

**CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY**



**THE THESIS IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)**

**FREIGHT MANAGEMENT
SYSTEM WITH PYTHON, JAVASCRIPT**

**Advisor:
Dr. Truong Quoc Dinh**

**Student:
Huỳnh Quan Nhật Hào
Student ID: B1809687
Course: 44**

Can Tho, November 2021

**CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY**



**THE THESIS IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)**

**FREIGHT MANAGEMENT
SYSTEM WITH PYTHON, JAVASCRIPT**

**Advisor:
Dr. Truong Quoc Dinh**

**Student:
Huỳnh Quan Nhật Hào
Student ID: B1809687
Course: 44**

Can Tho, November 2021

This image shows a full page of white paper with horizontal dashed lines, typical of primary-ruled notebook paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

(Instructor sign and write full name)

ACKNOWLEDGMENTS

First of all, I would like to thank the teacher of the College of Information and Communication Technology for guiding and teaching me during my study at Can Tho University. Especially Dr. Truong Quoc Dinh, who enthusiastically guided me throughout the process of researching and finding out. In addition, I also thank to my friends who supported and helped me during my research.

I would like to thank my family for always believing and supporting me in my studies and life.

Although I have tried very hard to complete my topic, but due to limited knowledge, my topic still has many incomplete points. I look forward to receiving feedback from teachers to improve the topic.

Finally, I would like to wish you a lot of health, happiness and success in your path. Best regards!

Can Tho, November 2021

Student

Huỳnh Quan Nhật Hào

TABLE OF CONTENTS

INSTRUCTOR’S COMMENTS	1
ACKNOWLEDGMENTS	2
LIST OF FIGURES	6
LIST OF TABLES	7
ABSTRACT.....	9
INTRODUCTION	11
1. Project purposes	11
2. Project methodology and approach.....	11
3. Project content	11
CONTENTS.....	12
CHAPTER i. REQUIREMENT OF SPECIFICATION	12
1. Order Management Model.....	12
1.1. Order Management Definition.....	12
1.2. Typical order management process	12
2. Web technology	12
2.1. Django Framework	12
2.2. ReactJS Framework	13
3. Other necessary libraries and libraries.....	14
3.1. React Native framework	14
3.2. Leaflet library	14
CHAPTER ii. DESIGN CONCEPTION	15
1. Description of the system	15
2. Main functions of the system.....	15
2.1. Functions of mobile application	15
2.1.1. Register	15
2.1.2. Login.....	16
2.1.3. Log out.....	16
2.1.4. Account management	16
2.1.5. Create new orders	17
2.1.6. Manage order information	18
2.1.7. Submit a request for an order.....	18
2.1.8. Cash flow control.....	19
2.2. Functions of web application.....	19
2.2.1. Register	19

2.2.2. Login.....	20
2.2.3. Log out.....	21
2.2.4. Account management	21
2.2.5. Orders available on the system management.....	22
2.2.6. Receive order for delivery	22
2.2.7. Track current order requests	23
2.2.8. Track current order information	23
2.2.9. Update the current location of the order	24
2.2.10. Update the delivery status of the current order.....	25
2.2.11. Confirmation of payment.....	25
CHAPTER iii. IMPLEMENT CONCEPTION	27
1. Database design	27
2. Server design with Django framework	32
2.1. Configure the setting file	32
2.2. Configure model file.....	33
2.3. Create processing file for customer application	34
2.4. Create processing file for driver website application	36
2.5. Create file to define urls.....	37
4. Mobile application for customer.....	37
4.1. Log-in page.....	38
4.2. Register page.....	38
4.3. Overview page	39
4.4. Personal information page	39
4.5 Money-flow page.....	40
4.6. Order information page.....	41
4.7. New order page.....	41
4.8. Order detail page.....	42
4.9. Map page.....	43
5. Web application for driver.....	44
5.1. Log-in page.....	44
5.2. Register page.....	45
5.2. Home page	45
5.3. Available orders on the system page	46
5.4. Personal information of the driver page	47
5.5. Current order page	47
CHAPTER iv. TESTING AND REVIEWING	50

1. Testing objectives	50
2. Testing contents	50
2.1. Details of the testing plan	50
2.1.1. Functions tested	50
2.1.3. Testing criteria	53
2.2. Testing environment	53
2.3. Mobile application test cases	53
2.3.1. Log in.....	53
2.3.2. Register	54
2.3.3. Edit account information	54
2.3.4. Create a new order	55
2.3.5. Submit a request for an order.....	55
2.3.6. View order list	55
2.3.7. View the information of a particular order	56
2.3.8. Track the location of an order on a map	57
2.4. Web application test cases	57
2.4.1. Log in.....	57
2.4.2. Register	58
2.4.3. Edit account information	58
2.4.4. View overview page	59
2.4.5. View current order details.....	59
2.4.6. View current order location on a map	60
2.4.7. View the details of an order available	60
2.4.8. Receive order	61
2.4.9. Update geographic coordinates.....	61
2.4.10. Update the status of an order	62
2.4.11. Confirmation of payment.....	62
CONCLUSION.....	64
1. Summary of results	64
2. Future works	64
REFERENCES	65

LIST OF FIGURES

Figure 1: Database design diagram of the system	27
Figure 2: Log-in page of mobile app	38
Figure 3: Register page of mobile app	39
Figure 4: Personal information page of mobile application	40
Figure 5: Order information of mobile application	41
Figure 6: New order page	42
Figure 7: Order details page	43
Figure 8: Map page	44
Figure 9: Log-in page of website	45
Figure 10: Available orders on the system page	46
Figure 11: Direction map	47

LIST OF TABLES

Table 1: Description of registration function for customers.....	16
Table 2: Description of login function for customers.....	16
Table 3:Description of log out function for customers.....	16
Table 4: Description of account management function for customers	17
Table 5: Description of creating new orders function for customers	18
Table 6: Description of managing order information new orders function for customers ..	18
Table 7: Description of submitting a request for an order new orders function for customers	19
Table 8:Description of cash flow control function for customers	19
Table 9: Description of register function for drivers	20
Table 10: Description of login function for drivers.....	21
Table 11: Description of log out function for drivers.....	21
Table 12: Description of account management function for drivers	22
Table 13: Description of orders available on the system management function for drivers	22
Table 14: Description of receiving order for delivery function for drivers	23
Table 15: Description of tracking current order requests function for drivers	23
Table 16: Description of tracking current order information function for drivers	24
Table 17: Description of updating the current location of the order function for drivers ..	25
Table 18: Description of updating the delivery status of the current order function for drivers	25
Table 19: Description of confirmation of payment function for drivers	26
Table 21: Database description table – RequestOptions.....	27
Table 22: Database description table – Request	28
Table 23: Database description table – ShipOption.....	28
Table 24: Database description table – OrderStatus.....	28
Table 25: Database description table – InstanceAddress.....	28
Table 26: Database description table – LocationUpdate	29
Table 27: Database description table – Customer	31
Table 28: Database description table – Driver.....	31
Table 29: Database description table – Order.....	32
Table 30: Description of list of server's urls	37
Table 31: Tested functions of mobile application table.....	51
Table 32: Tested functions of web application table	53
Table 33: Test log in moblie application function.....	54
Table 34: Test register mobile application function	54
Table 35: Test edit account information mobile application function.....	54
Table 36: Test create a new order mobile application function.....	55
Table 37: Test bubmit a request for an order mobile application function.....	55
Table 38: Test view order list mobile application function.....	56
Table 39: Test view the information of a particular order mobile application function.....	57
Table 40: Test track the location of an order on a map mobile application function	57
Table 41: Test log-in function	58
Table 42: Test register function	58
Table 43: Test edit account information function.....	59
Table 44: View overview page function.....	59

Table 45: Test view current order details	60
Table 46: Test view current order location on a map function	60
Table 47: Test View the details of an order available function	61
Table 48: Test Receive order function.....	61
Table 49: Test update geographic coordinates function	62
Table 50: Test update the status of an order function.....	62
Table 51: Test confirmation of payment function	63

ABSTRACT

Information technology is increasingly developing and has an important and indispensable role in modern life. Humans are creating more and more systems capable of recognizing and processing tasks automatically, which service the benefit of humans. In recent years, one of the systems that has received the most attention is the Freight management system. Although the traditional form of Freight management system has appeared for a long time, it is no longer suitable with the needs of people due to the long processing time, sometimes during peak hours, causing congestion.

Therefore, to solve the above problem, I decided to do the topic: " Freight management system ". To solve the above problem, I used Leaflet API specifically designed for ReactJS and React Native frameworks to get geo-coordinates, as well as give directions to drivers. The application is written in Python, Javascript language and runs on Windows 10 operating system. At the end, result is that I have will build a system of two applications, one application for customers using for delivery services, an application for drivers to manage and ship orders on the system, and finally. The same is the page for the manager integrated in the django framework to manage the system's information.

TÓM TẮT

Công nghệ thông tin ngày càng phát triển và có vai trò hết sức quan trọng không thể thiếu trong cuộc sống hiện đại. Con người ngày càng tạo ra nhiều những hệ thống có khả năng nhận biết và xử lý các công việc một cách tự động, phục vụ lợi ích cho con người. Trong những năm gần đây, một trong những hệ thống nhận được sự quan tâm nhất, đó chính là hệ thống quản lý việc vận chuyển hàng hóa. Dù hình thức quản lý vận chuyển hàng hóa truyền thống đã xuất hiện rất lâu, nhưng hiện nay đã không còn đáp ứng đủ như cầu của con người do thời gian xử lý rất lâu, đôi khi trong giờ cao điểm còn gây quá tải.

