**CAN THO UNIVERSITY**

**COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**🙞 🕮 🙜**

**Logo

Description automatically generated**

**GRADUATION THESIS IN**

**INFORMATION TECHNOLOGY**

**(HIGH-QUALITY PROGRAM)**

**EXERCISE CORRECTION WITH MACHINE LEARNING**

**Student: Ngô Hồng Quốc Bảo**

**Student ID: B1809677**

**Class: 2019 - 2023 (K44)**

**Advisor: Prof. Trần Công Án**

**Can Tho, 12/2022**

**CAN THO UNIVERSITY**

**COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**🙞 🕮 🙜**

**Logo

Description automatically generated**

**GRADUATION THESIS IN**

**INFORMATION TECHNOLOGY**

**(HIGH-QUALITY PROGRAM)**

**EXCERCISE CORRECTION WITH MACHINE LEARNING**

**Student: Ngô Hồng Quốc Bảo**

**Student ID: B1809677**

**Class: 2019 - 2023 (K44)**

**Advisor: Dr. Trương Quốc Định**

**Can Tho, 12/2022**

INSTRUCTOR’S COMMENTS

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Can Tho, ………………. 2022

(Instructor sign and write full name)

**ACKNOWLEDGMENT**

*I cannot express enough thanks to my instructors/teachers from the College of Information and Communication technology for their continued support and encouragement. My sincere thanks especially go to Prof. Trần Công Án for his guidance and advices throughout the development of this project*.

*My completion of this project could not have been accomplished without the support of my classmates. The supports and advices from them have always been the biggest motivation for me during the whole process. Their encouragement when the times got rough are much appreciated and duly noted.*

*Last but not least, I would like to thank my parents who helped me a lot in gathering different information, collecting data and guiding me, despite their busy schedules, they had always been there with me from time to time in making project.*

Can Tho, December 2022

Student

Ngô Hồng Quốc Bảo

**ABSTRACT**

Fitness is becoming an important part of human life as it brings many beneficial to personal health. However, exercises can also be ineffective and could be dangerous if they are performed incorrectly by the performer. In my project, I use machine learning to provides detailed analysis and recommendation on the performer’s exercises to improve their form.

In addition, a wide range of potential Machine Learning applications today rely on several fundamental baseline Machine Learning tasks. The development of Mediapipe from Google. MediaPipe provides cornerstone Machine Learning models for common tasks like hand tracking, posture detection, ... Make use of the power of Mediapipe’s pose detection, this project “Exercise Correction with Machine Learning” is built in order to analyze, detect and classifying the forms of fitness exercises. The experimental results show that the algorithm proposed in this paper can effectively identify correct and incorrect forms that are performed in a exercise.

**TABLE OF** **CONTENTS**

[INSTRUCTOR’S COMMENTS 1](#_Toc89681424)

[INTRODUCTION 10](#_Toc89681425)

[1. The purpose of the study 10](#_Toc89681426)

[2. The object and scope of the study 10](#_Toc89681427)

[3. Project methodology and approach 10](#_Toc89681428)

[4. Criteria for the project’s success 10](#_Toc89681429)

[5. Project contents 11](#_Toc89681430)

[CONTENTS 12](#_Toc89681431)

[CHAPTER 1: REQUIRED SPECIFICATION 12](#_Toc89681432)

[1.1. Delivery management system brief overview 12](#_Toc89681433)

[1.2. Main functionality 12](#_Toc89681434)

[1.2.1. Order management 12](#_Toc89681435)

[1.2.2. Employee Management 12](#_Toc89681436)

[1.2.3. Customer Management 12](#_Toc89681437)

[1.2.4. Delivery pricing list management 12](#_Toc89681438)

[1.2.4. Order tracking using Leaflet-routing technology 13](#_Toc89681439)

[1.2.5. Statistical report 13](#_Toc89681440)

[1.3. Other functionalities 13](#_Toc89681441)

[CHAPTER 2: AN OVERVIEW OF TECHNOLOGY USED 14](#_Toc89681442)

[1. Web Technology 14](#_Toc89681443)

[1.1. Django framework 14](#_Toc89681444)

[1.1.1. Django 14](#_Toc89681445)

[1.1.2. Django Rest Framework 15](#_Toc89681446)

[1.2. Vue.js – A Frontend JavaScript framework 16](#_Toc89681447)

[2. Leaflet 16](#_Toc89681448)

[2.1. What is Leaflet? 16](#_Toc89681449)

[2.2. Leaflet Routing Machine 17](#_Toc89681450)

[3. Open Route Service API 17](#_Toc89681451)

[CHAPTER 3: SYSTEM DESCRIPTION 19](#_Toc89681452)

[3.1. System overview 19](#_Toc89681453)

[3.2. Feature requirements 20](#_Toc89681454)

[3.2.1. Login 21](#_Toc89681455)

[3.2.2. Register 22](#_Toc89681456)

[3.2.3. Order searching 22](#_Toc89681457)

[3.2.4. Order placement 23](#_Toc89681458)

[3.2.5. Order Management 25](#_Toc89681459)

[3.2.6. Accounts Management 26](#_Toc89681460)

[3.3.7. Pricing list management 27](#_Toc89681461)

[3.2.8. Statistical reports 28](#_Toc89681462)

[3.3. System requirements 28](#_Toc89681463)

[3.4. User Interface 29](#_Toc89681464)

[CHAPTER 4: DELIVERY SYSTEM DESIGN 30](#_Toc89681465)

[1. DATABASE DESIGN: 30](#_Toc89681466)

[2. FINDING AND NAVIGATING BETWEEN LOCATIONS 33](#_Toc89681467)

[2.1. Finding consignee location 34](#_Toc89681468)

[2.2. Navigating and distance calculating for order delivery 35](#_Toc89681469)

[3. USER INTERFACE DESIGN 36](#_Toc89681470)

[3.1. Login/Register page 36](#_Toc89681471)

[3.2. Orders Management page 36](#_Toc89681472)

[3.3. Employee Management page 37](#_Toc89681473)

[3.4. Profile page 39](#_Toc89681474)

[3.5. Pricing list management page 39](#_Toc89681475)

[3.6. Reports page 40](#_Toc89681476)

[3.7. Order Detail page 41](#_Toc89681477)

[3.8. Order placement page 42](#_Toc89681478)

[CHAPTER 5: TESTING AND EXPERIMENTAL RESULTS 44](#_Toc89681479)

[1. The purposes of testing 44](#_Toc89681480)

[2. Testing explanation 44](#_Toc89681481)

[2.1. Details of the testing plan 44](#_Toc89681482)

[2.1.1. Testing features 44](#_Toc89681483)

[2.1.2. The goal of testing 45](#_Toc89681484)

[2.1.3. Result from the test cases 45](#_Toc89681485)

[2.2. Test environment 45](#_Toc89681486)

[2.3. Test cases 46](#_Toc89681487)

[2.3.1. Login 46](#_Toc89681488)

[2.3.2. Register 47](#_Toc89681489)

[2.3.3. Add/Edit pricing list 47](#_Toc89681490)

[2.3.4. Search for order by its id 48](#_Toc89681491)

[2.3.5. Find consignee information 49](#_Toc89681492)

[2.3.6. Navigate and calculate distance 50](#_Toc89681493)

[CONCLUSION 51](#_Toc89681494)

[1. Final results 51](#_Toc89681495)

[2. Future works 51](#_Toc89681496)

[REFERENCES 52](#_Toc89681497)

**LIST OF FIGURES**

[Figure 1: Introduction - Django 14](#_Toc89682426)

[Figure 2: Introduction - Django REST framework 15](#_Toc89682427)

[Figure 3: Introduction - Vue,js 16](#_Toc89682428)

[Figure 4: Introduction - Leaflet 17](#_Toc89682429)

[Figure 5: Introduction - Open Route Service 18](#_Toc89682430)

[Figure 6: Use case diagram of manager in the system 20](#_Toc89682431)

[Figure 7: Use case diagram of customer in the system 21](#_Toc89682432)

[Figure 8: Database diagram of the system 30](#_Toc89682433)

[Figure 9: System Screenshot - Register page 36](#_Toc89682434)

[Figure 10: System Screenshot - Login page 36](#_Toc89682435)

[Figure 11: System Screenshot - Order management page 37](#_Toc89682436)

[Figure 12: System Screenshot - Employee management 38](#_Toc89682437)

[Figure 13: System Screenshot - Consignor management page 38](#_Toc89682438)

[Figure 14: System Screenshot - Consignee management page 39](#_Toc89682439)

[Figure 15: System Screenshot - Profile page 39](#_Toc89682440)

[Figure 16: System Screenshot - Pricing list page on user view 40](#_Toc89682441)

[Figure 17: System Screenshot - Pricing list page on manager view 40](#_Toc89682442)

[Figure 18: System Screenshot - Report page 41](#_Toc89682443)

[Figure 19: System Screenshot - Order detail page 41](#_Toc89682444)

[Figure 20: System Screenshot - Tracking order in Order detail page 42](#_Toc89682445)

[Figure 21: System Screenshot - Order placement page 42](#_Toc89682446)

[Figure 22: System Screenshot - Order confirmation 43](#_Toc89682447)

**LIST OF TABLES**

[Table 1: Feature description - Login 21](#_Toc89681520)

[Table 2: Feature description - Register 22](#_Toc89681521)

[Table 3: Feature description - Order searching 23](#_Toc89681522)

[Table 4: Feature description - Order placement 24](#_Toc89681523)

[Table 5: Feature description - Order management 25](#_Toc89681524)

[Table 6: Feature description - Account management 26](#_Toc89681525)

[Table 7: Feature description - Pricing list 27](#_Toc89681526)

[Table 8: Feature description - Statistical report 28](#_Toc89681527)

[Table 9: Database design - User table 31](#_Toc89681528)

[Table 10: Database design - Profile table 31](#_Toc89681529)

[Table 11: Database design - Order table 32](#_Toc89681530)

[Table 12: Database design - Unit price table 33](#_Toc89681531)

[Table 13: Database design - Product table 33](#_Toc89681532)

[Table 14: Database design - Delivery table 33](#_Toc89681533)

[Table 15: Features require to be tested 45](#_Toc89681534)

[Table 16: Feature test cases - Login 46](#_Toc89681535)

[Table 17: Feature test cases - Register 47](#_Toc89681536)

[Table 18: Feature test cases - Add/Edit pricing list 48](#_Toc89681537)

[Table 19: Feature test cases - Order searching 49](#_Toc89681538)

[Table 20: Feature test cases - Find consignee location 50](#_Toc89681539)

[Table 21: Feature test cases - Navigate and calculate distance 50](#_Toc89681540)

INTRODUCTION

1. The purpose of the study

The main purpose of this project is to build a fully functional delivery system for a delivery company. The main functions are to manage the process of delivering and receiving package/goods from the customer, employee management and generating statistical report for the company. The system not only needs to be features rich but also user friendly.

1. The object and scope of the study

The delivery management system can be utilized by two types of users. However, it is created mainly for the managers of a delivery company that needs to monitor the process of receiving and delivering packages for customers. Because those processes are performed by the company’s drivers, the employee management is also an important feature of this system. Beside the company’s employees, customer’s information (both consignor and consignee) that relate to the delivery process is also kept and manage by the system. Besides, these main features, managers can also use the system to control their price tags or keep track of delivering orders and others additional features.

The second type of users who can make use of the system are users. Users can place orders, configure their location, information on for their orders. Users can also keep track of their own delivering orders.

1. Project methodology and approach

To be able to successfully build this project, I did a lot of analysis on how popular delivery systems in Vietnam such as Giao hàng nhanh, Giao hàng tiết kiệm setup their systems. I also researched on how they charge their system by looking into their price tags on different distances of delivery or the type of delivery that they offer. Finally, I inspect their websites’ user interface to take note in order to build an easy-to-user interface for my system.

1. Criteria for the project’s success

A delivery system needs to fulfil all of the below basic requirements:

* Company employee management.
* Customer information management including consignors and consignees.
* Orders management.
* Pricing List management.
* Accurate statistical reports.
* Monitoring delivering orders.

Additionally, there are several other features to assist the management process or to enhance the user experience when using the system.

1. Project contents

- **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: Required specifications

+ Chapter 2: An overview of technology used

+ Chapter 3: System descriptions

+ Chapter 4: Delivery system design

+ Chapter 5: Testing and experimental results

- **Conclusion**: Present the results achieved and the development direction of the system

CONTENTS

1. REQUIRED SPECIFICATION

1.1. Delivery management system brief overview

The delivery management system is a web-based application which is built in order to assist managing a delivery company. The purpose of the system is to simplify and automate a lot of tasks of a delivery company such as employee management, order management, pricing management and tracking the order location. It is simple to understand and can be used by anyone even if he/she is not familiar with the system before. This software project has been developed using Vue.js framework as a frontend and Python, MySQL for backend. The system has multi-user approach (built for manager and customer).

