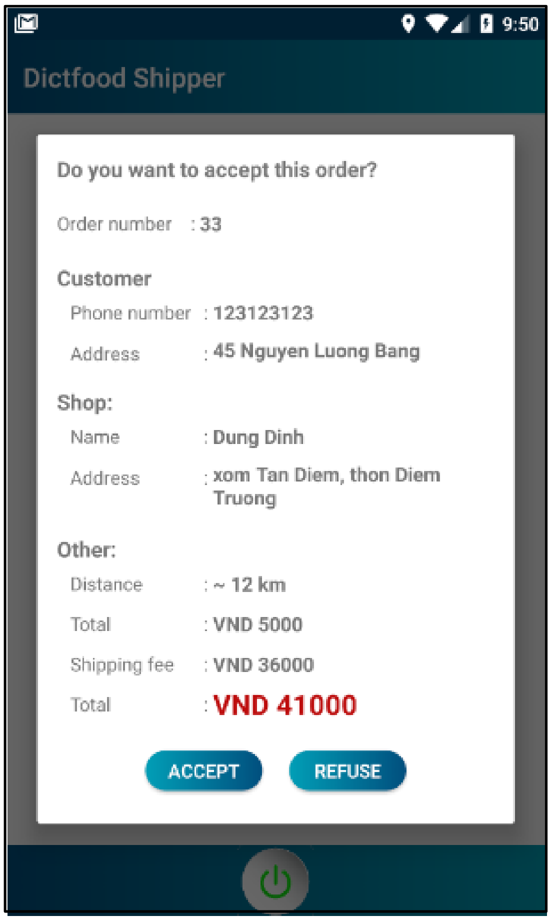
3.2.4. Shipper

Shipper can change status of device (on/off), accept/reject order, manage current received order and view history…



You earned today
VND 36000

TRAN DINH QUY VND 36000
0347723485
45 Nguyen Luong Bang



Do you want to accept this order?

Order number : 33

Customer

Phone number	: 123123123
Address	: 45 Nguyen Luong Bang

Shop:

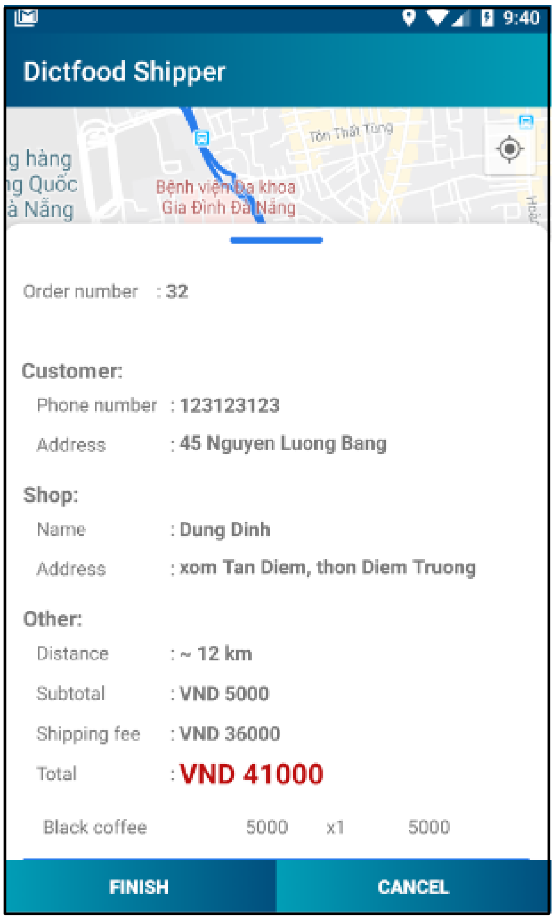
Name	: Dung Dinh
Address	: xom Tan Diem, thon Diem Truong

Other:

Distance	: ~ 12 km
Total	: VND 5000
Shipping fee	: VND 36000
Total	: VND 41000

Figure 3.9: Demonstration – Shipper – Change device status and view history

Figure 3.10: Demonstration – Shipper – Accept order



Order number	: 32

Customer:

Phone number	: 123123123
Address	: 45 Nguyen Luong Bang

Shop:

Name	: Dung Dinh
Address	: xom Tan Diem, thon Diem Truong

Other:

Distance	: ~ 12 km
Subtotal	: VND 5000
Shipping fee	: VND 36000
Total	: VND 41000

Black coffee	5000	x1	5000

Figure 3.11: Demonstration – Shipper – Manage current received order

3.3. Experiment result evaluation

3.3.1. Advantages

Basically, “Building a system for ordering foods and beverages promptly” meets the needs of companies in the scope of the senior project.

In terms of UX/UI, my system provides quite simple and familiar UI for users to interact and get information.

In my system, Administrator can manage accounts and shops information, Shop Owner can manage dishes and orders which are belong to that shop, Shipper can get order information and manage that order. And customer can get the recommended dishes through recommendation algorithm, place order and manage some individual information.

Especially, my system resolves two big problems, which still exist in some current delivery system. First, it can reduce the time when shipper go to the shop, and shop doesn’t need to wait for shipper coming, shop can start to make that order when find out a shipper for that order. Second, shipper can know whether that order is available or not by confirmation from shop.

3.3.2. Disadvantages

Besides the advantages mentioned above, my system still has some disadvantages:

- Need to get feedback from real users to apply to real situation.
- My system requires internet connection continuously.
- The synchronization of the system is not so stable and the performance is not too high.

CONCLUSION

1. Achievements

After building this system, I could understand how to make an application using ReactJS, Node.js, MySQL, etc and how to interact between client, database, and server. I also learn how to build a RESTful API with Node.js/Express framework. Besides, I gain more knowledge of the structure of Node.js, its workflow as well as the framework, API and how to apply them to my project.

I also improved my skills a lot, including research skills, technical skills, presentation skills, English and many other soft skills.

About the project, I have already followed the plan which was created in the first week of graduation project duration and implemented all functions in that plan. Besides, I also try to make the system easy to use. Last but not least, I made this system that helps users can know where to go out, where to buy foods and beverages, helps shop increase profit by raise the number of orders, secure for shippers. The system includes a lot of features:

- Manage accounts and shops.
- Manage dishes and orders.
- Suggest dishes for user.
- Place order.
- Find the nearest shipper for order.

Although certain results have been achieved, there are still many challenges, risks, and issues with products:

- The algorithm needs to be improved to be more exact.
- The system requires internet connection continuously.
- UX/UI is not really good for the user use it for the first time.

2. Future plan

With some disadvantage I have already mentioned in the evaluation part, I am aware of some development directions for this system in the future, such as:

- Improve UX/UI.
- Implement E–wallet to get profit from this system.
- Add more extensive functions for managing

REFERENCES

[1] https://nodejs.org/en/docs/

[2] https://www.javatpoint.com/nodejs–features

[3] https://developer.mozilla.org/en–US/docs/Learn/Serverside/Express_Nodejs

[4] https://reactjs.org/

[5] https://getbootstrap.com/docs/4.3/getting–started/introduction/

[6] https://dev.mysql.com/doc/refman/8.0/en/what–is–mysql.html

[7] https://sequelize.org/

[8] https://medium.com/@prajramesh93/getting–started–with–node–express–and–mysql–using–sequelize–ed1225afc3e0

[9] https://socket.io/

[10] https://kotlinlang.org/

[11] https://codelabs.developers.google.com/