Vì vậy để giải quyết vấn đề trên em quyết định thực hiện đề tài: “Hệ thống quản lý vận tải hàng hóa”. Để giải quyết vấn đề trên, em sử dụng API Leaflet được thiết kế đặc biệt cho ReactJS và React Native để lấy tọa độ địa lý, cũng như chỉ đường cho các tài xế. Ứng dụng được viết bằng ngôn ngữ Python, Javascript và chạy trên hệ điều hành Windows 10. Cuối cùng, kết quả là em có sẽ xây dựng một hệ thống gồm hai ứng dụng, một ứng dụng dành cho khách hàng sử dụng dịch vụ giao hàng, một ứng dụng dành cho tài xế để quản lý và vận chuyển đơn hàng trên hệ thống, và cuối cùng là trang dành cho người quản lý được tích hợp trong Django framework để quản lý thông tin của hệ thống.

INTRODUCTION

1. Project purposes

With the Leaflet library and two programming languages Python (Django framework) and Javascript (ReactJS frameworkd and React Native framework) build a freight management system, helping users to access shipping easily. as well as the interaction between the driver and the customer through two applications in the system.

2. Project methodology and approach

The thesis topic on managing data in freight management, calculating prices based on distance, as well as finding the shortest path between two geographic coordinates.

3. Project content

- **Introduction:** An overview of the thesis: an introduction to the topic, research methods and layout of the thesis.
- **Content part:** The content of the thesis is divided into 3 chapters
 - + Chapter 1: Requirements of specification
 - + Chapter 2: Design conception
 - + Chapter 3: Implement conception
 - + Chapter 4: Testing and reviewing
- **Conclusion:** Present the results achieved and the development direction of the system

CONTENTS

CHAPTER I. REQUIREMENT OF SPECIFICATION

1. Order Management Model

1.1. Order Management Definition

Order management is the process of receiving, tracking, and fulfilling customer orders. The order management process begins when the order is placed and ends when the customer receives their package.

In addition to tracking order status from processing to shipping, order management also manages customer data, processes, and partnerships needed to fulfill those orders.

1.2. Typical order management process

Receive orders - When customers place an online order, the business must confirm their order and process their payment. Orders will be placed from many different places, at different times, and on every existing business channel. All relevant information such as order details, delivery address, payment method, etc. It will be sent to the management system for processing. If a product is out of stock, the company must cancel the order or save it as a pre-order. After that, the business owner should quickly buy new inventory from the supplier and notify the customer.

Pick up orders - At this stage, the order has been received by the warehouse and ready to be fulfilled. Based on the information of the orders, the warehouse staff will be responsible for selecting the products accurately and quickly. The inventory arrangement significantly affects the time it takes to execute each order. After the products have been selected in full and accurately according to the list, they will be shipped to the packing station.

Pack orders - All items in an order will be packed into boxes, barrels or pallets, depending on the size and quantity of the product. Employees need to use appropriate and safe packaging materials, ensuring that the goods are intact when reaching consumers. For example, fragile goods, such as glass, should be packed with bubble wrap to avoid damage during transportation. Packaging not only helps protect the condition of goods from accidents that occur during the fulfillment of orders, but also helps optimize the load according to the length, width, and height of the vehicle.

Delivery orders - After the order has been packed correctly, the staff will deliver the goods to the delivery point. They will print the shipping label with the customer's address and attach the sales invoice to the package. Customers can choose the carrier that suits their needs and priorities (delivery time, shipping cost,...). In addition, information about the status of orders will also be updated on the management system, and shared with customers via app, email, or phone message.

2. Web technology

2.1. Django Framework

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and

active community, great documentation, and many options for free and paid-for support. Django helps you write software that is:

Complete - Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile - Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Secure - Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Scalable - Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable - Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable - Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

2.2. ReactJS Framework

React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI

components. React was first created by Jordan Walke, a software engineer working for Facebook. React first deployed on Facebook’s newsfeed in 2011 and on Instagram.com in 2012.

React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template. It can be used with a combination of other JavaScript libraries or frameworks, such as Angular JS in MVC. React JS is also called simply to React or React.js.

3. Other necessary libraries and libraries

3.1. React Native framework

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user’s location.

React Native currently supports both iOS and Android, and has the potential to expand to future platforms as well. In this book, we’ll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some anecdota: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.

3.2. Leaflet library

Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 39 KB of JS, it has all the mapping features most developers ever need.

Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.

CHAPTER II. DESIGN CONCEPTION

In this chapter, I will focus on the detailed design of the functions of the system, including the design of the orientation of the system, and the necessary functions of the system.

1. Description of the system

The system will include three main applications: a mobile app for customers, a website for drivers, and a manager management page integrated into the Django framework.

Mobile application for customers, the system will allow customers to register for delivery for orders with the attributes that the system requires. Customers can also monitor the status and location of the order is constantly updated on the system.

About the website application for drivers, it will allow the drivers to track the orders available on the system, receive orders, deliver based on the instructions on the system, and confirm the status of the order.

The manager's page integrated into the Django Framework will help managers control all the information on the system, including disputes about customers, documents, orders, as well as possible changes to the information of the above entities.

2. Main functions of the system

2.1. Functions of mobile application

2.1.1. Register

Name of function: Register	Code of function: MB001
Agent: Customer	Priority level: High
Content: Help customers create a new account on the system	
Condition: The device must be connected to the internet.	
Handle: When a customer logs in to the system, users can click the " Account Registration " section to proceed with the new account registration. The system will require entering the information: <ul style="list-style-type: none">- Name of shop- Phone number- Email- Password- Address- Name of bank account- Number of bank account- Name of bank- Address of bank The system will then verify the registration information.	
Result: If the information entered by the customer is valid, the system will create and store the information into the database, after which the customer can log in to the system.	

In case the input information is invalid, the system will display the notification and ask the customer to re-enter.

Note:

Table 1: Description of registration function for customers

2.1.2. Login

Name of function: Log in	Code of function: MB002
Agent: Customer	Priority level: High
Content: Allows customers to log into the system to use the available functions.	
Condition: The device must be connected to the internet. Must be logged in with an account registered in the system	
Handle: When the user accesses the app, then clicks the " Login " section, the system proceeds to ask the customer to enter the information: - Email - Password The system will then confirm the login information.	
Result: If the customer's login information is correct, the system will confirm, and put the user in the main interface once logged in. Conversely, if the customer's login information is invalid, the system will display the notification, and ask the customer to re-enter.	
Note:	

Table 2: Description of login function for customers

2.1.3. Log out

Name of function: Log out	Code of function: MB003
Agent: Customer	Priority level: High
Content: Allow customer to log out of their account.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered in the system.	
Handle: When the customer is logged into the system, the user can click the " Log out " section .	
Result: The system will proceed to log out of the customer's current account, and bring the customer back the login screen.	
Note:	

Table 3:Description of log out function for customers

2.1.4. Account management

Name of function: Account management	Code of function: MB004
Agent: Customer	Priority level: High
Content: Allow customers to manage and correct the information of their account.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered in the system.	
Handle: Once the customer has logged into the system, the customer can click the " User " section to be able to correct the account's information. Information that customers can correct includes: <ul style="list-style-type: none">- Name- Phone number- Email- Password- Name of bank account- Number of bank account- Name of bank- Address of bank- Address to get orders The system will then proceed to process the correction of the information.	
Result: If the information that the customer enters is valid, the system will store and notify the customer. Conversely, if the information that the customer enters is invalid, the system will notify and ask the customer to enter the information again.	
Note:	

Table 4: Description of account management function for customers

2.1.5. Create new orders

Name of function: Create new orders	Code of function: MB005
Agent: Customer	Priority level: High
Content: Allow customers to create new orders on the system	
Condition: The device must be connected to the internet. The system must be logged in by an account registered in the system.	
Handle: When a customer logs into the system, clicking the " Order " section appears in the " Create new order " section so that the customer can create a new order on the system. The information that the system requires to create an order: <ul style="list-style-type: none">- Phone number of recipient- Name of recipient	

<ul style="list-style-type: none">- Address of recipient- Form of transportation- Name of product- Weight of product- Number of product- Cast <p>The system will then process the creation of the order.</p>
<p>Result:</p> <p>After the system confirms that the order information is valid, the system proceeds to create, and store the order into the system, then notify the customer.</p> <p>In case the order information is invalid, the system will ask the customer to enter the order information again.</p>
<p>Note:</p>

Table 5: Description of creating new orders function for customers

2.1.6. Manage order information

Name of function: Manage order information	Code of function: MB006
Agent: Customer	Priority level: High
Content: Allow customers to manage order information	
Condition: The device must be connected to the internet. The system must be logged in by an account registered in the system. Customers must have an order created in the system.	
Handle: When the customer clicks the " Orders " section, the interface will now show the orders being processed on the system.	
Result: When clicking on the system, it will provide information, the status of the order such as: <ul style="list-style-type: none">- The current status of the order- Note- Name of product- Updated status list- List of order requests- Map showing the current location of the order	
Note:	

Table 6: Description of managing order information new orders function for customers

2.1.7. Submit a request for an order

Name of function: Submit a request for an order	Code of function: MB007
--	--------------------------------

Agent: Customer	Priority level: Medium
Content: Allow customers to submit requests for the order.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. Customers must have an order registered on the system.	
Handle: When the customer clicks the " Orders " section, the interface will now show the orders being processed on the system.	
Result: When clicking on the system, it will provide information, the status of the order such as: <ul style="list-style-type: none">- The current status of the order- Note- Name of product- Updated status list- List of order requests- Map showing the current location of the order	
Note:	

Table 7: Description of submitting a request for an order new orders function for customers

2.1.8. Cash flow control

Name of function: Cash flow control	Code of function: MB008
Agent: Customer	Priority level: Medium
Content: Help customers control the cash flow of the existing order.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system.	
Handle: When the customer clicks the " Money Flow " section. The system will provide information about cash flow.	
Result: The system will provide information about cash flow such as: <ul style="list-style-type: none">- Delivery fee- Unchecked money- Money has been paid.	
Note:	

Table 8:Description of cash flow control function for customers

2.2. Functions of web application

2.2.1. Register

Name of function: Register	Code of function: WS001
-----------------------------------	--------------------------------

Agent: Driver	Priority level: High
Content: Help drivers create a driver account on the system	
Condition: The device must be connected to the internet.	
Handle: When a driver logs in to the system, users can click the " Account Registration " section to proceed with the new account registration. The system will require entering the information: <ul style="list-style-type: none">- Name of driver- Phone number- Email- National identity card- Password- Driving license- Age- Username- Name of bank The system will then verify the registration information.	
Result: If the information entered by the driver is valid, the system will create and store the information into the data-sized mechanism, after which the driver can log in to the system. In case the input information is invalid, the system will display the notification and ask the driver to re-enter.	
Note:	

Table 9: Description of register function for drivers

2.2.2. Login

Name of function: Login	Code of function: WS002
Agent: Driver	Priority level: High
Content: Allows driver to log into the system to use the available functions.	
Condition: The device must be connected to the internet. Must be logged in with an account registered in the system	
Handle: When the driver accesses the app, then clicks the " Login " section, the system proceeds to ask the user to enter the information: <ul style="list-style-type: none">- Username- Password The system will then confirm the login information.	
Result: If the driver's login information is correct, the system will confirm, and put the driver in the main interface once logged in.	