1.2. Main functionality

1.2.1. Order management

Every order which is placed by customer along with its information such as valid information of consignor and consignee that related to for the receive and deliver process will be saved to the database. In addition, other information such as the created date, or the package details will all be contained. The system can then make use of saved information to manage the state of the orders (for example checking whether or not the order is delivering). The managers can access to any order’s information, beside monitoring, they also can perform C.R.U.D (create, read, update and delete) actions to any specific orders.

1.2.2. Employee Management

There are two types of employee that verifies by the system: manager (admin) and driver. Information about the company’s employee such as name, address, phone number and driver license (for driver) can be access and manage through the system.

1.2.3. Customer Management

To be able to place an order on the system, customer as a consignor need to register an account on the system. Information such as full name, address or phone number of the customer account will be stored so that the customer don’t have to type them again if he/she makes another other. Apart from personal information, the system also can tell how many orders an account had placed and the status of those orders (successfully delivered or not).

A consignee does not require to register an account, however the information that related to an order will still be kept by the system.

1.2.4. Delivery pricing list management

The price of a delivery order depends on the distance that the company’s drivers have to travel to deliver the package, in order words, the distance between the company and the consignee’s location.

The managers are the only ones can manage or perform actions such as update, delete, add … to this category.

1.2.4. Order tracking using Leaflet-routing technology

When an order is at delivering state, which means it is being delivered by a company driver. Managers and users that related to that order can track its location using the system, therefore have access to the latest information to that order.

1.2.5. Statistical report

The system can automate tasks such as calculate, sort and counting orders and the company revenue then display them. Managers are the only type of user that can view these statistics.

1.3. Other functionalities

Beside the main functionalities that require for the system, it also equip with server other ones to improve the system overall and enhance the user experience.

* **Assign / Unassign order to the driver**: manager can directly assign processing orders to the drivers on the system to indicate that the driver is delivering those orders.
* **Order preview:** helps non-account user can view limited information of an order.
* **Search for order:** helps user to has faster access to their order’s information.

1. AN OVERVIEW OF TECHNOLOGY USED
2. Web Technology

1.1. Django framework

1.1.1. Django



Figure : Introduction - Django

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. At first, Python was developed to perform on Unix platform. However, overtime Python was gradually growth to compatible to a lot more operating systems such as MS-DOS, Mac OS, OS/2, Windows, Linux and other Unix operating systems.

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005.

Django is "somewhat opinionated" framework, and hence delivers the "best of both worlds". It provides a set of components to handle most web development tasks and one (or two) preferred ways to use them. However, Django's decoupled architecture means that you can usually pick and choose from a number of different options, or add support for completely new ones if desired

1.1.2. Django Rest Framework

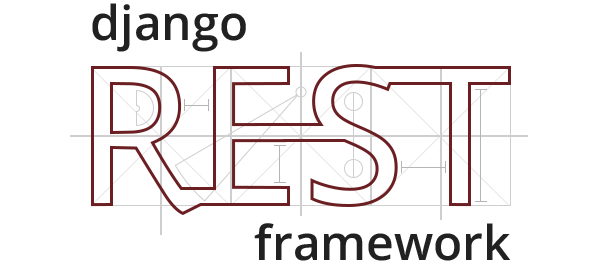


Figure : Introduction - Django REST framework

The Django REST Framework is a flexible and robust tool kit that makes it easy for developers to build web APIs. It also offers class-based generalized views and serializers for the API. Being a source code, it is abbreviated as DRF, which represents a Python library for developing web application programming interfaces. As in this article, we are talking about development APIs with step-by-step instructions. But before getting started with the guide, it is important to understand why the framework is even important. Many available frameworks allow developers to build APIs for their block applications easily, but the Django REST framework is preferred. This framework is convenient to use in a number of ways and offers the below-mentioned advantages:

* It offers web browsers an able Application Programming Interface, which is a great win for developers.
* It has authentication policies including the packages for OAuth1, OAuth2, etc …
* The serialization processor in it supports the ORM (Object-Relational Mapping - a technique that lets you query and manipulate data from a database using an object-oriented paradigm) and non-ORM data sources.
* Django REST framework has extensive documentation and offers great community support.
* It is utilized and trusted by great platforms, including Mozilla, Heroku, RedHat, etc …

1.2. Vue.js – A Frontend JavaScript framework



Figure : Introduction - Vue,js

Vue is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

Vue was released way back in 2014. Since then, it is continuously evolving. At the beginning of 2018, Vue.js started beating Angular and becoming more famous in the market. Later, in September 2018, Evan You decided to announce the release of Vue 3.0. Vue.js is continuously evolving with the rapid growth in the usage and community of this framework. The community will keep growing because it was built on the best features combination of Angular and React.

1. Leaflet

2.1. What is Leaflet?



Figure : Introduction - Leaflet

Leaflet is an open-source JavaScript library used to build web mapping applications. First released in 2011, it supports most mobile and desktop platforms, supporting HTML5 and CSS3. Among its users are FourSquare, Pinterest and Flickr.

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. Leaflet allows developers without a GIS background to very easily display tiled web maps hosted on a public server, with optional tiled overlays. It can load feature data from GeoJSON files, style it and create interactive layers, such as markers with popups when clicked.

It is developed by Vladimir Agafonkin, who joined Mapbox in 2013.

2.2. Leaflet Routing Machine

Leaflet Routing Machine is an easy, flexible and extensible way to add routing to a Leaflet map. Using the default is just a few lines of code to add fully functional routing, but you can still customize almost every aspect of the user interface and interactions.

Leaflet Routing Machine supports multiple features such as:

* Standard Leaflet control, with Leaflet look and feel.
* Routing from start to destination, with possibility of via points.
* Support for several routing engines such as Open-Source Routing Machine.
* Highly customizable for advanced use.

3. Open Route Service API



Figure : Introduction - Open Route Service

The Open Route Service API provides global spatial services by consuming user-generated and collaboratively collected free geographic data directly from OpenStreetMap. It is highly customizable, performant and written in Java.

The following services are available via a HTTP interface served by Tomcat.

* Directions - Returns a route between two or more locations for a selected profile with customizable additional settings and instructions.
* Isochrones - Obtains areas of reachability from given locations.
* Matrix - Computes one-to-many, many-to-one or many-to-many routes for any mode of transport provided by Open Route Service.

1. SYSTEM DESCRIPTION

3.1. System overview

As mentioned in the main purpose, there are in total three types of users that the delivery system was developed for. Every kind of user need to login into an account with a unique email to use the system main features. After login, the system will automatically evaluate what type of user an account belongs to and provide appropriate features accordingly. There are several types of information that can be edited by the account owner such as name, address or phone number, etc.

* For a manager, after login to the system, user can manage orders, pricing lists, employees and customers that involve with the company.
  + Manager can perform C.R.U.D actions (create, read, update or delete) on the orders in the systems. Every order has a unique id, belong to one consignor and one consignee, its state can be failed, processing, delivering or delivered. If it is in delivering state, its shipper and location are also visible to the user.
  + Manager can also perform C.R.U.D actions to the pricing list of the company. Each pricing list contains information about the lower and upper distance that a company’s driver has to travel to deliver an order, and the price tag correspond to that distance.
  + Every active account (employee or customer) can be managed by the manager. Manager can view other accounts’ information but cannot edit them.
  + Manger also can access to the report page of the system with contains the statistic information of the such as number of orders and revenue of the company in a period of time.
* For a user, after login to the system, they can manage their own history orders and place a new order. When placing a new order, user is required to fill necessary information to that order such as, consignee’s name, address, phone number, the packages from these orders. He/she can also set how they want to a package to be delivered (such as if a consignee allows to try the items inside a package) or can simply leave a note to when place an order.

Beside all the management features, one of the most important features of the delivery system is that manager or user can track a delivering order’s location. When an order is placed, it will come to “processing” state, after being processed, that order can be assigned to a driver and come to “delivering” state, from there, the order’s location can be tracked by using the driver’s location. Whenever, the driver updates his/her location, the correspond orders’ locations that are being delivered will be update on the system. This process will continue until the orders are delivered or failed to deliver.

3.2. Feature requirements

The two below figures are the use case diagrams for the manager and customer of the system. The details of these features are displayed from table 1 to table 8.



Figure : Use case diagram of manager in the system

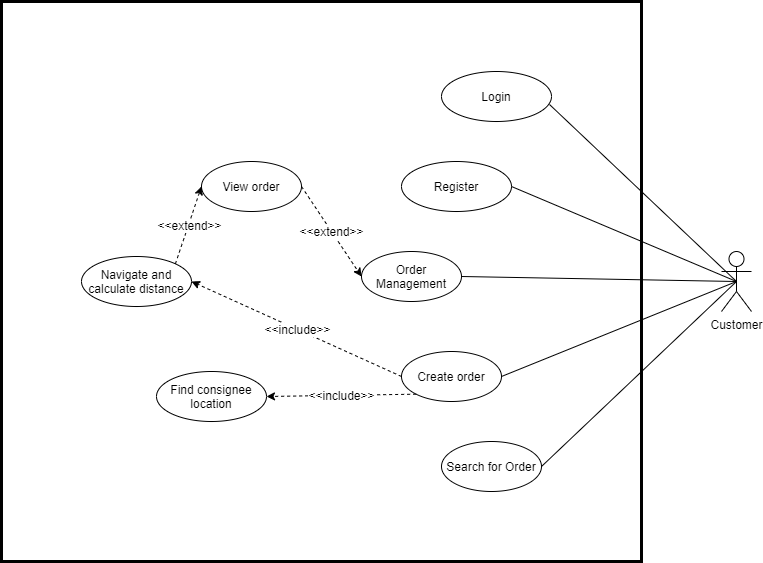


Figure : Use case diagram of customer in the system

3.2.1. Login

| **Feature:** Login | **Feature ID:** FT001 |
| --- | --- |
| **Agent:** Manager and Customer | **Priority:** High |
| **Description:** Allow user to login into the system. | |
| **Condition:** User’s email is registered on the system | |
| **Operation:**  User goes to the login page and fill out the two following fields: email and password. Then the system will verify the inputs. | |
| **Result:**   * If the inputs are verified, user are redirected to the Order management page. * Otherwise, the system will display an error and ask the user to retype the inputs. | |
| **Notes:** | |

Table : Feature description - Login

3.2.2. Register

| **Feature:** Register | **Feature ID:** FT002 |
| --- | --- |
| **Agent:** Customer | **Priority:** High |
| **Description:** Allow user to register an email to use the delivery system. | |
| **Condition:** | |
| **Operation:**  User goes to the login page and fill out the three following fields: email, password and confirm password. Then the system will verify the inputs. | |
| **Result:**   * If the inputs are valid, an account will be created with the user’s email then user are redirected to the Login page. * Otherwise, the system will display an error and ask the user to retype the inputs. | |
| **Notes:** | |

Table : Feature description - Register

3.2.3. Order searching

| **Feature:** Order searching | **Feature ID:** FT004 |
| --- | --- |
| **Agent:** Manager and Customer | **Priority:** High |
| **Description:** Allow user to search for any order by using its id. | |
| **Condition:** Logged-in user | |
| **Operation:**  User types in the search input in the navigation bar at the top of the page. | |
| **Result:**   * If the order id is valid:   + If the account is from the manager, the system will redirect to the Order detail page of that order.   + If the account is from the user, the system will redirect to the Order detail page of that order only when he/she owns that order. Otherwise, the Unauthenticated page will be redirected to. * Otherwise, the system will redirect to the 404 page if the order id is not exist. | |
| **Notes:** | |

Table : Feature description - Order searching

3.2.4. Order placement

| **Feature: Order placement** | **Feature ID:** FT004 |
| --- | --- |
| **Agent:** Manager and Customer | **Priority:** High |
| **Description:** User can place an order. | |
| **Condition:** Logged-in user | |
| **Operation:**  User clicks on the “Make order” button on the navigation bar. After redirect to the Order Create page, user is required to fill all the necessary information to place an order:   * Consignor information (if the below fields are updates by the consignor in the profile page, they will be prefilled):   + Full name   + Address   + Phone number * Consignee information:   + Full name   + Address   + Phone number * Information of all products that need to be delivered includes their name and prices * Other delivery information: note, payment method and consignee preview’s right. | |
| **Result:**   * If the inputs are valid and the customer hits the “Submit” button. The system will make a couple of API calls: * Post requests to Open Route Service API to retrieve the location of the consignee. * Post requests to Leaflet Routing Machine API to retrieve the estimated distance and time to travel from the company location to the consignee location. * Post requests to the backend API to retrieve the service price depends on the distance. * Otherwise, the system will display errors when the inputs are invalid and requires user to redo. | |
| **Notes:** | |

Table : Feature description - Order placement

3.2.5. Order Management

| **Feature:** Order management | **Feature ID:** FT005 |
| --- | --- |
| **Agent:** Manager, User | **Priority:** High |
| **Description:** Allow view or edit the detail of an order. If the order is in delivering state, manager or user can also track it location. | |
| **Condition:** Logged-in user | |
| **Operation:**  User goes to the Orders detail page of an order by clicking to an individual order in table of the Orders page or search an order using its id in the search bar. The Order detail page displays every information related to that order. Manager can view every order and normal user can view only the order that he/she owns.  If manager view an order detail, he/she can edit it by clicking the edit button, the system will redirect manager to the edit page. The editable information includes:   * The state of an order (processing, delivering, delivered or failed). * The payment methods (pay by consignor or pay by consignee). * Manager can also assign or un-assign a driver for that specific order.   If the order is in delivering state, at the bottom of the page there will be a map to display the current location of the order by detecting the current location of its driver. User can click on the “Focus in your order” button to has a close view on the location. | |
| **Result:**  The system will automatically update the changes right after the user finish his/her actions.  When user click on the “Focus in your order”, the map will be zoomed in and centered at the order’s location. | |
| **Notes:** | |

Table : Feature description - Order management

3.2.6. Accounts Management

| **Feature:** Accounts management | **Feature ID:** MN001 |
| --- | --- |
| **Agent:** Manager | **Priority:** High |
| **Description:** Manager can view all the information of an account (customer or company’s employee). | |
| **Condition:** Logged-in user and admin (manager) of the system | |
| **Operation:**  User clicks on the “Management” link on the sidebar. Every accounts registered to the system are displayed in 3 categories: employee, consignor or consignee. User can use the navigation slide to navigate between categories.  Depends on the category, manager has different operations on each of them.   * Employee account: manager can view the profile of each account. With the accounts that belong to drivers, manager can assign the orders for them by clicking the “Order” buttons. * Consignor account: manager can view the profile and how many orders owned by that account. * Consignee’s information: Consignee does not require an account, therefore his/her information is linked to his/her order. | |
| **Result:**  Depends on the user actions, the system will return different outcomes.   * View profile with employee and consignor’s account, the system will redirect to a Profile page where display all the given personal information about an account. * View profile of a consignee, the system will redirect to the Order detail page that owned by the consignee. * Assign order, the system will redirect to the “Assign Order” tab of the profile page where manager can assign any processing order to the driver. | |
| **Notes:** | |

Table : Feature description - Account management

3.3.7. Pricing list management

| **Feature:** Pricing list management | **Feature ID:** MN002 |
| --- | --- |
| **Agent:** Manager | **Priority:** High |
| **Description:** Manager can view all pricing list of the system, update, delete or create a unit price for the system. | |
| **Condition:** Logged-in user and admin (manager) of the system | |
| **Operation:**  User clicks on the “Pricing list” link on the sidebar. Pricing page displays the pricing list of the system. A unit price contains 3 properties: the lower distance, the upper distance and the price. If the distance to deliver a product falls into any of the unit price lower and upper distance that will be the price of that order.   * Create a new unit price: click the “Add price” button then fill 3 require fields: lower distance, upper distance and price, then click the “Submit” button. * Update an existence unit price: click on the edit icon on the unit price, edit wanted fields then click “Submit” button. * Delete an existence unit price: click the delete icon on the unit price. | |
| **Result:**  Depends on the user actions, the system will return different outcomes.   * When the user creates a new or updates existence unit price, if all fields are valid, the system will update the pricing list. * When the user delete an unit price and confirm the action, that unit price will be deleted. | |
| **Notes:** | |

Table : Feature description - Pricing list

3.2.8. Statistical reports

| **Feature:** Statistical reports | **Feature ID:** MN003 |
| --- | --- |
| **Agent:** Manager | **Priority:** High |
| **Description:** The system automatically generates statistic reports out of all the orders and the revenue from those orders. | |
| **Condition:** Logged-in user and admin (manager) of the system | |
| **Operation:**  User clicks on the “Report” link on the sidebar. Report pages display the count of all the orders and the revenue that came from them categorized by their staged (processing, delivering, delivered and failed). There are three timelines for the reports: today, last 7 days and this month. User can toggle between different timeline by clicking on the select dropdown. | |
| **Result:**  Depends on the user actions, the system will return different outcomes.   * The reports for “today” timeline will always be visible. * The chart to display the reports from “last 7 days” or “this month” will be toggle by the user. | |
| **Notes:** | |

Table : Feature description - Statistical report

3.3. System requirements

In order to build the delivery system, there are some requirements on both the software and hardware:

Hardware:

* Computer (Desktop or Laptop)
* Input device: Mouse
* Output device: Display panel

Software:

* Programming language: Python (version 3.7) and JavaScript (node.js version 14.17.4).
* Python libraries: Django (version 3.2.5) and Django REST framework (version 0.1.0).
* JavaScript libraries: Vue.js (version 2.6.11), leaflet (version 1.7.1) and leaflet-routing-machine (version 3.2.12).
* Code editor: VS Code or Sublime Text.
* Web Browser: Google Chrome, Edge or Firefox.

3.4. User Interface

The delivery system’s user interface is designed to be simple, clean and friendly for any kind of users to operate.

1. DELIVERY SYSTEM DESIGN

1. DATABASE DESIGN:

The below image represents how the database of delivery system is designed. There are 6 tables, the details of every table will be described from the tables below.

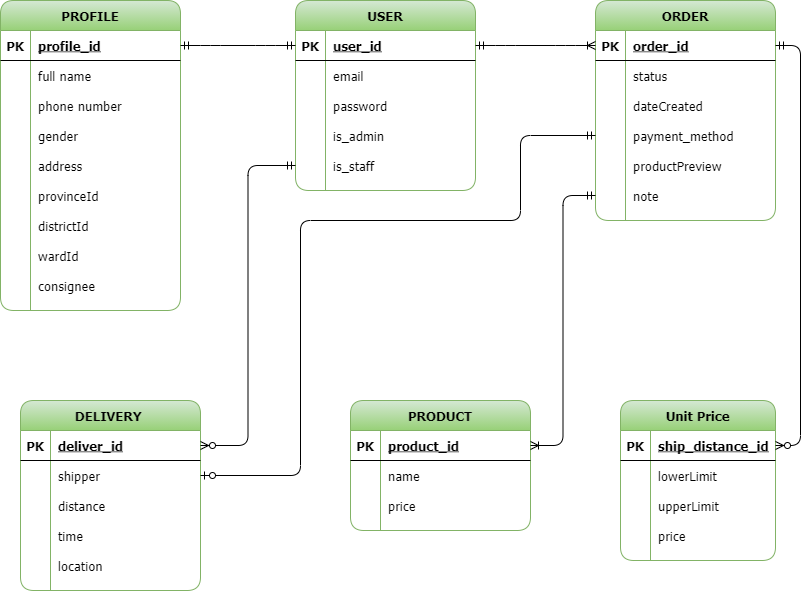


Figure : Database diagram of the system

The User table contains private and identity information about an account.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | user\_id | int | x | x | User identification |
| 2 | email | varchar |  | x | Account email for login and reset password |
| 3 | password | varchar |  |  | Account password for authentication |
| 4 | is\_admin | boolean |  |  | To identify if account is an admin (manager) |
| 5 | is\_staff | boolean |  |  | To identify if account is an employee of the system (driver or manager) |

Table : Database design - User table

The Profile table contains personal information about an account or a consignee that have used the system.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | profile\_id | int | x | x | Profile identification |
| 2 | full\_name | varchar |  |  | Account’s owner full name |
| 3 | phone | varchar |  |  | Account’s owner phone number |
| 4 | gender | boolean |  |  | Account’s owner gender |
| 5 | address | boolean |  |  | Account’s owner detail address, include house number plates and street names |
| 6 | province\_id | int |  |  | Vietnam’s province code |
| 7 | district\_id | int |  |  | Vietnam’s district code |
| 8 | ward\_id | int |  |  | Vietnam’s ward/subdistrict code |
| 9 | consignee | boolean |  |  | Check if this profile belong to a consignee of an order |

Table : Database design - Profile table

The Order table holds information of an order.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | order\_id | int | x | x | Order identification |
| 2 | status | varchar |  |  | Status of an order: processing (a consignor just places an order), delivering (the order are being delivered), delivered (the order’s successfully delivered) and failed (fail to deliver) |
| 3 | paymentMethod | int |  |  | How customers choose to pay for an order:  Pay by consignor  Pay by consignee |
| 4 | productPreview | int |  |  | Restriction when a consignee receives an order.  Consignee does not allow to observe the product.  Consignee allows to observe but not to try the product.  Consignee allows to try the product. |
| 5 | note | text |  |  | Any note from consignor for the driver or consignee. |
| 6 | dateCreated | datetime |  |  | Date and time when the order is created |

Table : Database design - Order table

The Unit Price table contains information about the price of an order depends on the distance to deliver.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | unit\_id | int | x | x | Unit price identification |
| 2 | lower\_limit | int |  | x | If the distance is greater than this attribute, the price of the order will be this unit price.  Otherwise, the price will belong to other unit price |
| 3 | upper\_limit | int |  | x | If the distance is lower than this attribute, the price of the order will be this unit price.  Otherwise, the price will belong to other unit price |
| 4 | price | int |  |  | The price corresponds to the 2 limits (Vietnam Dong) |

Table : Database design - Unit price table

The Product table holds information about each product from the package of an order.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | product\_id | int | x | x | Unit price identification |
| 2 | name | varchar |  |  | Name of a product |
| 3 | price | int |  |  | Price of a product |

Table : Database design - Product table

The Delivery table holds information about the delivery process of a delivering order.

| **No** | **Attribute** | **Data type** | **Primary**  **Key** | **Unique** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | deliver\_id | int | x | x | Delivery process identification |
| 2 | distance | float |  |  | The calculated distance has to travel to deliver an order when using Leaflet-Routing-Machine API (meter) |
| 3 | time | float |  |  | The calculated time has to travel to deliver an order when using Leaflet-Routing-Machine API (second) |
| 4 | location | varchar |  |  | The current location of an order, update according to the driver location. This attribute is saved as a JSON string format. |

Table : Database design - Delivery table

2. FINDING AND NAVIGATING BETWEEN LOCATIONS

One of the most important features of the system is calculate the distance between the consignee and the company location to announce the delivery price for the customer. Therefore, the system has to be able to identify the geolocation (latitude and longitude) of the consignee address. Besides pricing, determine consignee geolocation also helps with routing and navigating for easier delivery of an order and tracking its location.

2.1. Finding consignee location

The process of finding consignee location will be happened once per order, and it is start when the consignor submits their order placement in the Order Placement page.

The tools I used for finding location is Open Route Services API. It is a free API service that provide API endpoints for direction, finding geolocation, … In this system, I use it to find the geolocation of the consignee.

By feeding the Open Route Services API the address as a string format, the service will return a geolocation of that address if it is found by the service. Throughout out the process of using and testing this service, I realize that the service works at its best only when the address does not contain specific information such as house number plates or street names. Therefore, an address will contain only the ward, the district and the province/city.

The data returns from the API will be in a JSON format of an array. That array contains all the results that the API found about that address. For each element in the array, there are latitude and longitude of a location. In this case, I will choose the first element from the array as the position for the address as I found that it is the most reliable element.

Example of an address for API call: “Hưng Lợi, Cái Răng, Cần Thơ”

Example of a JavaScript code making POST request using axios library to the Open Routes Services API:

const url = "https://nominatim.openstreetmap.org/search?format=json&limit=3&q=" + encodeURIComponent(‘Hưng Lợi, Cái Răng, Cần Thơ’);

const { data } = await axios.get(url);

if (data.length > 0) {

const { lat, lon } = data[0];

}

Example of a geolocation of an address:

const data = {

lat: 10.01792665,

lon: 105.78839074112506

}

2.2. Navigating and distance calculating for order delivery

Leaflet Routing Machine API is the tool that I use for navigate and calculate the distance between the consignee and the company location.

The inputs for the API are the geolocation (latitude and longitude) of a consignee’s location and the company’s location.

The data returns from the API is packed with a lot of information such as the routes found from the two locations, the estimated travel distance (in meter), estimated time to travel (in second), the instructions to navigate to consignee locations ....

Example of inputs data for the API calls:

const data = {

lat: 10.01792665,

lon: 105.78839074112506

}

Example of using Leaflet Routing Machine API to navigate and calculate distance for the order to the consignee’s location

const routing = L.Routing.control({

waypoints: [

L.latLng({ lat: 10.001, lon: 105.123 }),

L.latLng({ lat: 10.001, lon: 105.567 }),

],

…

})

Example of a data that return from the API:

const data = {

summary: {

totalDistance: 63742.4,

totalTime: 3730.5,

},

instructions: [

{

directions: ‘S’,

distance: 91.1,

road: “Đường B14”,

}

…

],

…

}

3. USER INTERFACE DESIGN

The delivery system’s user interface is designed to be simple, clean and friendly for any kind of users to operate.

3.1. Login/Register page

Register page is for users to register an account to use the delivery system. Login page is for both user and manager of the system to login.