Conversely, if the driver's login information is wrong, the system will display the notification, and ask the driver to re-enter.

Note:

Table 10: Description of login function for drivers

2.2.3. Log out

Name of function: Log out	Code of function: WS003
Agent: Driver	Priority level: High
Content: Allow drivers to log out of their account.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system.	
Handle: When the drivers is logged into the system, the user can click the " Sign out " section .	
Result: The system will proceed to log out of the driver's current account, and bring the driver back the login screen.	
Note:	

Table 11: Description of log out function for drivers

2.2.4. Account management

Name of function: Account management	Code of function: WS004
Agent: Driver	Priority level: High
Content: Allow drivers to manage and correct the information of their account.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system.	
Handle: Once the driver has logged into the system, the drivers can click the " Edit driver information " section to be able to correct the account's information. Information that drivers can correct includes: <ul style="list-style-type: none">- Name- Phone number- Email- Password- National identity card- Driving license The system will then proceed to process the correction of the information.	
Result: If the information that the customer drivers is valid, the system will store and notify the drivers.	

Conversely, if the information that the drivers enters is invalid, the system will notify and ask the drivers to enter the information again.

Note:

Table 12: Description of account management function for drivers

2.2.5. Orders available on the system management

Name of function: Orders available on the system management	Code of function: WS005
Agent: Driver	Priority level: High
Content: Help the drivers can see the orders available on the system so that new orders can be received for delivery.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system.	
Handle: When accessing the system, the driver can click the " Available orders " section. The system then proceeds to retrieve the data, and takes the driver to the page that displays the orders available on the system.	
Result: The system will now return to the driver a colloed list of orders available on the system including information: the name of the recipient, the recipient's phone number, the name of the item, the total cast. The driver can then click to see the details of the order, at which point the system will provide more information about the order: <ul style="list-style-type: none">- Name of the recipient- Recipient's phone number- Name of the item- Total cast- Number of items- Form of delivery- Note	
Note:	

Table 13: Description of orders available on the system management function for drivers

2.2.6. Receive order for delivery

Name of function: Receive order for delivery	Code of function: WS006
Agent: Driver	Priority level: High
Content: Allows the driver to receive pre-existing orders on the system, and proceeds to deliver.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system.	

Handle: When accessing the system, driver can click the " Get order " button in the " Current Orders " page. At this time, the system will handle the receipt of orders for the driver.
Result: The system, after conducting a confirmation that the driver is eligible to receive the goods, will save it to the database, and notify the driver. Conversely, if the driver is not eligible to receive the order, the system will still notify the driver.
Note:

Table 14: Description of receiving order for delivery function for drivers

2.2.7. Track current order requests

Name of function: Track current order requests	Code of function: WS007
Agent: Driver	Priority level: Normal
Content: Allows the driver to track the requests the customer sends back for the current order the driver is receiving.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. The driver is receiving a certain order.	
Handle: When accessing the system, at the home page, the driver can see the current requirements of the order through the " Requests of the Order " section.	
Result: If an existing order exists a request from a customer, the system displays a list form that includes: the request time, and the requested content. In the event that the current order does not have a request, the system will display the notification to the driver.	
Note:	

Table 15: Description of tracking current order requests function for drivers

2.2.8. Track current order information

Name of function: Track current order information	Code of function: WS008
Agent: Driver	Priority level: High
Content: Help the driver can track the information of the current order, serve the shipping, update the status, location for the order. Help bring information to customers.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. The driver is receiving a certain order.	
Handle:	

When accessing the system, driver can track the information of the current order in the " Current Order " section.
<p>Result:</p> <p>When the driver logs into the system, the system collects the information of the current order, and displays a summary form that includes the following information:</p> <ul style="list-style-type: none"> - The recipient's name. - Recipient's phone number - Name of item - Total cast - Delivery address - A miniature map showing the delivery location of the order, the driver's current location, and the shortest travel from the current location to the location to be delivered. <p>The system also provides a detailed interface to display the current information of the order, which will now show but the information:</p> <ul style="list-style-type: none"> - The recipient's name. - Name of item - Recipient's phone number <p>The number of items.</p> <ul style="list-style-type: none"> - Weight of item - Delivery address - Note <p>In case the driver has not received any orders, the system will display the notification to the driver.</p>
Note:

Table 16: Description of tracking current order information function for drivers

2.2.9. Update the current location of the order

Name of function: Update the current location of the order	Code of function: WS009
Agent: Driver	Priority level: High
Content: Allow drivers to update the geographic coordinates of current order.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. The driver is receiving a certain order.	
<p>Handle:</p> <p>If the driver clicks the " Update location " button, the system will proceed to update the location of the current order.</p>	
<p>Result:</p> <p>The system will take the driver's current coordinates, then send it to the server to conduct an update, and store it in the database, which will eventually show the notification to the driver.</p>	

Note:

Table 17: Description of updating the current location of the order function for drivers

2.2.10. Update the delivery status of the current order

Name of function: Update the delivery status of the current order	Code of function: WS0010
Agent: Driver	Priority level: Normal
Content: Allows the driver to update the current status of the order.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. The driver is receiving a certain order.	
Handle: If the driver clicks the " Update status " button in the page that displays the details of the current row. The system displays a pop-up that includes the current status of the order, and a list for the purpose of letting the driver choose the status name to update. The system will then proceed to update the delivery status for the order.	
Result: The system will now take the state code, send it to the server for the server to update, and save it to the database, which will eventually show the notification to the driver.	
Note:	

Table 18: Description of updating the delivery status of the current order function for drivers

2.2.11. Confirmation of payment

Name of function: Confirmation of payment	Code of function: WS0011
Agent: Driver	Priority level: High
Content: Allow the driver after delivery to send a signal to the server to update the information.	
Condition: The device must be connected to the internet. The system must be logged in by an account registered at the system. The driver is receiving a certain order.	
Handle: If the driver clicks the " Confirmation of payment " button in the page that displays the details of the current row. The system will conduct the payment confirmation process for the order	
Result: The system will send it to the server for the server to update, and save it to the database, which will eventually show the notification to the driver.	

Note:

Table 19: Description of confirmation of payment function for drivers

CHAPTER III. IMPLEMENT CONCEPTION

In this study, I will focus on building a server that can manage database, receive and send signals and information to applications in the system. The two applications for customers and drivers can easily operate, as well as manage the information of the orders.

1. Database design

Below is a description of the design of the database

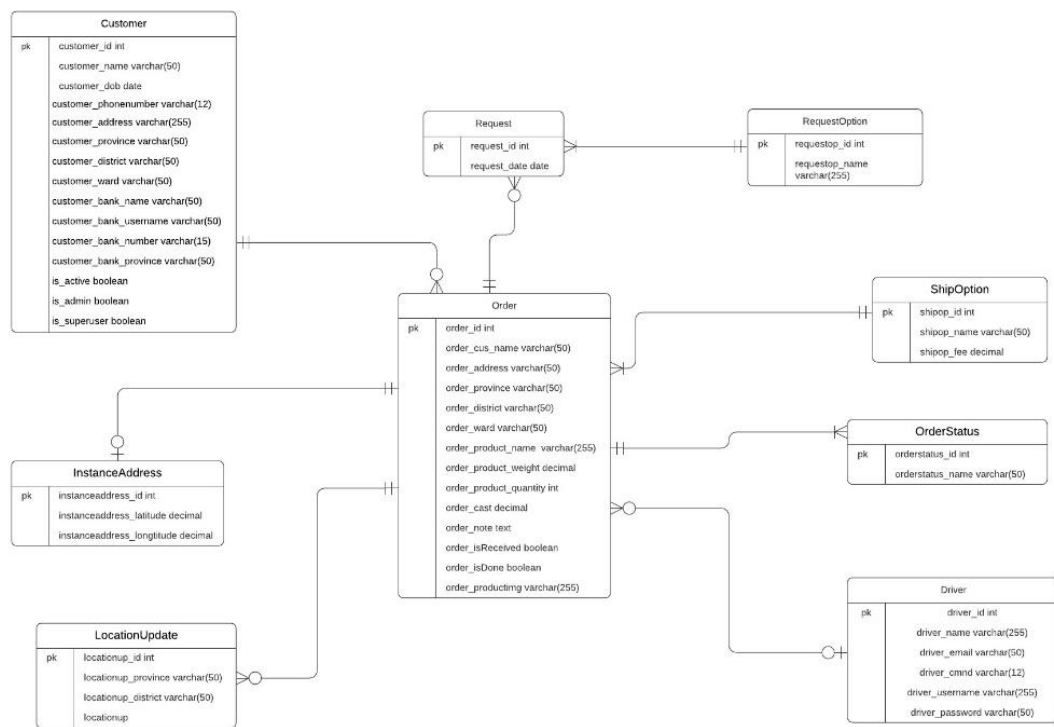


Figure 1: Database design diagram of the system

The RequestOption table is used to store the options that can be requested for the order:

Attribute	Type	Key	Unique	Null	Default	Explain
requeststop_id	int	x	x	False		Code of request option
requeststop_name	varchar			False		Name of request