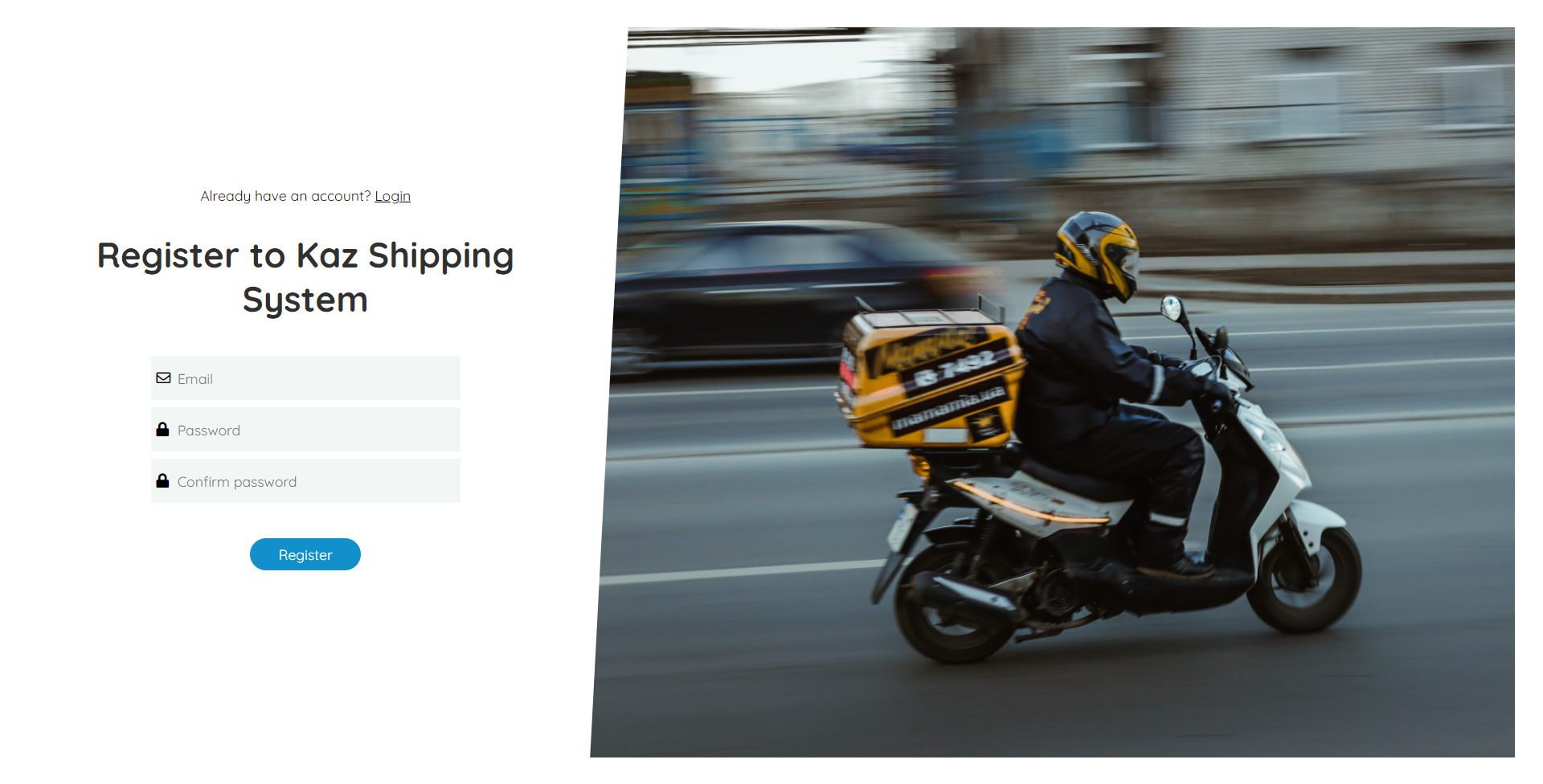


Figure : System Screenshot - Register page

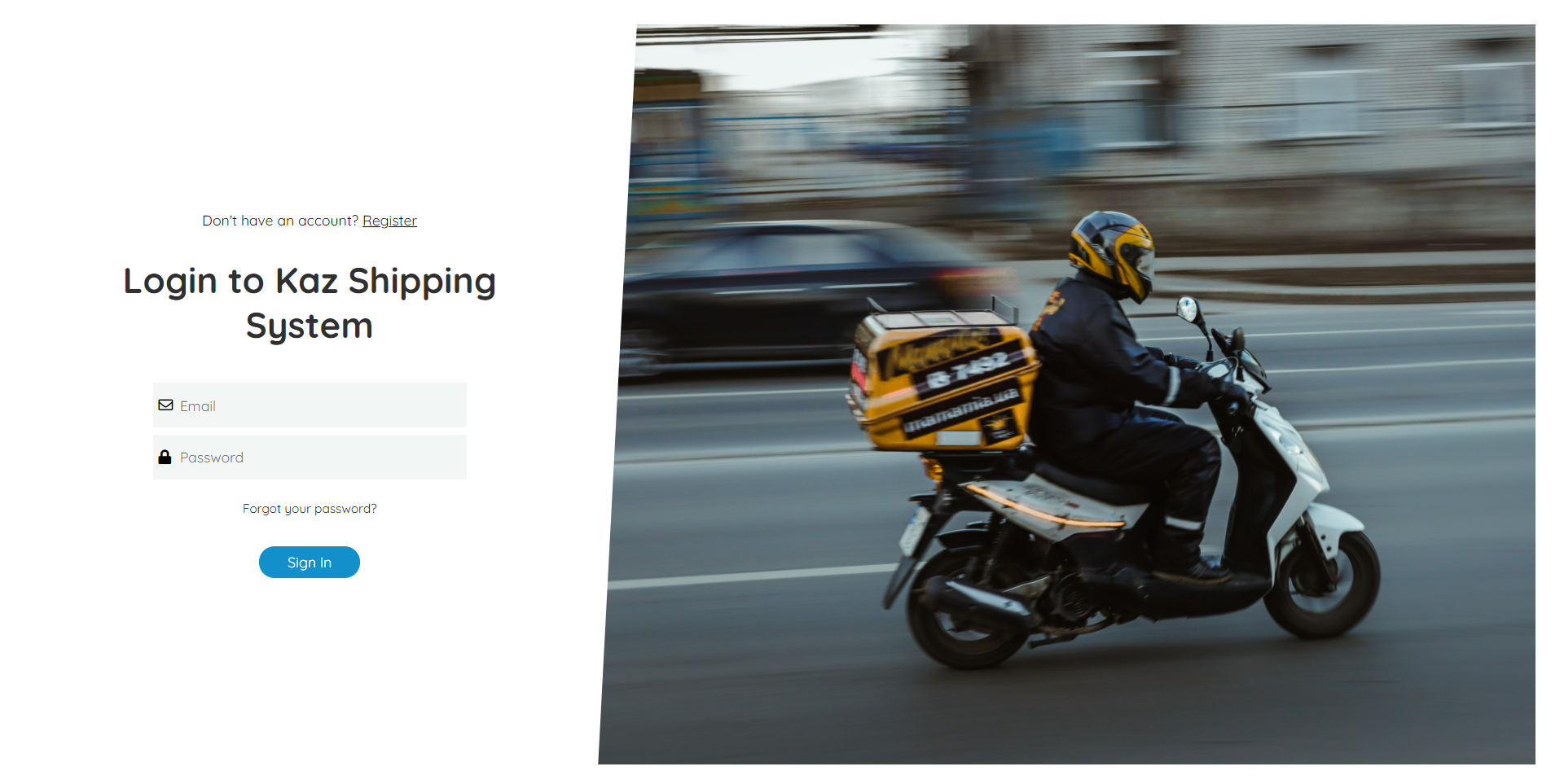


Figure : System Screenshot - Login page

3.2. Orders Management page

Orders management page helps users/manager to manage their orders. Managers can manage all the orders that are registered in the system. Users can only manage that they made.

Orders management page provides a filter tool for users to easily manage their orders. Users can filter by the orders’ status (processing, delivering, delivered or failed), the payment method of the orders (paid by consignee or by consignor) and the period of time that the orders had been made.

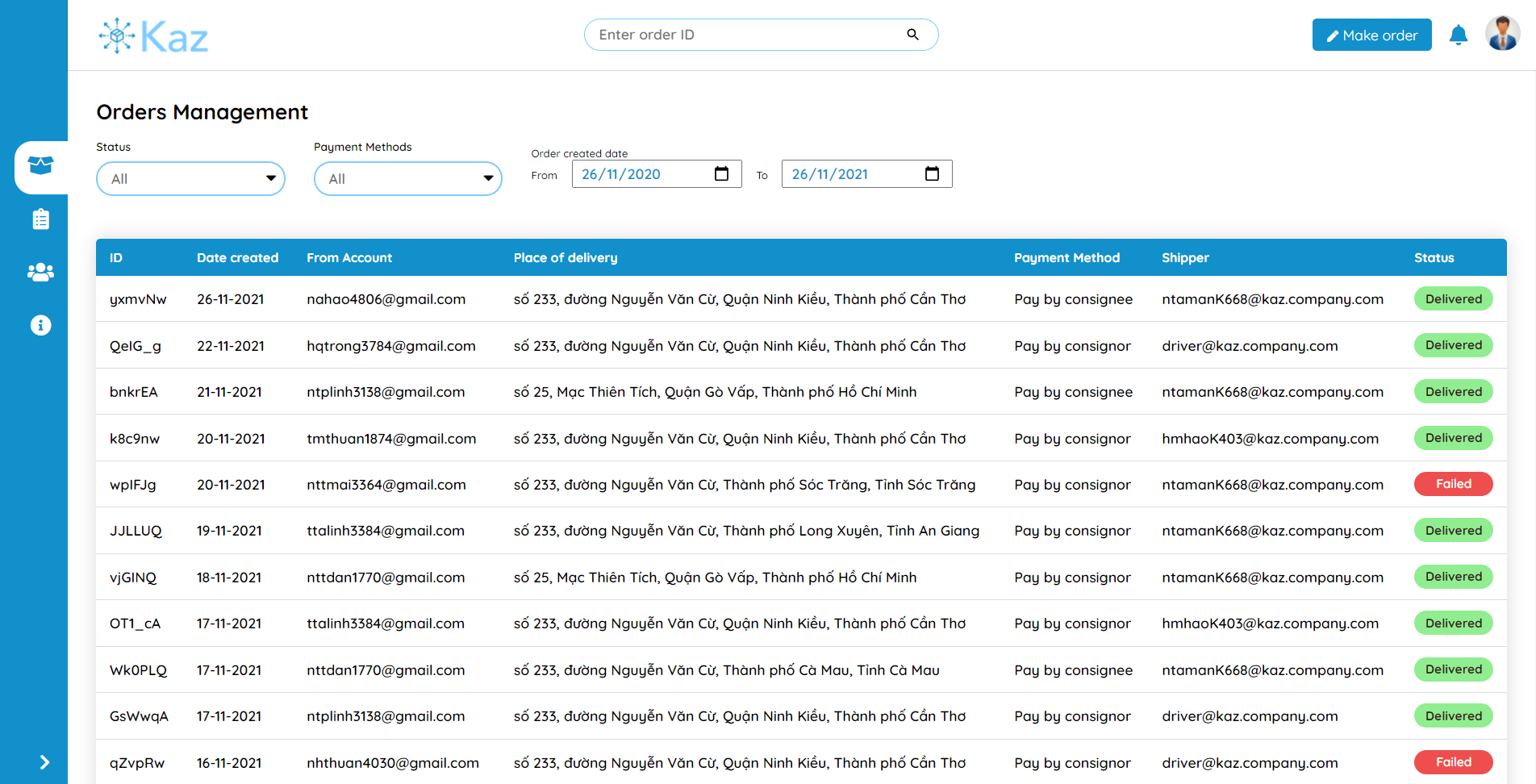


Figure : System Screenshot - Order management page

3.3. Employee Management page

The only-manager access page for manager to employees and their customers. There are 3 tabs (employee, consignor and consignee) for the managers to toggle, each tab will display the information correspond to its name.

For each employee, if it is a manager account, the other manager can click on the “Profile” button to view his/her profile which contains personal and work related information (name, phone number, …). Otherwise, if it is a driver account, the manager can view not only the account’s profile but also the orders which are delivering or delivered by him and the manager can also assign processing orders for that driver account by click on the “Orders” button.

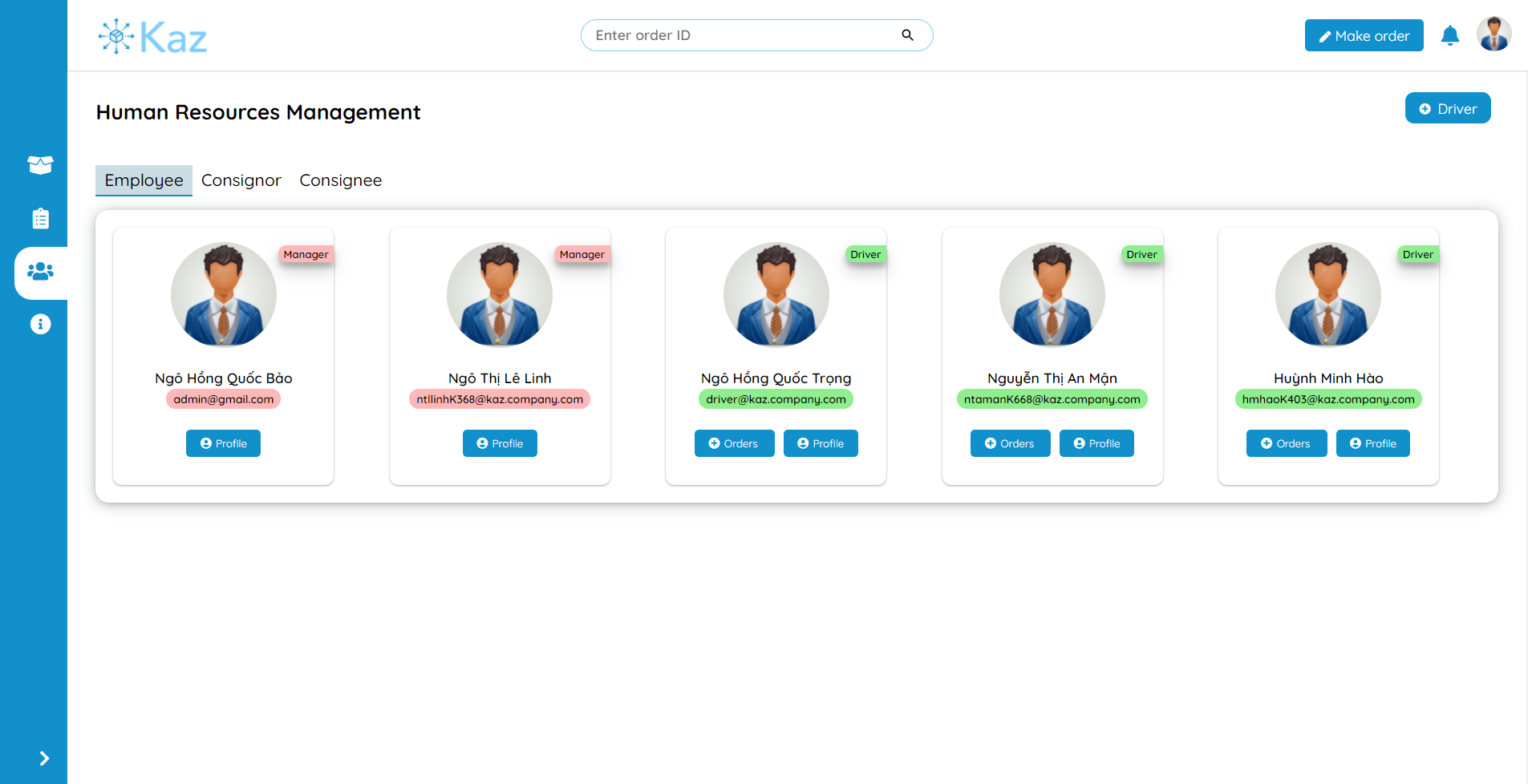


Figure : System Screenshot - Employee management

For the consignor tabs, the system will display a table of users that registered their account on the system. The manager can click on the row that corresponding to an account to view his/her information and the orders that he/she has placed.

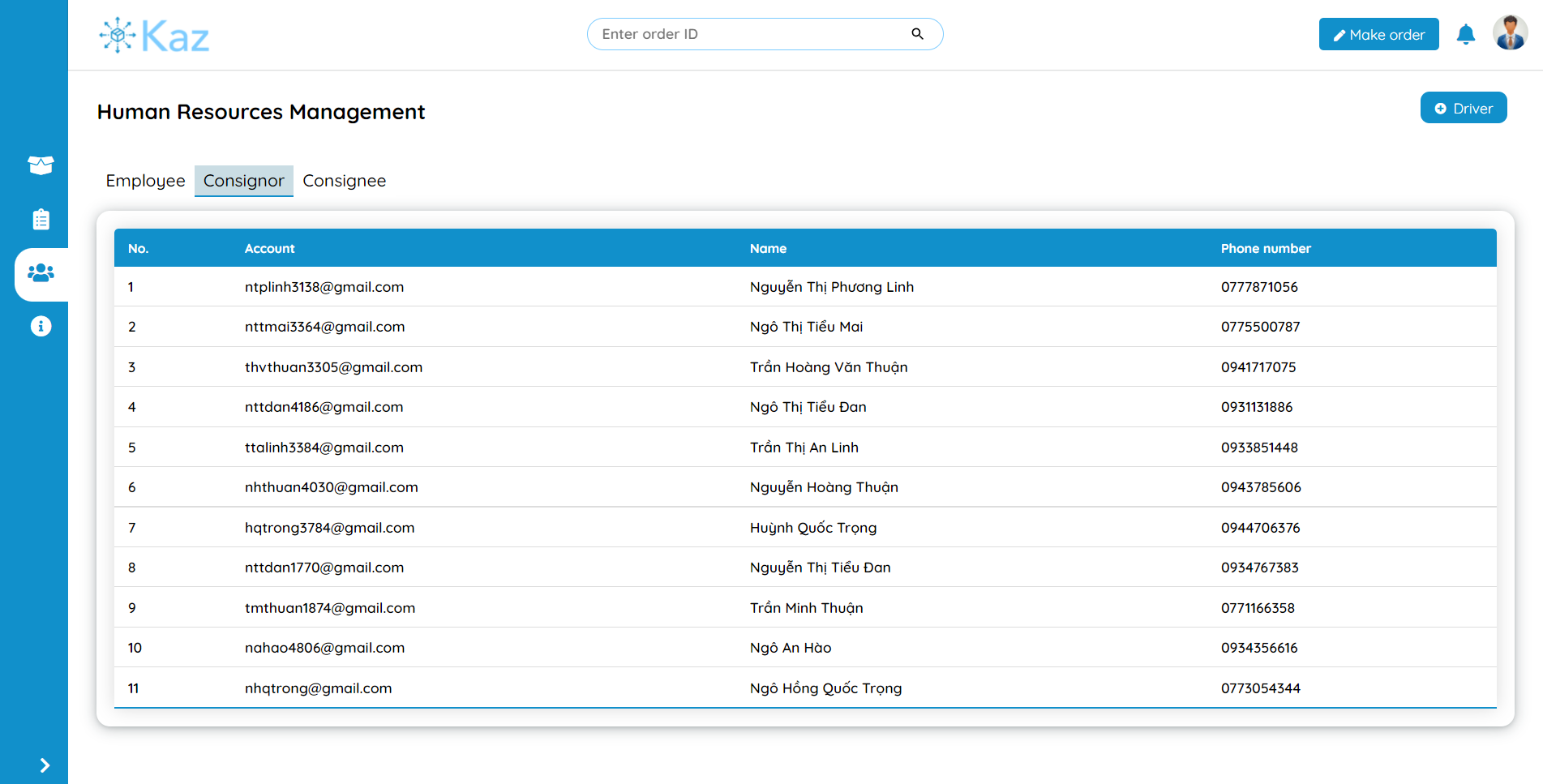


Figure : System Screenshot - Consignor management page

For the consignee tabs, the system will also display a table of consignee whose information has been recorded when a consignor account makes an order. Manager can click to that the row that that corresponding to a consignee to view all the information about that consignee with that his/her order.

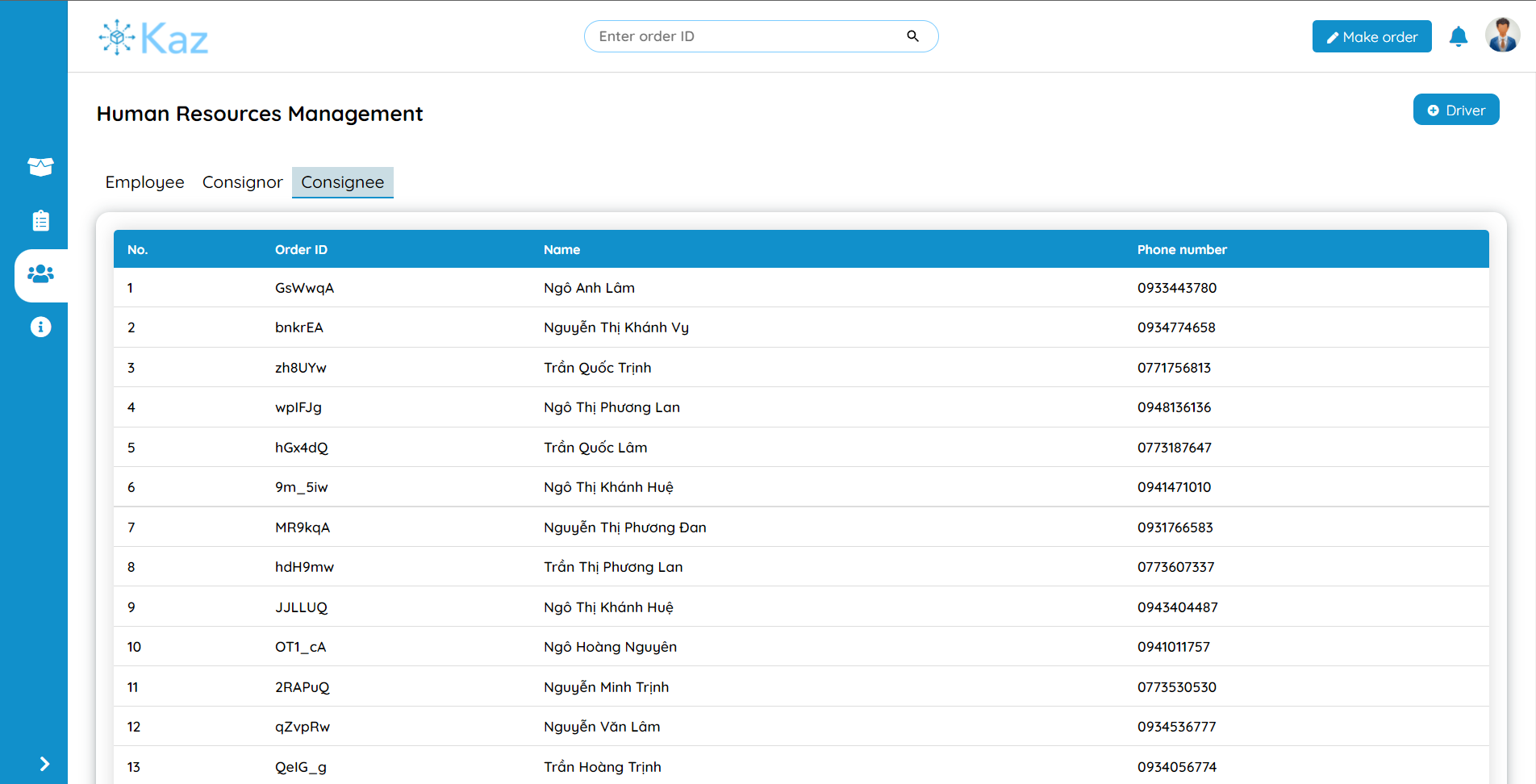


Figure : System Screenshot - Consignee management page

3.4. Profile page

Profile page displays the personal and work related information of an account. The user can use to the profile page to edit their own information. There is also an “Orders” tab that displays all the orders that an account has made.

If a manager account visits the “Profile” page of a driver account, the manager can also have a tab to assign orders to a driver.