Table 20: Database description table – RequestOption

The Request Table contains requests requested for an order:

Attribute	Type	Key	Unique	Null	Default	Explain
request_id	int	x	x	False		Code of request

request_time	datetime			False		The time the request was created
--------------	----------	--	--	-------	--	----------------------------------

Table 21: Database description table – Request

The ShipOption table contains options for shipping:

Attribute	Type	Key	Unique	Null	Default	Explain
shipop_id	int	x	x	False		Code of shipping's option
shipop_name	varchar			False		Name of shipping's option
shipop_fee	int			Null		Cost of shipping's option

Table 22: Database description table – ShipOption

OrderStatus table contains the shipping status of an order:

Attribute	Type	Key	Unique	Null	Default	Explain
orderstatus_id	int	x	x	False		Code of order's status
orderstatus_name	varchar			False		Name of order's status

Table 23: Database description table – OrderStatus

InstanceAddress table contains the current geographic coordinates of an order:

Attribute	Type	Key	Unique	Null	Default	Explain
insaddress_id	int	x	x	False		Code of the current location of the order
insaddress_latitude	varchar			False		The current latitude of the order
insaddress_longitude	varchar			False		The current longitude of the order

Table 24: Database description table – InstanceAddress

LocationUpdate table of the most up-to-date locations of an order

Attribute	Type	Key	Unique	Null	Default	Explain
locationupdate_id	int	x	x	False		Code of the updated location

						of the order
locationupdate_province	varchar			False		province of the updated location of the order
locationupdate_district	varchar			False		district of the updated location of the order
locationupdate_ward	varchar			False		ward of the updated location of the order
locationupdate_time	datetime		x	False		time of the updated location of the order

Table 25: Database description table – LocationUpdate
Customer table containing customer account information

Attribute	Type	Key	Uni - que	Null	Default	Explain
customer_id	int	x	x	False		The code of the customer's account
customer_name	varchar		x	False		Name of customer
customer_date_of_birth	date			False		Date of birth of customer
customer_phone_number	varchar		x	False		phone number of customer

customer_address	varchar			False		detail address of customer
customer_province	varchar			False		province address of customer
customer_district	varchar			False		district address of customer
customer_ward	varchar			False		ward address of customer
customer_bank_name	varchar			False		Name of the bank used for payments
customer_bank_username	varchar			False		Username of the bank used for payments
customer_bank_number	varchar		x	False		Bank account number used for payment
customer_bank_province	varchar			False		Province of the account's bank
customer_is_active	boolean			False	True	Account activation status
customer_is_admin	boolean			False	False	Status showing if the account is a manager's account
customer_is_superuser	boolean			False	False	Status showing if the account is

						a manager's account
--	--	--	--	--	--	---------------------------

Table 26: Database description table – Customer

Driver's table saves driver's account information

Attribute	Type	Key	Unique	Null	Default	Explain
driver_id	int	x	x	False		Code of the updated location of the order
driver_name	varchar		x	False		Name of driver
driver_email	varchar		x	False		Email of driver
driver_cmnd	varchar		x	False		Passport of driver
driver_username	varchar		x	False		Username of driver's account
driver_password	varchar			False		Password of driver's account

Table 27: Database description table – Driver

Order table containing the information of an order

Attribute	Type	Key	Unique	Null	Default	Explain
order_id	int	x	x	False		Code of the order
order_customer_phone number	varchar			False		Recipient's phone number
order_customer_name	varchar			False		Recipient's name
order_detail_address	varchar			False		Recipient's detailed address
order_province	varchar			False		Recipient's province address
order_district	varchar			False		Recipient's district address
order_ward	varchar			False		Recipient's ward address
order_product_name	varchar			False		Name of the item to be

						sent of the order
order_product_weight	float			False		Weight of the item to be sent of the order
order_product_quantity	int			False	1	Quantity of the item to be sent of the order
order_cast	int			False		total cast of the order
order_note	text			False		Note of the order
order_isReceived	boolean			False	False	Status showing whether the order has been received by the driver
order_isDone	boolean			False	False	Status showing whether an order was delivered or not
order_productimg	varchar					An image of the order uploaded from customer

Table 28: Database description table – Order

2. Server design with Django framework

In this section I will build a server through the Django framework. This server is responsible for storing databases, processing requests between two applications for drivers and customers.

2.1. Configure the setting file

In order for the server to function the full functionality required by the system, I needs to configure the file setting for the Django framework.

First of all, the security section will now use JWT (JSON Web Token) to perform authentication between the two applications and the server when exchanging data. The first to speak briefly about the JWT concept, JWT is a means of representing

the transfer requirements between the two client-server parties, the information in the JWT chain is formatted by JSON. The token chain must have three header parts, the payload and signature sections blocked with the "." Here I will use the simple-jwt plugin for security purposes, now in the setting file we need to add:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny'
    ],
}
```

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny'
    ],
}
```

Regarding the installation of JWT parameters, we need to pay attention to the following important parameters:

```
SIMPLE_JWT = {

    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=60),

    'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
    ...

    'AUTH_HEADER_TYPES': ('Bearer', ),

    'AUTH_HEADER_NAME': 'HTTP_AUTHORIZATION',

    'USER_ID_FIELD': 'id',

    'USER_ID_CLAIM': 'user_id',

}
```

To help the server interact with mobile apps and website apps for drivers, I use the plugin corsheaders. And here are some things to configure in the setting file.

```
INSTALLED_APPS = [
    ...
    'corsheaders',
    ...
]
```

```
CORS_ORIGIN_WHITELIST = (
    'http://localhost:3000',
)
```

2.2. Configure model file

A model is the single, definitive source of information about data. It contains the essential fields and behaviors of the data system is storing. Generally, each model maps to a single database table. Each attribute of the model represents a database field.

So I'm going to create entities in the database diagram that's been drawn up before. However, creating a customer will be a little different from the rest. In order for the server side to understand that the customer is actually the authenticate user class for the system, the customer class must inherit from an `AbstractBaseUser` layer, along with functions such as: `is_staff()`, `has_perm()`, `has_module_perms()`. Here's the content:

```
@property
def is_staff(self):
    "Is the user a member of staff?"
    return self.is_admin

def has_perm(self, perm, obj = None):
    return True

def has_module_perms(self, app_label):
    return True
```

At the same time, we also have to re-configure the setting file:

```
AUTH_USER_MODEL = 'api.Customer'
```

2.3. Create processing file for customer application

The `views.py` file will be the place where all functions and layers are used for the processing of requests sent from the customer's mobile application side.

In terms of functions and classes for authentication, the system will have 4 classes: ***Middleware*** - The class will include the get function for the purpose of authenticate the token sent from the application side; ***RegisterView*** - The class consists of a post function that has the ability to receive but the information is sent from the application side, then authenticates the information, if the information is valid, the function will proceed to create a password is an encryption chain from the password received from the application side, then save to the database; ***UpdatePasswordView*** - the class will include a post function with the function of receiving information that requires updating the password from the application side, then proceed to update the password, then save to the database. One class consists of a function that requires multiple processing stages, the ***SignInView*** class , which includes a post function with the following content:

```
Function post
    Set attribute serializer according to data of request
    If serializer is valid:
        Call authenticate function
        Set attribute user according result of authenticate
function
    If user is valid:
        Set attribute refresh_token, access_token according to class
TokenObtainPairSerializer
        Return Response with refresh_token, access_token
    Return error Response
```

On the side of functions, the classes serves the functions of the mobile application including:

- ***UserInformationView*** class: the class will include a get function that will receive a GET request from the app side, which will then return the customer's information that the application requests; a post function with the function of receiving requests for password updates from the application side, creating a new password by encrypting the password sent from the application side, and then saving to the database.

- ***BankingInformation*** class - the class consists of a post function with the function of receiving requests to update the bank information of the account, after confirming that the data is valid will proceed to the update, and save to the database.

- ***LocationInformation*** class - the class will include a post function with the function of receiving requests to update the address of the account's current receipt, after confirming that the information is correct, the system will proceed to update, and save to the database.

- ***OrderView*** class - the class that will include get will receive requests for a list of orders of the current account, which will return the list information of those orders. A post function with the following:

```
Function post
Set attribute user according to user of request
Set attribute shipOption according to data of request
Set attribute statusInstance according to data of request
Create a new order with data from request and above attributes
Return completed Response
```

- ***OrderDetailView*** class - class consists of a get function with the function of receiving requests to retrieve the information of the order, then processing and returning the details of that order as JSON.

- ***RequestView*** class - the class will consist of a request receiving function that creates a request for a specified order, after confirming that the request's information, and the order is valid, the system will proceed, creating a new order request and saving it to the database.

- ***ListRequestView*** class - the class will include a get function that receives requests to return the reserves of the available requests of a specified order.

- ***PaidMoneyView*** class - includes a get function with the following content:

```
Function get
Set attribute userInstance according to data of request
Set attribute orders according userInstance
Set attribute moneyStatus according to data of request
Set attribute totalMoney equal to 0.0
If moneyStatus equal to "1":
  For order in orders:
    If status id of order equal to "1":
      Update totalMoney equal cast of order + totalMoney
If moneyStatus equal to "2":
  For order in orders:
    If status id of order equal to "5":
      Update totalMoney equal cast of order + totalMoney
```

```
If moneyStatus equal to "3":  
    For order in orders:  
        Update totalMoney equal fee of order + totalMoney  
    Return Reponse with totalMoney
```

- ***SpecificOrderView*** class - the class will include a get function that receives requests to send back to a list of orders with a certain state, after processing will return the book of orders in JSON form.

- ***StatusUpdate*** class - class consists of a get function with the function of receiving requests to retrieve the list of updated states of a given order; post function with the function of creating a status update with status, and order specified.

- ***StatusOrderView*** class - class consists of a get function with the function of receiving requests from the application side and returning the status of the specified order; a post function with the function of updating or creating a new status update object for the specified order.

2.4. Create processing file for driver website application

The driver_views.py file will be the place where all functions and classes are used for the processing of requests sent from the driver's mobile application side.