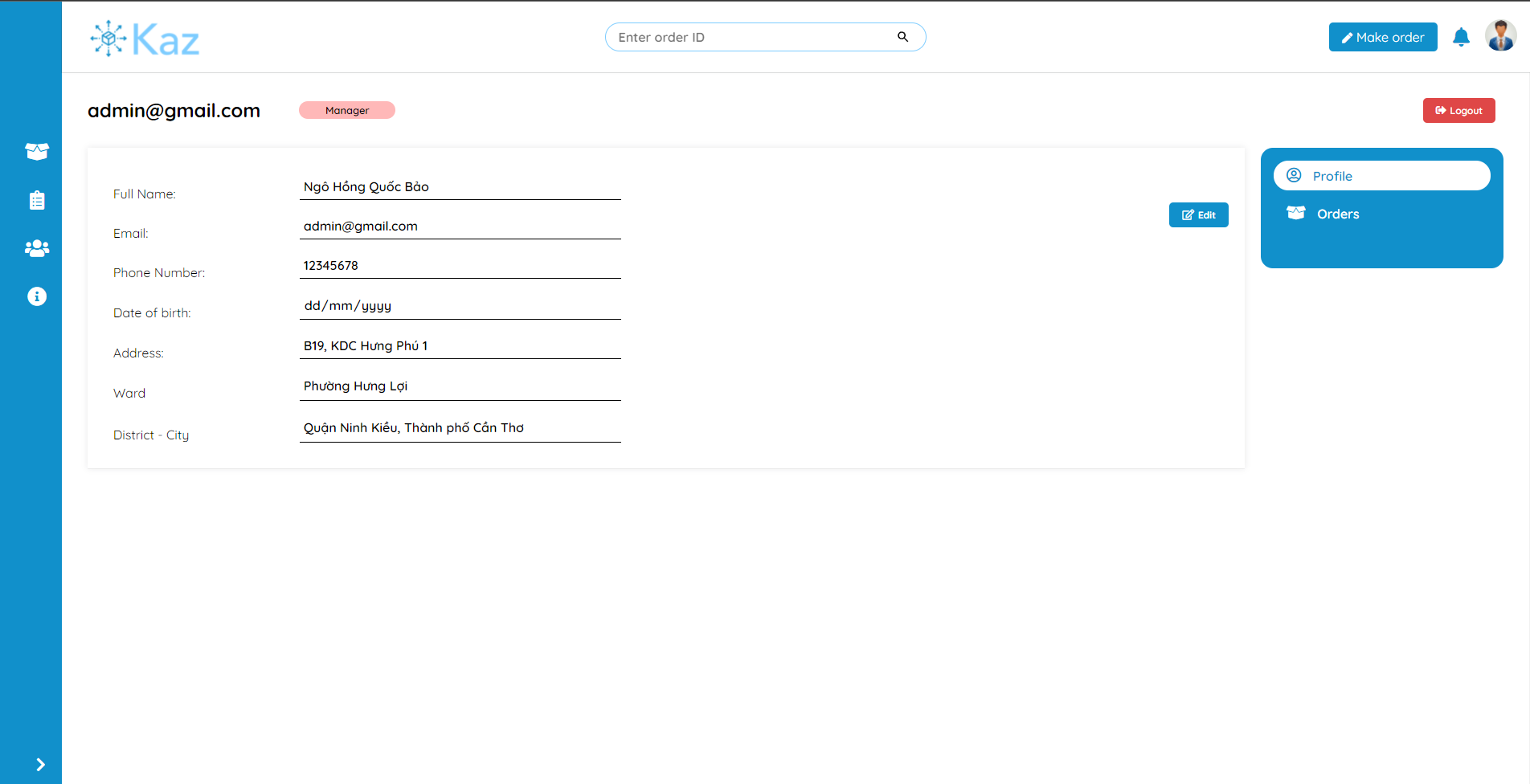


Figure : System Screenshot - Profile page

3.5. Pricing list management page

Pricing list page displays the prices correspond to each range of distance that a driver has travel to deliver an order.

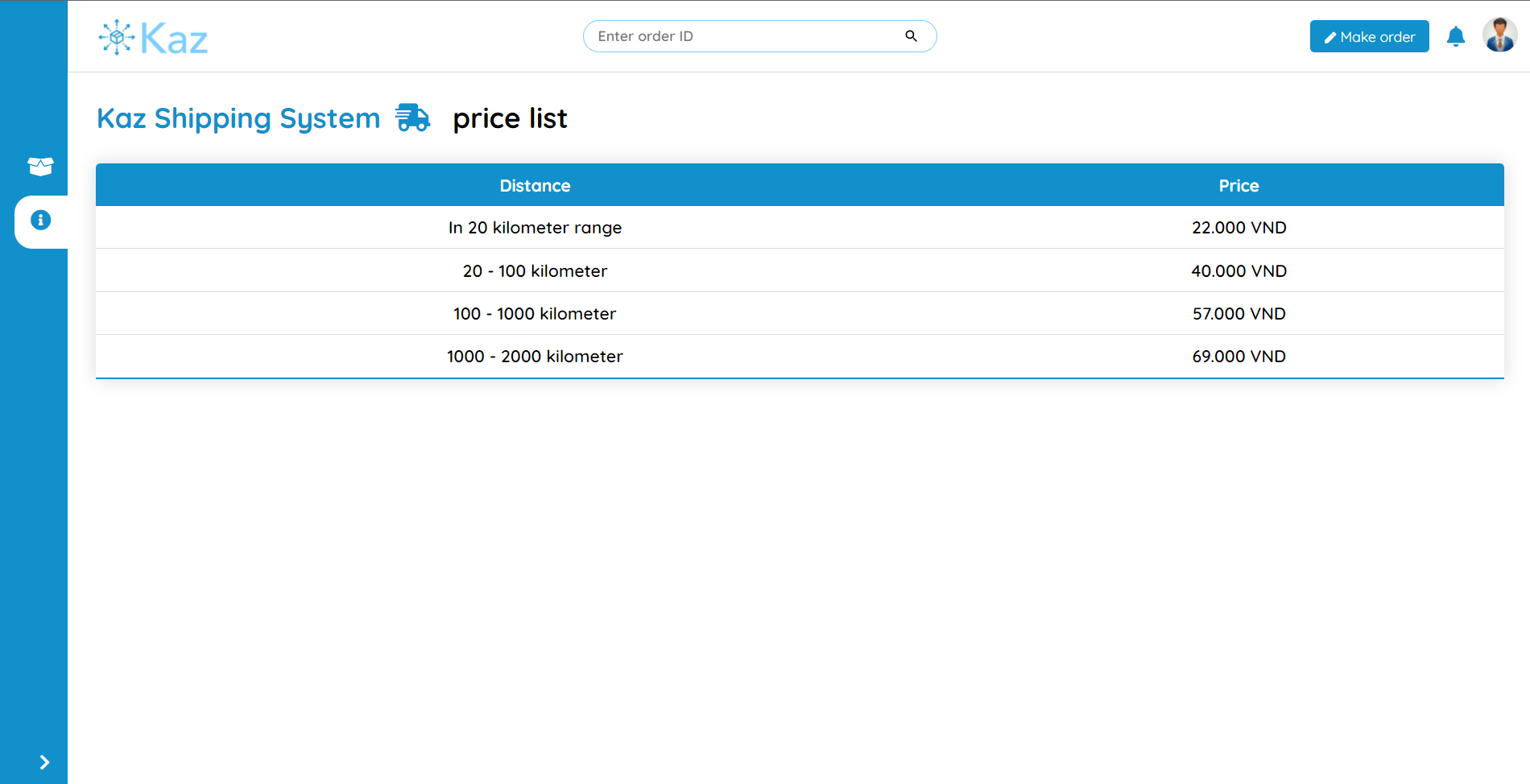


Figure : System Screenshot - Pricing list page on user view

For a manager account, he/she can also edit, delete or create a new price tag.

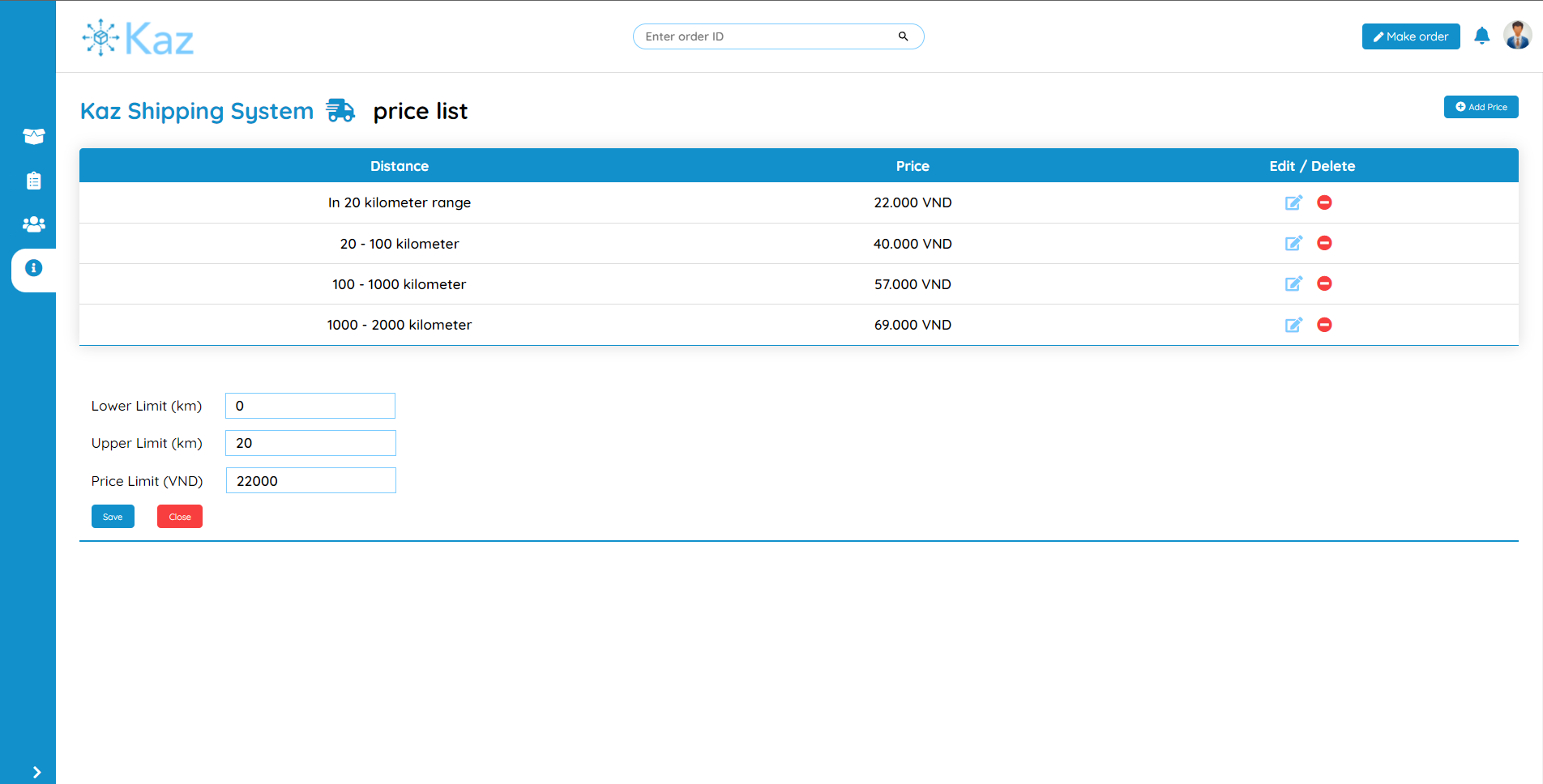


Figure : System Screenshot - Pricing list page on manager view

3.6. Reports page

Report page displays information about all the orders which are placed by customers in different periods of time (today, last 7 days, last month and all time) and also the calculated revenue from them.

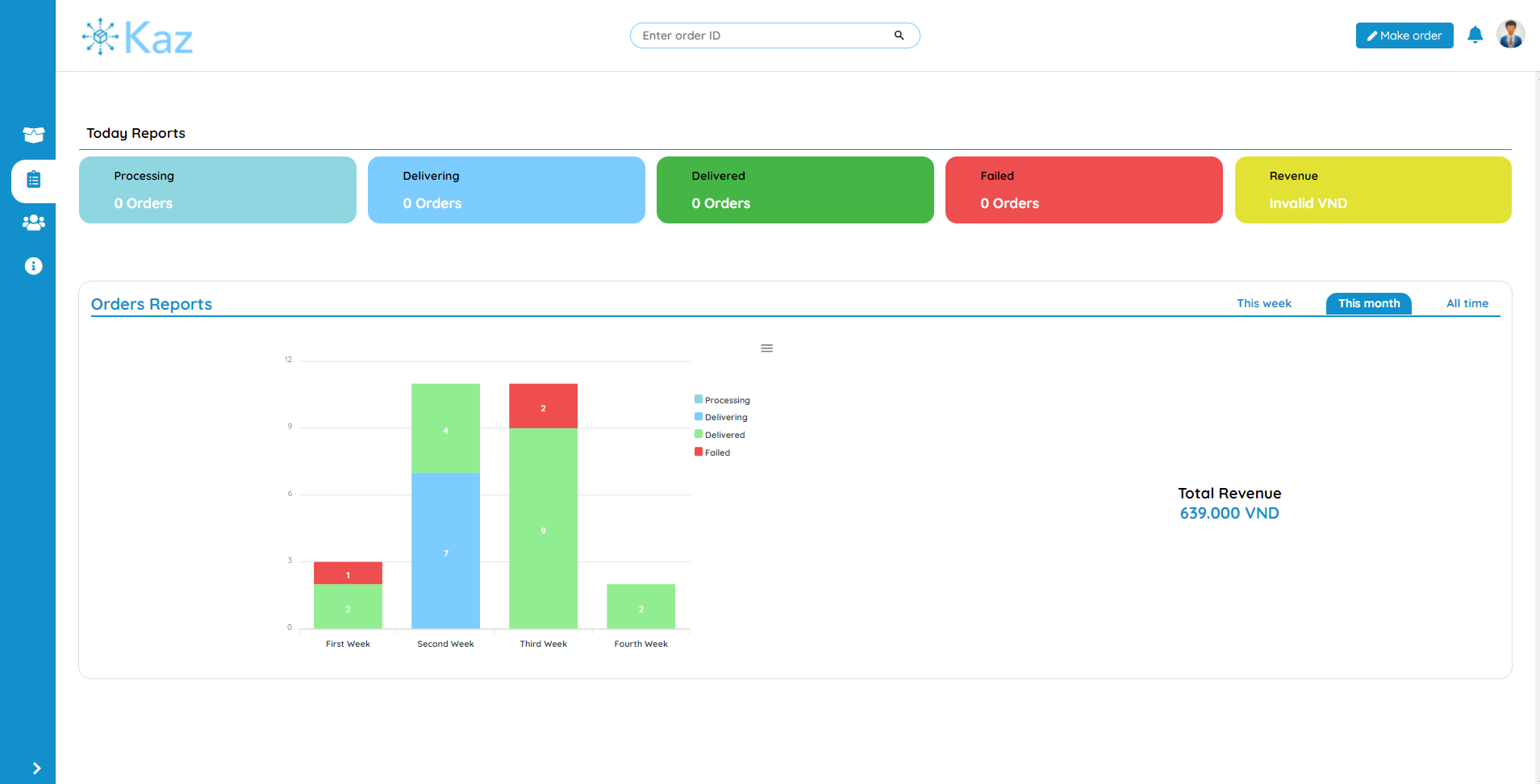


Figure : System Screenshot - Report page

3.7. Order Detail page

Order detail page displays information of an order. This information includes consignor and consignee’s information, the status, note, payment method and the date created of the order, and the package information from that order.

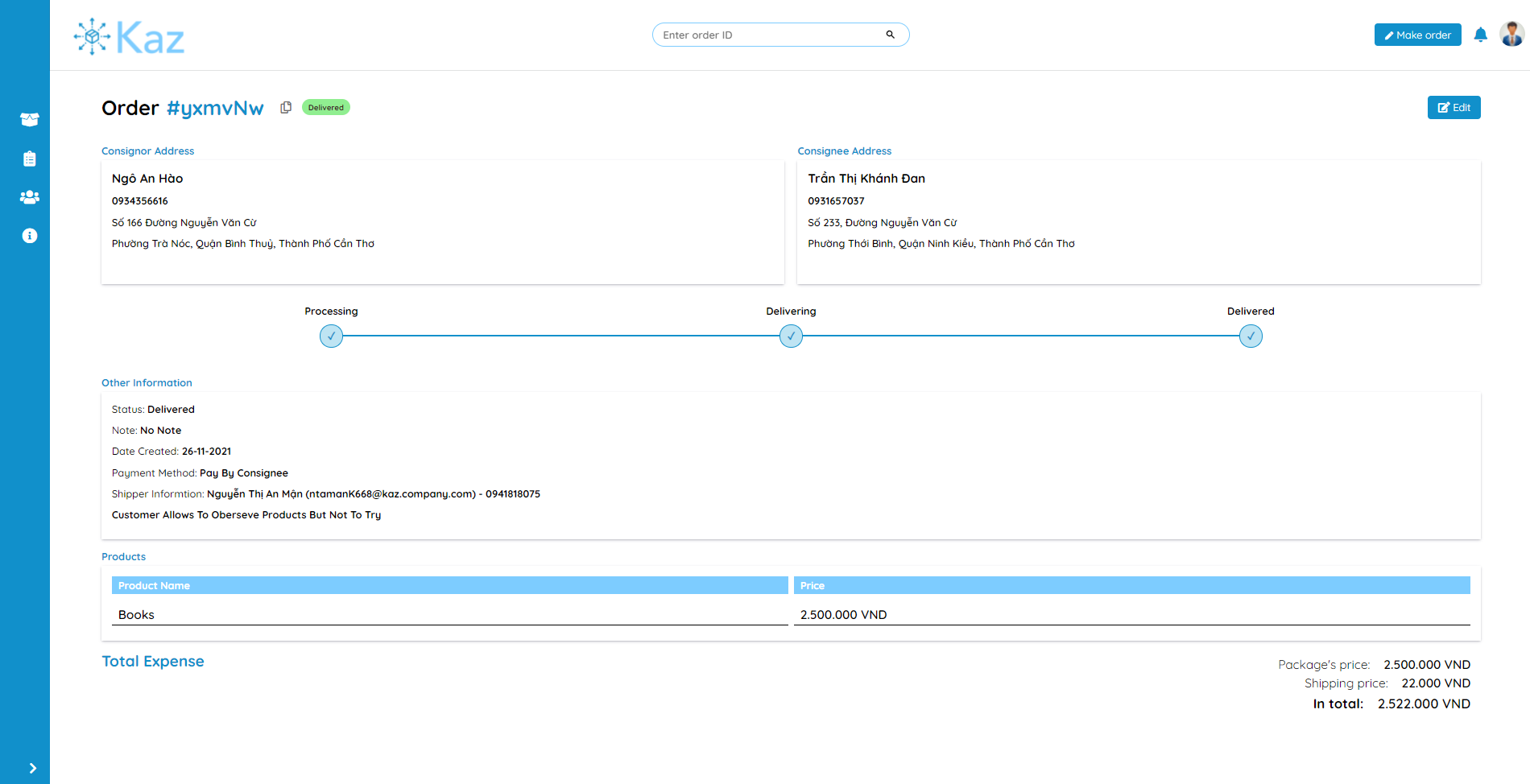


Figure : System Screenshot - Order detail page

For an order that is delivering, there is also a map displays on the page for user to easily track the current location of that order. User can freely manipulate the map (zoom, move around, …). There is also a “Focus on your order” button to focus the map to zoom in to the order location.

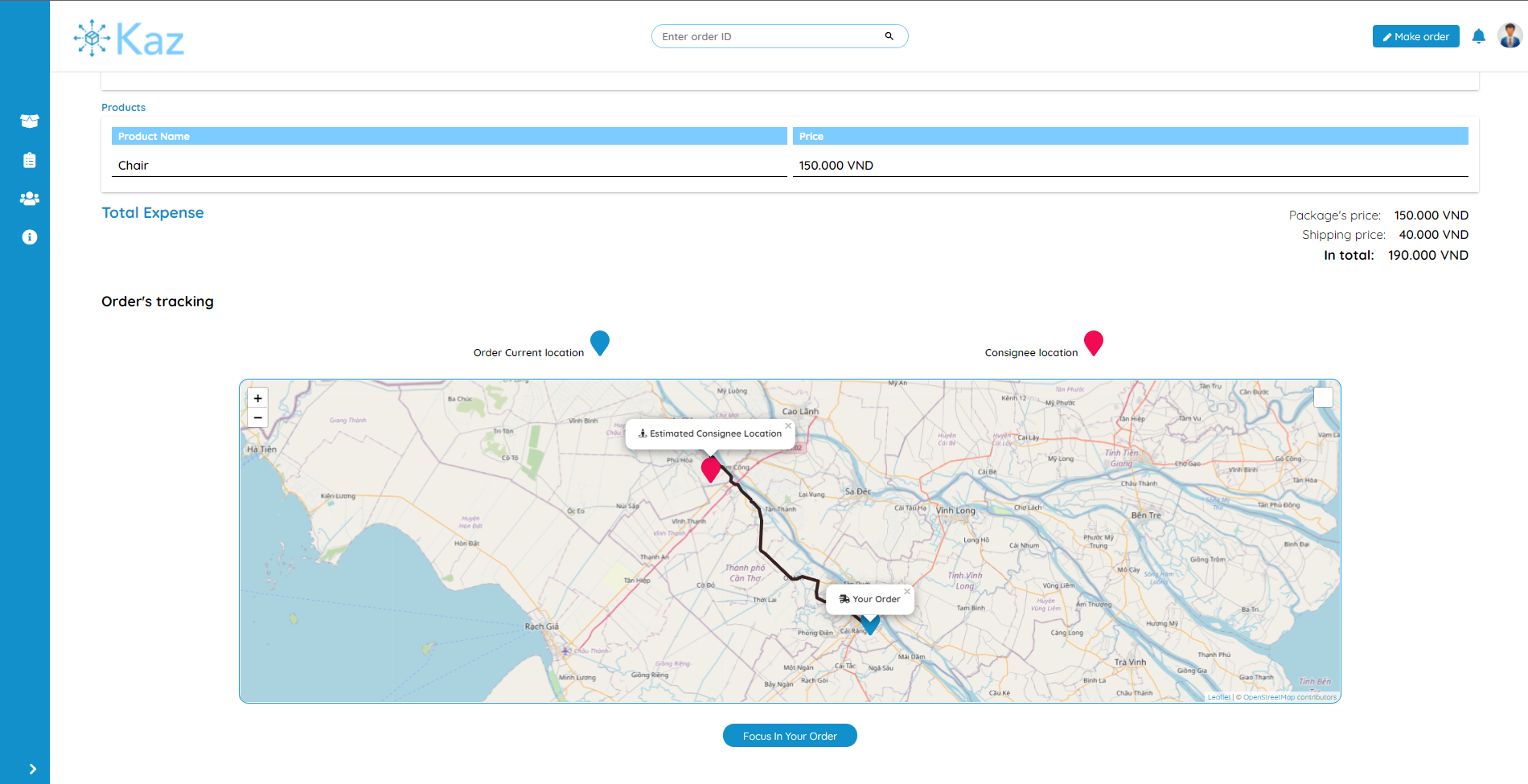


Figure : System Screenshot - Tracking order in Order detail page

3.8. Order placement page

A page for customer to place an order. The fields that requires to place an order include consignor and consignee’s addresses, information of the package and other information regarding the delivering process such as the payment method, how the consignee can preview the product, or any note for the package.

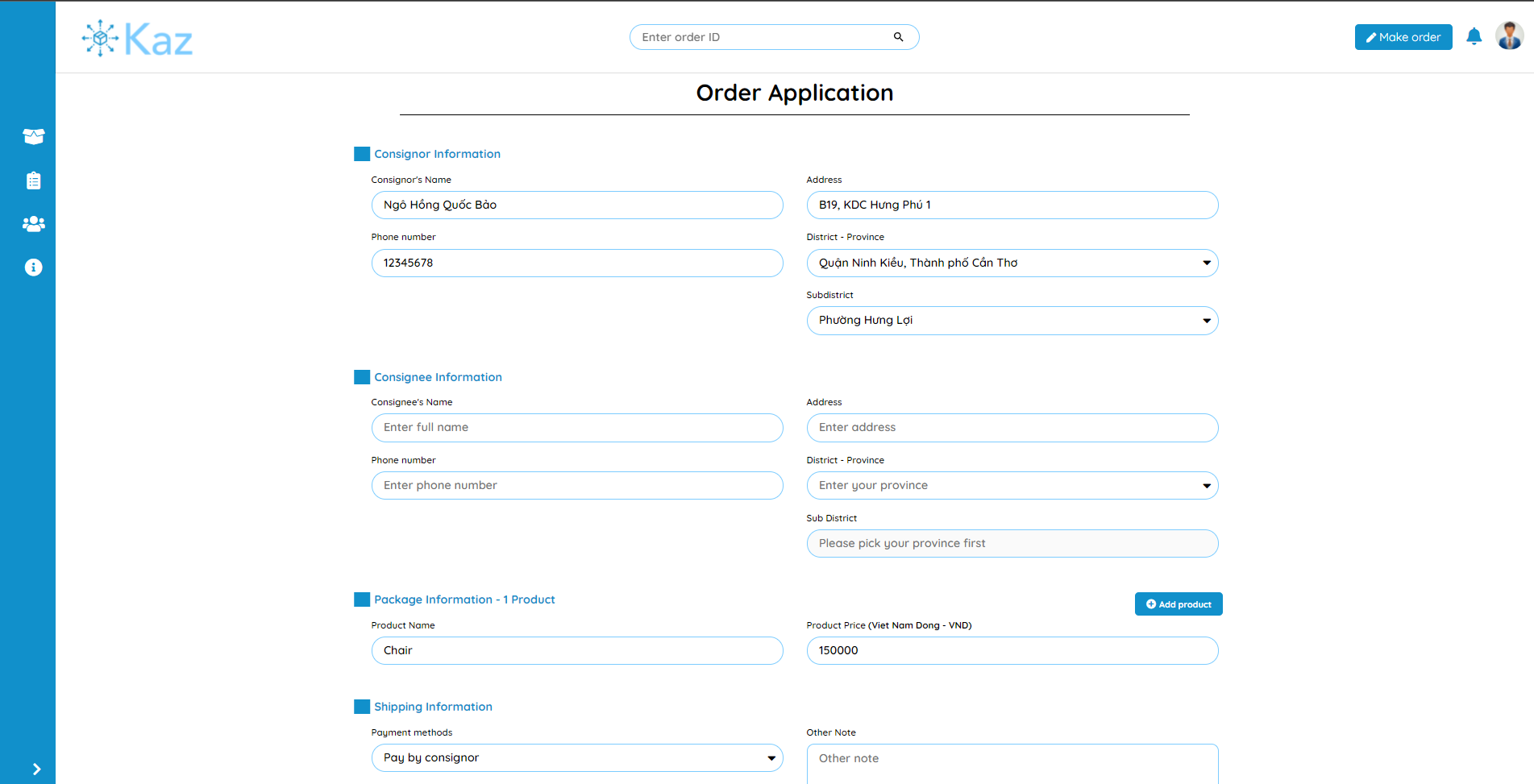


Figure : System Screenshot - Order placement page

When the all the fields are filled and valid, and the customer hits the “Submit” button. The system will make a couple of API calls:

* Post requests to Open Route Service API to retrieve the location of the consignee.
* Post requests to Leaflet Routing Machine API to retrieve the estimated distance and time to travel from the company location to the consignee location.
* Post requests to the backend API to retrieve the service price depends on the distance.