In terms of functions and classes for authentication, the system will have 2 classes: ***Middleware*** - The class will include the get function for the purpose of authenticate the token sent from the application side; ***SignInView*** - the functional layer consists of a post function with the function of receiving login requests from the application side, after confirming that the login information is valid, the system will create a refresh token and access token and then send it to the application side to serve the login authentication later.

On the side of functions, the classes serves the functions of the website application including:

DriverView class - class consists of a get function with the function of receiving requests, and returns the driver's information through the token included in request.headers; The post function has the function of receiving a new account registration request for the driver, after verifying the information, will proceed to create the password by encrypting the password sent from the application side, then save to the database.

UpdateDriverInformationView class - class will include a post function with the function of receiving requests to update information for the driver's account, which will then proceed to update the information, and save to the database.

LocationUpdateView class - class consists of a get function with the function of receiving requests and returning the list of updated locations of a nearest order based on the driver's ID; a post function with the function of updating or creating the current address of the specified order.

OrderDriver class - class consists of a get function with the function of receiving requests from the application side, and returns the list of requests of the order that is in processing based on the driver's ID.

InstanceOrdereView class - class includes a get function with the function of receiving requests, and returns the list of orders that are available on the system.

SetDriverOrderView class - class consists of a post function with the function of receiving a driver update request for the specified order, If the system determines the order is being received by another driver, the system will send a notification on the application side, in case the order has not yet been received by the driver, the system will update the designated driver for that order.

InstanceOrderView class - class consists of a get function that receives a request, and returns the current order of the receiving driver. 3. Establish a connection between server and application.

UpdatePaidOrder class - class consists of a post function with the function of receiving requests to update the status of the specified order as delivered, and then save to the database.

2.5. Create file to define urls

The url.py file will include an array urlpattern where the definition has a path contained in the api section of the system. The following is the table that defines the URL, and the function for that url.

Code	Name	Function
URL001	log-in	views.SignInView
URL002	middleware	views.MiddleWare
URL003	register	views.RegisterView
URL004	user-information	views.UserInformationView
URL005	order-detail	views.OrderDetailView
URL006	order-information	views.OrderView
URL007	bank-information	views.BankingInformation
URL008	location- information	views.LocationInformation
URL009	request	views.RequestView
URL010	specific-order	views.SpecificOrderView
URL011	status-update	views.StatusUpdate
URL012	list-request	views.ListRequestView
URL013	status-order	views.StatusOrderView
URL014	paidmoney	views.PaidMoneyView
URL015	total-order	views.TotalOrderView
URL016	driver-view	driver_view.DriverView
URL017	driver-signin	driver_view.SignInView
URL018	driver-middleware	driver_view.MiddleWare
URL019	update-location	driver_view.LocationUpdateView
URL020	order-drivers	driver_view.OrderDriver
URL021	available-orders	driver_view.InstanceOrderView
URL022	update-driver	driver_view.UpdateDriveInformationView
URL023	receive-order	driver_view.SetDriverOrderView
URL024	instance-order	driver_view.InstanceOrderView
URL025	set_paid_order	driver_view.UpdatePaidOrder

Table 29: Description of list of server's urls

4. Mobile application for customer

4.1. Log-in page

The log-in page is where the customer can log into the system provided that the customer has an account registered at the system. To log in, the system requires the customer to enter two information: email and password, after which the customer clicks the "Login" button, the system will now through the axios library create a post request sent towards the server. After the server receives the request, the information will be verified, which will send the application a response, based on the state of the Response, the application will display the notification to the customer.

In case the customer wants to create an account, the customer can click the "Account Registration" section, at which point the system will take the customer to the login page.

Figure 2: Log-in page of mobile app

4.2. Register page

To register a new account, the system requires customers to enter a variety of information such as name, phone number, email, password, address, bank account name, bank account number, bank name, bank address.

In order to get the list of banks in Vietnam, I used an API of api.vietqr.io. Here's the code to get a list of banks in Vietnam:

```
axios.get(`https://api.vietqr.io/v1/banks/`)
  .then((response) => {
    this.setState({
      banks: response.data.data
    });
  })
  .catch((error) => {
    console.log(error);
  });
```


To get the list of Provinces available in Vietnam, I used an API of provinces.open-api.vn. Here's the code to get a list of provinces in Vietnam:

```
axios.get('https://provinces.open-api.vn/api/')
  .then((response) => {
    this.setState({
      provinces: response.data
    });
  })
  .catch((error) => {
    console.log(error);
  });
```

After filling in the full information that the system requires, the customer can click the "Register" button, the system will create a post request sent to the server, after the server handles returning the application a response, the system will rely on the status of the Response to display the notification to the customer.

In case the customer wants to log in to the system, the customer can click the " Log in now " section, which will go to the login page.



Figure 3: Register page of mobile app

4.3. Overview page

Overview page is the place to display the general information of the system such as: Terms of regulation, frequently asked questions, news.

4.4. Personal information page

The personal information page is the screen to display the information of the customer account. The page will consist of three main sections:

Customer basic information: name, phone number, email, password. When the customer clicks the "Edit" button, the system redirects to the editing page of the above information. After filling in the information, the system confirms that it is valid, the customer can click the "Save" button to proceed with the information flow, at this time the system will create a post request sent to the server, after the server handles

and sends the application a response, the system will rely on the status of the response to send notifications to customers.

The banking information display is where banking information is displayed such as bank account name, bank account number, bank name, bank address. Customers can click the "Edit" button to edit the above information. After filling in the information, the system confirms that it is valid, the customer can click the "Save" button to proceed with the information flow, at this time the system will create a post request sent to the server, after the server handles and sends the application a response, the system will rely on the status of the response to send notifications to customers.

The delivery address editing section displays the necessary information when taking orders including: name, phone number, address. This section also provides a "Edit" button so that the driver can edit the above information. After clicking the "Edit" button, the system will take the customer to the page to edit the information, the driver can edit: name, phone number, address. As for address editing, customers can choose ward, district based on province, to do this I used the api of vapi.vnappmob.com and provinces.open-api.vn.

Tài khoản

The screenshot displays the 'Tài khoản' (Account) page of a mobile application. At the top, there is a user profile section with a person icon, the username 'IHT2', and the ID 'S321312'. Below this, there are two main sections for editing user information, each with a 'Sửa' (Edit) button in orange.

Thông tin cơ bản (Basic information):

- Home icon: IHT2
- Phone icon: 918026392
- Email icon: admin@gmail.com
- Card icon: (empty field)

Thông tin ngân hàng, đối soát (Banking information, reconciliation):

- Home icon: HUYNH QUAN NHAT HAO
- Card icon: 701213123123
- Bank icon: Ngân hàng TMCP Á Châu
- Location icon: Thành phố Hà Nội

At the bottom of the screen, there is a navigation bar with four icons: Overview, MoneyFlow, Orders, and User (highlighted in red).

Figure 4: Personal information page of mobile application

4.5 Money-flow page

Money-flow page is a page that displays the current account's cast status information such as delivery fees, unpaid total cast, total cast paid.

It also includes the "Settings" button, when a customer clicks, can edit bank account-related information like the personal information page.

4.6. Order information page

Order information page is a page that displays a list of orders in case the user has created an order on the system.



Figure 5: Order information of mobile application

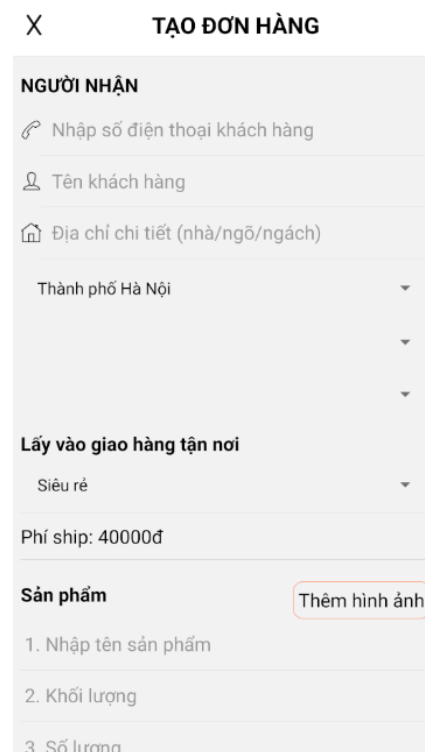
Once the customer has registered an order at the system, the system displays the order information in the form of a list. The basic information that the system provides to customers includes: Order number, recipient's name, recipient's phone number, total cast, current status of order.

In case the customer does not have any orders at the system, the system will notify the customer that the list is empty.

For each order displayed in case the user already has an order at the system, the system will display a "Send a request" button with the function of sending a request in exchange with the selected order, at which point the system will display a list of requests that can be sent to the system such as: urge delivery orders, urges to take orders, etc. At this point, customers can click the " Send a request " button again to proceed with sending the request towards the server. After the request is successfully sent, the system will send a notification to the customer, on the contrary, if the request is sent to the server unsuccessfully, the system will send an alert to the customer.

4.7. New order page

A new order page is one that allows customers to create an order to start shipping.



X TẠO ĐƠN HÀNG

NGƯỜI NHẬN

📞 Nhập số điện thoại khách hàng

👤 Tên khách hàng

🏠 Địa chỉ chi tiết (nhà/ngõ/ngách)

Thành phố Hà Nội

Lấy vào giao hàng tận nơi

Siêu rẻ

Phí ship: 40000đ

Sản phẩm Thêm hình ảnh

1. Nhập tên sản phẩm

2. Khối lượng

3. Số lượng

Figure 6: New order page

When a customer clicks the "Create new order" section of the order information page, the system displays a pop-up that allows the customer to proceed with creating a new order.

The system will require the customer to enter some information necessary for transportation such as: recipient's name, recipient's phone number, recipient's province, recipient's district, recipient's ward, recipient's detail address.

Regarding the information about the delivery method, the system will ask customers to enter: delivery method - each delivery method will have a different price added directly to the total cast (Big - 30000, Small - 20000, ExtraFast - 10000), the name of the product, the image of the product (if any), the weight of the product, the quantity of the product, the money to be collected. The change of delivery method, quantity, money to collect will directly change the total cast displayed at the end of the page.

After the customer has entered all the necessary information, the customer can click the "Create order" button to proceed to send information to the server. After the server receives the request, and the information to create the order, the server will confirm and send a response to the mobile application, and the system will display notifications to the customer about the status of the order creation.

4.8. Order detail page

An order details page is a place that displays information including basic information, and updated information by the driver and manager.



Figure 7: Order details page

About the basic information of the order will be displayed by the system including: the current status of the order - the order during shipping will be updated multiple states at various times, and this is the status at the time the customer views the details of the order, product notes are entered by customers when creating orders, product name.

As for the information given by the driver, the manager will be displayed including: a list of updated statuses at each specific time - when the transport driver will determine the status at that time of the order and update to the system. This is intended to make it easier for customers to track the status of their orders; The list of customer requests - the system will also display a list of requests submitted by the customer to the system side along with the specific time.

4.9. Map page

A map page is a page that shows the location of an order through a map with the aim of helping users track their order location in more detail.



Figure 8: Map page

First talking about how the map works, I used the "react-native-map" library provided by Google to display information about the geographic coordinates of the order. When a customer creates a new order, the customer will have to provide a delivery address for the system to save to the database, and when the customer accesses the Map page, the system will go through an API of positionstack.com to be able to obtain the geographical coordinates of the order through the address, at the same time, when the driver receives the customer's order, the geographical coordinates of the driver will also be regularly updated to the system, so the customer's mobile app can get the geographical coordinates of the driver. Once the system has obtained the geographic coordinates of the delivery location, and the driver, the system displays on the Map component.

Once the order has been updated to the "Delivered" status, the system will now notify the customer that the order has been delivered, and no longer display a detailed map of the order's location.

5. Web application for driver

5.1. Log-in page

The login page is where the driver can proceed to log in to the system. At this point, the system will ask the driver to enter two information: username and password, after the driver fills in the information, and click the "Log in" button, the system will proceed to send a request to the server side through the axios library with the following content:

```
axios.post(`${ipAddress}/api/log-in/`, {  
  email: this.state.email,  
  password: this.state.password  
})
```

After sending the request to the server side, if the server confirms that the information is correct, the server will send it to the application refresh token and

access token, at which time the application will save access token to local storage as follows:

```
await localStorage.set('token', response.data.access_token);
```

In case the driver does not have an account, the driver can click the " Register now " section to redirect to the new account registration page.

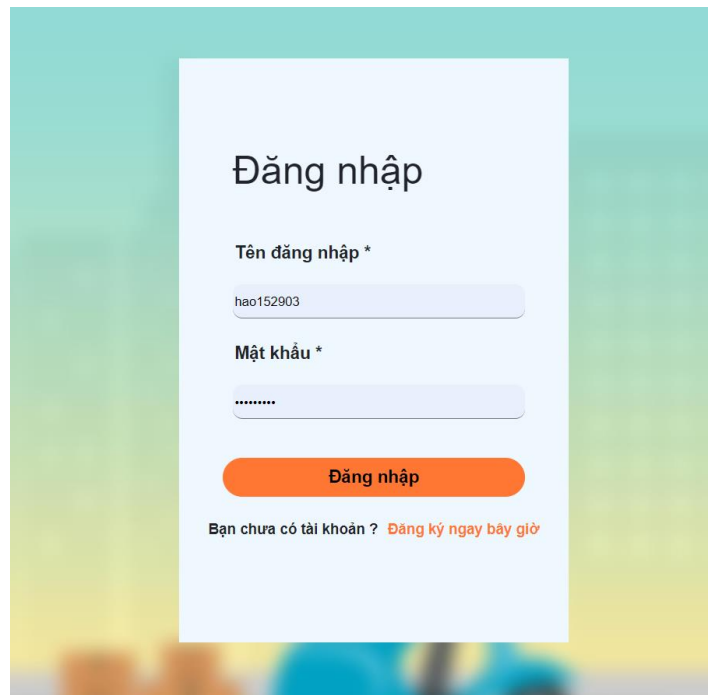


Figure 9: Log-in page of website

5.2. Register page

The registration page is where the driver can proceed to register a new account at the website. To be able to register a new account, the driver needs to enter information such as: full name, phone number, email, household, age, driver's license, username, and password. Once the information is filled in, the driver clicks the " Register " button, at which point the system will retrieve the information entered and send it to the server. If the response sent back from the server is valid, the system will notify the driver, and take the driver to the login page, in case the response received is invalid, the system will display the notification and ask the driver to re-enter the information.

In case the driver already has an account, and wants to log in to the system, the driver can click the " Log-in now " section, the system will redirect to the log-in page.

5.2. Home page

The home page is where the system's summary information includes:

Basic information of the driver: is the section that displays a summary of the driver's information such as full name, phone number, driver's license, username. The purpose of this section is to make it possible for the driver to look through important information when there is a need for work on the road.

The updated location list of the last drive: is the section that displays the list of updates on the status of the driver's last ride, in case the driver is just starting work, the system will show as a list.

The list of requests of the current order: is the display of the list of requests from the customer for the current order, in case the order does not exist any requests on the customer's side, the system will display the empty list.

Displays the basic information of the current drive: is to display a summary of the information of the order that the driver is receiving. As for the information entered by the customer including: recipient name, recipient phone number, name of item, total cast, delivery address. This section also includes a miniature map that integrates directions from the driver's current location for the location in need of delivery. Every five seconds the system updates the driver's current location towards the server so that the user can track the current location of his order. In case the driver has not received any orders, the system will show that no orders exist.

The list of orders available on the system: the section that displays the list of orders that are in waiting state for delivery, each item will also include summary information such as customer name, shipping method, total cast. Each item on the list also includes a button so that the driver can see the details of the order, at this time the information is displayed including: recipient name, recipient phone number, address, name of item, quantity, total cast, delivery form, note. When the driver looks through the order request information, the " Receive order " section can be clicked to proceed with the receipt of the order for delivery. In the absence of any orders waiting for delivery, the system will show that it is not valid for any order.

The home page is also a place that includes navigation bar that can be redirected to pages such as: current orders, existing orders on the system, personal information of the driver.

5.3. Available orders on the system page

Existing orders on the system are individual displays for orders that are ready for delivery on the system.

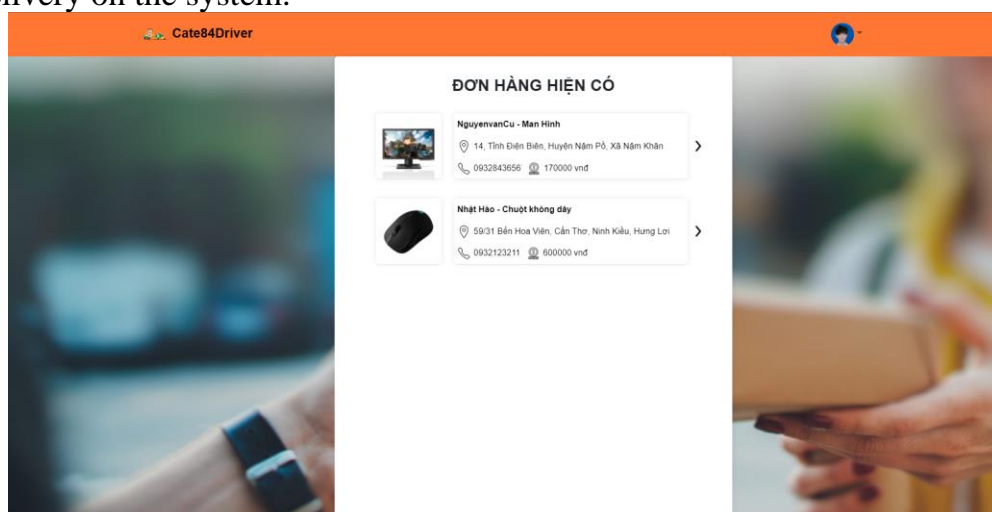


Figure 10: Available orders on the system page

When the driver opens existing orders on the system page, the system displays a list of orders according to summary information such as the recipient's name, the recipient's phone number, and the item name, and the total cast. At the same time as the home page, the system also allows the driver to view the details of the order, then receive to start delivery. After the driver confirms, the system will through the axios library create a post request and send it to the server, if the server confirms that the information sent by the website is valid, the server will send a response, then the system will rely on the response information that displays the notification to the driver.

5.4. Personal information of the driver page

Personal information of the driver page is a place where drivers can track account details, as well as change them. But the information that the driver can interact with includes: Full name, email, phone number, driver's license, ID card, password.

When the driver wants to edit the information, the system requires the driver to enter the personal information in accordance with the requirements of the system, then click the " Save information " button, at which point the system will conduct an information search, then through the axios library create a post request and send it to the server, after the server receives and handles will send to a response then the system will depend on the status of the response that displays the notification to the driver.

5.5. Current order page

This can be considered the most important part of the system, which is a page that helps drivers to observe and update the situation of the order to the system.

On the information side of the order, the details of the order are displayed including: recipient name, recipient phone number, total cast, number of items, weight, delivery address, province, district, ward, note.

The page also includes a large map where the driver can track the current location, and the nearest path is indicated by the system to the location of delivery.

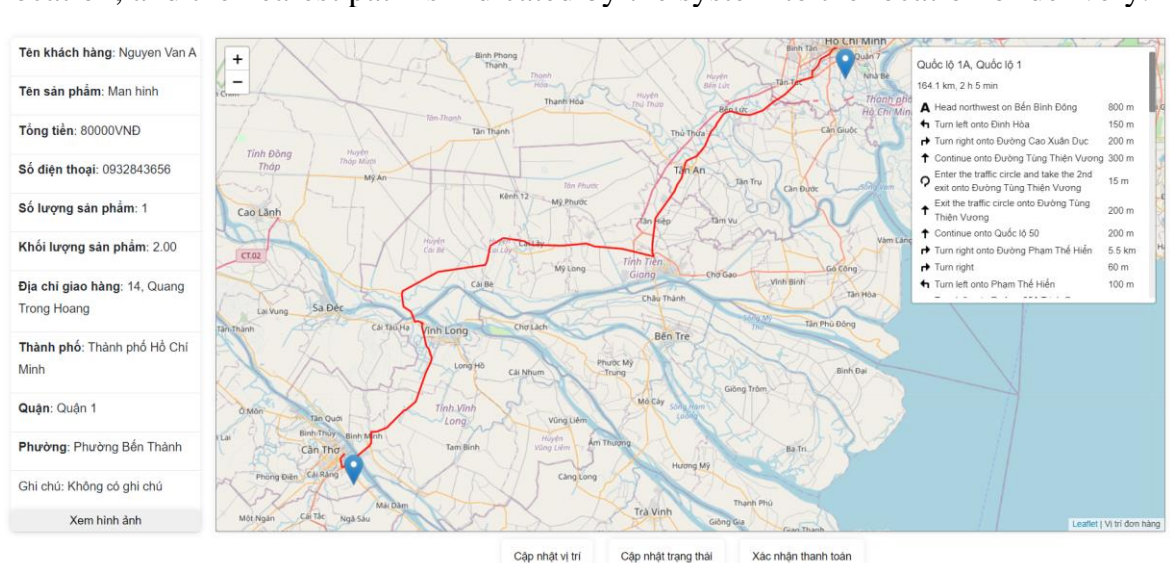


Figure 11: Direction map

To find the shortest route between the driver's current location and the location where the delivery is needed, the system will first receive the information of the current order, then ask axios library to create a post request to send to an api created for the purpose of relying on the address changed to coordinates.

```
axios.get(`http://api.positionstack.com/v1/forward?access_key=ee95aa7c3e382e9aa806014b08955f13&query=1600${response.data.province}`)
  .then((response) => {
    this.setState({
      deliveredAddress: response.data.data[0]
    });
  })
```

Once the page is loaded, the system will start taking the driver's current coordinates, and every two seconds thereafter, the system updates the driver's current coordinates, and sends it toward the server.

```
navigator.geolocation.getCurrentPosition((position) => {
  let context = {
    latitude: position.coords.latitude,
    longitude: position.coords.longitude
  }
  this.setState({
    instanceAddress: context
  });
});
```

And to find the shortest path when there are already coordinates of two points, we will use the leaflet-routing-machine library, the content to create a component that receives two coordinates of two points and draws out the shortest path:

```
const instance = L.Routing.control({
  waypoints: [
    L.latLng(props.delivered[0], props.delivered[1]),
    L.latLng(props.current[0], props.current[1])
  ],
  lineOptions: {
    styles: [{ color: "#6FA1EC", weight: 2}]
  },
  show: false,
  addWaypoints: false,
  routeWhileDragging: true,
  draggableWaypoints: true,
  fitSelectedRoutes: true,
  showAlternatives: false
});
return instance;
};
```

In order for the driver to proactively update the information for the order, the system also provides three functions:

Location update - the driver's gas presses the " Update location " button, the system will proceed to take the current coordinates of the order, and create a post request sent to the server to conduct a location update for the current order, then rely

on the content of the Response sent from the server side that displays the notification to the driver.

Confirm the delivery, and payment for the order - When the driver clicks the " Confirm payment " button, the system will proceed to create a post request and send it to the server for the purpose of updating the status assigned to the current order to " delivered ". After the server confirms, and sends back to the website a Response, the system redirects the driver to the home page to proceed with receiving new orders.

Status update - when the driver clicks the " Update status " button, the system will appear a pop-up showing the status status of the order, and accompanied by a list of statuses that the driver can update to the order, after the driver clicks the " Confirm " button, the system will create a post request sent to the server to proceed with the status update for the current order. When the server is finished processing, and sends back to the website a response, the system will be based on the status of the Response that displays the notification to the driver.

CHAPTER IV. TESTING AND REVIEWING

To check for possible errors in app deployment and ensure the system performing the functions as required, I apply the testing method Black box to conduct system testing

1. Testing objectives

Make sure the final product meets the user's requirements.

The document was created to help you test the app to estimate the calculations testing capacity, scope of testing, implementation schedule. For service to the insurance process sustained and developed in the future.

2. Testing contents

The testing contents include details of testing plans, testing process management and some cases as well as test scenarios built to evaluate the entire system.

2.1. Details of the testing plan

In this testing plan, I would like to give priority to presenting the test results of some of the key functions required in the system. Functions similar to or similar to those presented will not be specified. In addition, I will also clearly state the testing criteria and products handed over of the test.

2.1.1. Functions tested

Below is a summary of functions of customer's mobile application that need to be tested with the expected results for each test case.

Function	Overview	Cases	Results
Log in	Customers log in to the system	- Success - Wrong email - Wrong password	- Successful login - Corresponding error message when an error occurs
Register	Customers register accounts on the system	- Success - Wrong email format	- Successful register - Corresponding error message when an error occurs
Edit account information	Customers edit their account information on the system	- Success - Wrong information format	- Successful edit information - Corresponding error message when an error occurs
Create a new order	Customers create new orders on the system	- Success - Wrong information format - Lack of information	- Successful create new order - Corresponding error message when an error occurs

Submit a request for an order	Customers submit a request for a specific order created on the system	<ul style="list-style-type: none"> - Success - An error occurred during the submission of the request 	<ul style="list-style-type: none"> - Successful submit a request - Corresponding error message when an error occurs
View order list	Customers ask to see a list of orders created on the system	<ul style="list-style-type: none"> - When the system has an order - When the system does not have a customer order 	<ul style="list-style-type: none"> - See the list of orders created on the system - Corresponding error message when an error occurs
View the information of a particular order	Customers ask to see the information of a particular order	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the information of a particular order 	<ul style="list-style-type: none"> - Successful view the information of a particular order - Corresponding error message when an error occurs
Track the location of an order on a map	Customers ask to track the location of the order on the map	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the information of a particular order 	<ul style="list-style-type: none"> - Successful track the location of an order on a map - Corresponding error message when an error occurs

Table 30: Tested functions of mobile application table

Below is a summary of the functions that need to be tested by the driver's web application with the expected results for each test case.

Function	Overview	Cases	Results
Log in	Drivers log in to the system	<ul style="list-style-type: none"> - Success - Wrong email - Wrong password 	<ul style="list-style-type: none"> - Successful login - Corresponding error message when an error occurs
Register	Drivers register accounts on the system	<ul style="list-style-type: none"> - Success - Wrong email format 	<ul style="list-style-type: none"> - Successful register - Corresponding error message when an error occurs
Edit account information	Drivers edit their account information on the system	<ul style="list-style-type: none"> - Success - Wrong information format 	<ul style="list-style-type: none"> - Successful edit information - Corresponding error message when an error occurs
View overview page	Drivers after logging in to see the overview of	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the 	<ul style="list-style-type: none"> - Drivers can view information related to accounts divided into different sections.

	the account's information	information of overview information of the account	- Corresponding error message when an error occurs
View current order details	Drivers click the "View order details" button to go to the order details page	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the information of overview information of the account 	<ul style="list-style-type: none"> - Successful driver forwards to the details page of the current order - Corresponding error message when an error occurs
View order details	Drivers ask to keep track of the details of the current order	<ul style="list-style-type: none"> - Existing orders exist on the system - There are no current orders on the system 	<ul style="list-style-type: none"> - Successful drivers can see the details of the current order. - Corresponding error message when an error occurs
View current order location on a map	Drivers ask to view current order location on a map	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the information of overview information of the account 	<ul style="list-style-type: none"> - The driver can see the shortest route from the current location to the location of the order - Corresponding error message when an error occurs
View available orders on the system	Drivers ask to view available orders on the system	<ul style="list-style-type: none"> - There are available orders on the system - There are no available orders on the system 	<ul style="list-style-type: none"> - Drivers can see a list of available orders on the system - Corresponding error message when an error occurs
View the details of an order available	Drivers ask to view the details of an order available	<ul style="list-style-type: none"> - Success - There is an error in the process of obtaining the information of an order available 	<ul style="list-style-type: none"> - Drivers can see the details of an available order on the system - Corresponding error message when an error occurs
Receive order	Drivers request to receive orders to proceed with the shipping process.	<ul style="list-style-type: none"> - Success - There is an error in the process of receiving order 	<ul style="list-style-type: none"> - Successful receive order - Corresponding error message when an error occurs

Update geographic coordinates	Drivers update current coordinates to server side	- Success - There is an error in the process of updating geographic coordinates	- Successful updating geographic - Corresponding error message when an error occurs
Update the status of an order	Drivers update the status of an order to server side	- Success - There is an error in the process of updating the status of an order to server side	- Successful updating the status of an order to server side - Corresponding error message when an error occurs
Confirmation of payment	Drivers send payment confirmations to the server	- Success - There is an error in the process of sending payment confirmations to the server	- Successful sending payment confirmations to the server - Corresponding error message when an error occurs

Table 31: Tested functions of web application table

2.1.3. Testing criteria

Successful test criteria: test results that meet the requirements in the specification document, test results must return the desired result and have a state of success or return errors with a clear and understandable message.

Test failure criteria: actual test results differ from the expected results that have been recorded in the specification document or errors are not mentioned.

2.2. Testing environment

Hardware:

- Personal computers have an Internet connection and connect to the network of the system.

- Intel(R) Core(TM) i7-7700H CPU @ 2.80GHz 3.30 GHz; GeForce GTX 1060 6GB GDDR5 GPU; RAM 24.00 GB; SSHD 1TB; SSD 250MB.

Operating system and software:

- Windows 10 Home Single Language operating system, 20H2 version

- Google Chrome Browser 91.0.4472.106 (Official version) (64 bits)

2.3. Mobile application test cases

2.3.1. Log in

Test case code: MB001

Condition: The tester must have an account granted access to the system.

Account for testing - admin:admin@gmail.com; password: hao152903

Case code	Case	Input data	Expected results	Actual results	Evaluate
-----------	------	------------	------------------	----------------	----------

MB001 - 01	Empty input information	None	The system will notify that the information is invalid.	As expected	Successful
MB001 - 02	Email doesn't exist in the system.	Email is not registered in the system.	The system will notify the email is not valid	As expected	Successful
MB001 - 03	Invalid password	Email is valid, password is invalid	The system will notify the password is not valid	As expected	Successful
MB001 - 04	Successful log-in	Email and password are valid	Successfully log in to the home screen	As expected	Successful

Table 32: Test log in mobile application function

2.3.2. Register

Test case code: MB002

Condition: none

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB002- 01	The input information is empty	None	The system will notify that the information is invalid	As expected	Successful
MB002- 02	Successful registration	The input information is valid	Account created and logged in	As expected	Successful

Table 33: Test register mobile application function

2.3.3. Edit account information

Test case code: MB003

Condition: The tester must log into the system with a registered account

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB003- 01	The input information is invalid	Invalid input information	The system will notify that the information is invalid	As expected	Successful
MB003- 02	Successful edit information	The input information is valid	Account information has been updated	As expected	Successful

Table 34: Test edit account information mobile application function

2.3.4. Create a new order

Test case code: MB004

Condition: The tester must log into the system with a registered account

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB004- 01	Information to create a new order is Invalid	Invalid input information	The system will notify that the information is invalid	As expected	Successful
MB004- 02	Successful edit information	The input information is valid	Account information has been updated	As expected	Successful

Table 35: Test create a new order mobile application function

2.3.5. Submit a request for an order

Test case code: MB005

Condition: The tester must log into the system with a registered account, and there are orders that have been created on the system.

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB005- 01	An error occurred during the submission of a request for a specified order	None	The system will notify that an error occurred during the submission of the request	As expected	Successful
MB005- 02	Send a request for a specified order successfully	None	The system displays a successful request send notification	As expected	Successful

Table 36: Test submit a request for an order mobile application function

2.3.6. View order list

Test case code: MB006

Condition: The tester must log into the system with a registered account.

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB006- 01	Customers have not	None	The system will show that there	As expected	Successful

	registered any orders at the system		are no orders on the system that do not yet exist.		
MB006- 02	An error occurred during the retrieval of data from the server.	None	The system will display an error message during data retrieval	As expected	Successful
MB006- 03	The customer has registered the order at the system, and No errors occurred during the process of retrieving data from the server	None	The system displays a list of customer orders that have been registered on the system.	As expected	Successful

Table 37: Test view order list mobile application function

2.3.7. View the information of a particular order

Test case code: MB007

Condition: The tester must log into the system with a registered account, and there are orders that have been created on the system.

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB007- 01	An error occurred during the retrieval of data from the server.	None	The system will display an error message during data retrieval	As expected	Successful
MB007- 02	Successfully get an order	None	The system will display the information of the order	As expected	Successful

	order from the server				
--	-----------------------	--	--	--	--

Table 38: Test view the information of a particular order mobile application function

2.3.8. Track the location of an order on a map

Test case code: MB008

Condition: The tester must log into the system with a registered account, and there are orders that have been created on the system.

Case code	Case	Input data	Expected results	Actual results	Evaluate
MB008- 01	An error occurred during the process of obtaining information from the server.	None	The system will display an error message during data retrieval	As expected	Successful
MB008- 02	The unfilled load of two coordinates of the two coordinates requires directions.	None	The loading image displays to let the driver know that the system is processing	As expected	Successful
WB006 - 03	Successful geo coordinate load	None	The system displays two geographic coordinates of the order, and of the driver	As expected	Successful

Table 39: Test track the location of an order on a map mobile application function

2.4. Web application test cases

2.4.1. Log in

Test case code: WB001

Condition: The tester must have an account granted access to the system.

Account for testing:username: hao152903; password: hao152903

Case code	Case	Input data	Expected results	Actual results	Evaluate
-----------	------	------------	------------------	----------------	----------

WB001- 01	Empty input information	None	The system will notify that the information is invalid.	As expected	Successful
WB001- 02	Username doesn't exist in the system.	Usernames is not registered in the system.	The system will notify the username is not valid	As expected	Successful
WB001- 03	Invalid password	Username is valid, password is invalid	The system will notify the password is not valid	As expected	Successful
WB001- 03	Successful log-in	Username and password are valid	Successfully log in to the home screen	As expected	Successful

Table 40: Test log-in function

2.4.2. Register

Test case code: WB002

Condition: none

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB002 - 01	The input information is empty	None	The system will notify that the information is invalid	As expected	Successful
WB002 - 02	Successful registration	The input information is valid	Account created and logged in	As expected	Successful

Table 41: Test register function

2.4.3. Edit account information

Test case code: WB003

Condition: The tester must log into the system with a registered account

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB003 - 01	The input information is invalid	Invalid input information	The system will notify that the information is invalid	As expected	Successful
WB003 - 02	Successful edit	The input information is valid	Account information has been updated	As expected	Successful

	informatio n				
--	-----------------	--	--	--	--

Table 42: Test edit account information function

2.4.4. View overview page

Test case code: WB004

Condition: The tester must log into the system with a registered account

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB004 - 01	An error occurred during the process of obtaining information from the server.	None	The system will display an error message during data retrieval	As expected	Successful
WB004 - 02	Get information from the server successfully.	None	Information is successfully retrieved and displayed to the screen	As expected	Successful

Table 43: View overview page function

2.4.5. View current order details

Test case code: WB005

Condition: The tester must log into the system with a registered account, and the account received an order.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB005 - 01	An error occurred during the process of obtaining information from the server.	None	The system will display an error message during data retrieval	As expected	Successful
WB005 - 02	Get information from the server successfully.	None	Information is successfully retrieved and displayed to the screen	As expected	Successful

Table 44: Test view current order details

2.4.6. View current order location on a map

Test case code: WB006

Condition: The tester must log into the system with a registered account, and the account received an order.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB006 - 01	An error occurred during the process of obtaining information from the server.	None	The system will display an error message during data retrieval	As expected	Successful
WB006 - 02	The unfilled load of two coordinates of the two coordinates requires directions.	None	The loading image displays to let the driver know that the system is processing	As expected	Successful
WB006 - 03	Successful geo coordinate load	None	The system displays the shortest path between two geographical coordinates	As expected	Successful

Table 45: Test view current order location on a map function

2.4.7. View the details of an order available

Test case code: WB007

Condition: The tester must log into the system with a registered account.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB007 - 01	An error occurred during the process of obtaining information	None	The system will display an error message during data retrieval	As expected	Successful

	n from the server.				
WB007 - 02	Get information from the server successfully.	None	Information is successfully retrieved and displayed to the screen	As expected	Successful

Table 46: Test View the details of an order available function

2.4.8. Receive order

Test case code: WB008

Condition: The tester must log into the system with a registered account.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB008 - 01	An error occurred during the delivery of an order request to the server	None	The system will display an error message	As expected	Successful
WB008 - 02	Send a successful order request to the server	None	The system displays a successful order receive message	As expected	Successful

Table 47: Test Receive order function

2.4.9. Update geographic coordinates

Test case code: WB009

Condition: The tester must log into the system with a registered account, and the account received an order.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB009 - 01	An error occurred during the driver's current geographic coordinate retrieval.	None	The system will display an error message	As expected	Successful

WB009 - 02	An error occurred during the sending of information to the server.	None	The system will display an error message	As expected	Successful
WB009 - 03	Successfully update the current geographic location of the order	None	The system displays successful update notifications	As expected	Successful

Table 48: Test update geographic coordinates function

2.4.10. Update the status of an order

Test case code: WB0010

Condition: The tester must log into the system with a registered account, and the account received an order.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB0010 - 01	An error occurred during an order status update	None	The system will display an error message	As expected	Successful
WB0010 - 02	Successfully update the status of the order	None	The system will notify that the status of the order has been updated	As expected	Successful

Table 49: Test update the status of an order function

2.4.11. Confirmation of payment

Test case code: WB0011

Condition: The tester must log into the system with a registered account, and the account received an order.

Case code	Case	Input data	Expected results	Actual results	Evaluate
WB0011 - 01	An error occurred during the payment	None	The system will display an error message	As expected	Successful

	confirmati on process				
WB0011 - 02	Successful ly update the paid status for the order	None	The system will notify that the order has been successfully paid for.	As expected	Successful

Table 50: Test confirmation of payment function

CONCLUSION

In this section, I summarized the results we have achieved through the time of working and implementing this project. At the same time, I would like to outline some directions for the development and improvement of this project in the future.

1. Summary of results

After the process of research and implement. I have built a system that includes two main applications: mobile application, web applicatiton, and manager page integrated into the Django framework. I also learned how to use the leaflet library integrated with reactjs and react-native frameworks, as well as gain experience using different programming languages, and frameworks in the same project.

2. Future works

Currently, I have only created a simple freight management system, in the future, I want to create a system that can manage greater freight frequency, accompanied by an application for managers built individually to be easier to manage.

REFERENCES

- [1] <https://www.vietqr.io/danh-sach-api/api-danh-sach-ma-ngan-hang>
- [2] <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>
- [3] [https://en.wikipedia.org/wiki/React_\(JavaScript_library\)](https://en.wikipedia.org/wiki/React_(JavaScript_library))
- [4] <https://provinces.open-api.vn/>
- [5] [https://vapi-vnappmob.readthedocs.io/en/latest/province.html#get--api-province-district-\(string-province_id\)](https://vapi-vnappmob.readthedocs.io/en/latest/province.html#get--api-province-district-(string-province_id))
- [6] <https://realpython.com/getting-started-with-django-channels/>
- [7] <https://www.byprogrammers.com/2020/11/how-to-integrate-google-maps-in-react-native-app/>
- [8] <https://www.byprogrammers.com/2020/11/how-to-generate-google-maps-api-key-for-mobile-app/>
- [9] <https://viblo.asia/p/react-native-location-tracking-XL6lANBA5ek>