If there are any errors from any of these API calls, an error message will be display to inform the customer to try again later. Otherwise, a pop-up will appear to display the information from the API calls and re-confirm all the inputs from the customer, if everything is valid, customer can place an order by click the “Place order” button.

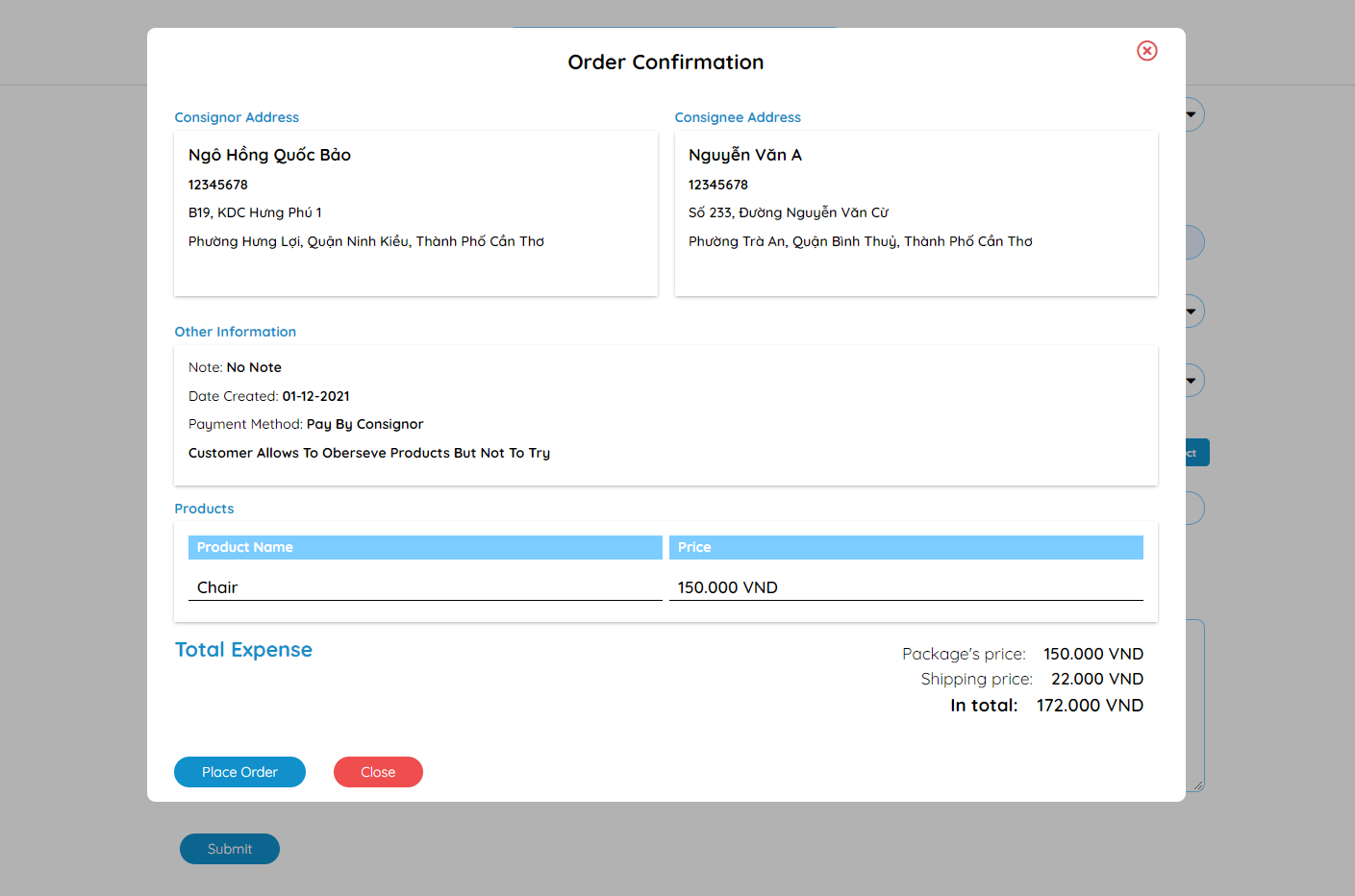


Figure : System Screenshot - Order confirmation

1. TESTING AND EXPERIMENTAL RESULTS

1. The purposes of testing

To certain that the final form of the system satisfies all the basis required features. Testing will be helpful to estimate how the system’s features work, the scope of the features, the progress of the building the features. Therefore, the testing results can be served for maintenance and development of the system.

2. Testing explanation

2.1. Details of the testing plan

For the testing plan, I will prioritize to display the test cases’ results from some of the system’s main features. Similar features will not be tested and displayed. I will also clearly present the criteria and variation of the test cases.

2.1.1. Testing features

The table below displays all the features that were tested with their expected results.

| **No** | **Feature** | **Overview** | **Cases** | **Expected Results** |
| --- | --- | --- | --- | --- |
|  | Login | Customer or Manager logins to the system | * Valid inputs * Wrong email * Wrong password | * Successfully login * Display error corresponds to the input |
|  | Register | Customer registers his/her email to the system | * Valid inputs * Invalid inputs (email format, insecure password) | * Successfully register * Display error corresponds to the input |
|  | Add/Edit pricing list | Manager adds or edits a price in the pricing list | * Valid inputs * Invalid inputs (empty fields, invalid price format) | * Successfully add/update price. * Display error corresponds to the input |
|  | Search order | Manager or customer search for an order by using its id | * Valid order id * Invalid order id (non-existence id or order not belongs to customer) | * Successfully view the detail of an order * Display error corresponds to the input |
|  | Find consignee location | Calculate the coordinate by using the Open Route Service API of the consignee location base on the input | * Calculate the location from the address | * Find the geolocation of the consignee’s address successfully * Display errors when no location is found. |
|  | Navigate and calculate distance | Calculate distance and routes from the consignee and the order’s location | * Distance from the company to the consignee’s location when the order is first placed. * Distance from the driver to the consignee when the order is being delivered. | * Successfully calculate distance and routes to display. * Display error when no routes are found. |

Table : Features require to be tested

2.1.2. The goal of testing

* Successful test case: Test case is considered success if the experimental results of that test case meet the expected results. If there is an error which are expected, the message for that error must be clear and explicit.
* Failed test case: Test case is considered failed if the experimental results are unexpected compared to the expected results or there are errors appear without mentioning before that affect negatively to how the system work.

2.1.3. Result from the test cases

* Plan for the testing.
* All the cases for the testing.
* Reports of the test results.

2.2. Test environment

Hardware:

* Personal computer with stable internet connection.
* CPU Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz; RAM 8.00 GB; SSD 256GB.

Software:

* Windows 10 operating system.
* Google Chrome version 96.0.4664.45 (official version) (64 bit).

2.3. Test cases

From the list of features that are displayed from in Table 8, I began to build the plan for testing each feature. Every test case will have the input, experimental results along with the evaluation. The plans for each feature are describe in the tables below.

2.3.1. Login

Test case ID: TC001

Conditions: Testing device is connected to internet and tester is required to have an account which is authenticated by the system.

Account for test case:

* Email: admin@gmail.com
* Password: kaz123

| **No** | **Plan** | **Inputs** | **Expected result** | **Experimental**  **result** | **Evaluation** |
| --- | --- | --- | --- | --- | --- |
| TC001-01 | No information filled | No input | Display error message that all information needs to be displayed | Expected | Success |
| TC001-02 | Invalid email or invalid password | Email is not registered in the system  Wrong password | Display “Invalid email or password” message | Expected | Success |
| TC001-03 | Login successfully | All inputs are invalid | Login successfully and redirect to the management page according to user’s role | Expected | Success |

Table : Feature test cases - Login

2.3.2. Register

Test case ID: TC002

Conditions: Testing device is connected to the Internet

| **No** | **Plan** | **Inputs** | **Expected result** | **Experimental**  **result** | **Evaluation** |
| --- | --- | --- | --- | --- | --- |
| TC002-01 | No information filled | No input | Display error message that all information needs to be displayed | Expected | Success |
| TC002-02 | Email is already taken | Email is already registered in the system  Any password | Display “Invalid email” message | Expected | Success |
| TC002-03 | Register successfully | All inputs are invalid | Register successfully and redirect to login page | Expected | Success |

Table : Feature test cases - Register

2.3.3. Add/Edit pricing list

Test case ID: TC003

Conditions: Tester is login to the system and tester account is an admin/manager account.

| **No** | **Plan** | **Inputs** | **Expected result** | **Experimental**  **result** | **Evaluation** |
| --- | --- | --- | --- | --- | --- |
| TC003-01 | No information filled | No input | Display error message that all information needs to be displayed | Expected | Success |
| TC003-02 | Price is not valid (under 10.000 VND) | Any upper and lower limit  Under 10.000 VND price | Display “Invalid price tag” message | Expected | Success |
| TC003-03 | Invalid lower limit and upper limit | Lower limit is bigger than upper limit  Any price that bigger than 10.000 VND | Display “Invalid lower and upper distance” message | Expected | Success |
| TC003-04 | Add/Edit successfully | Lower limit is smaller than upper limit  Any price that bigger than 10.000 VND | Add/Edit successfully | Expected | Success |

Table : Feature test cases - Add/Edit pricing list

2.3.4. Search for order by its id

Test case ID: TC004

Conditions: There is at least one order in the database.

Testing accounts:

* Manager account: admin@gmail.com
* Customer account: baobao@gmail.com

| **No** | **Plan** | **Inputs** | **Expected result** | **Experimental**  **result** | **Evaluation** |
| --- | --- | --- | --- | --- | --- |
| TC004-01 | Search order by its id without login to the system | Valid order id | Redirect to the Order Preview page. | Expected | Success |
| TC004-02 | Search order by its id with the manager account | Valid order id | Redirect to the Order Detail page. | Expected | Success |
| TC004-03 | Search order by its id with the customer account | Valid order id and the order is owned by the customer | Redirect to the Order Detail page. | Expected | Success |
| TC004-04 | Search un-related order by its id with the customer account | Valid order id but the order is not owned by the customer | Redirect to the Unauthenticated page. | Expected | Success |
| TC004-05 | Search order with invalid id (non-exist order) | Order id is not recorded in the database. | Redirect to the 404 page. | Expected | Success |

Table : Feature test cases - Order searching

2.3.5. Find consignee information

Test case ID: TC005

Conditions: Tester is login to the system.

Scenario: In this test, I will test the system by feeding the Open Route Service API a lot of valid addresses, which spread across all three part of Vietnam.

Expected results: The Open Route Service API return a geolocation (latitude and longitude) for each address.

Input format: The address format will only contain the ward name, the district name and the province/city name.

Example of an address: “Hưng Lợi, Cái Răng, Cần Thơ”

| **Number of cases** | **Successful case** | **Success rate** |
| --- | --- | --- |
| 50 | 37 | 74% |

Table : Feature test cases - Find consignee location

2.3.6. Navigate and calculate distance

Test case ID: TC005

Conditions: Tester is login to the system.

Scenario: In this test, the inputs are from the cases which the system had successfully get the geolocations (of the consignee addresses) from test TC005.

Expected results: The Leaflet Routing Machine API returns the path and the distance from the company location to the consignee location.

Input format: Geolocation of the company location and geolocation of the consignee address.

| **Number of cases** | **Successful case** | **Success rate** |
| --- | --- | --- |
| 37 | 33 | 89.19% |

Table : Feature test cases - Navigate and calculate distance

CONCLUSION

1. Final results

Throughout the time of research and producing this thesis with the “Delivery management system” topic, I had an opportunity to expose myself with a lot of web development technologies and services. I am also able to build a full functional delivery system that matches all the basis criteria. My system can help the manager to manage the employees, the pricing list and the orders, track their locations and the customer can place an order, track the order. The system’s user interface is built with user experiment in mind and met the goal of being user friendly, easy to use and comprehensible.

2. Future works

In this topic, due to limited time and resources I mainly focused in developing and finalizing the basis and most important features of a delivery system. In other hand, I also prepare some future paths to improve the system in the future.

* Using VISA cards to use the Google Map API for better finding and routing between locations.
* Implementing online paying methods such as PayPal, MoMo for the consignor to pay the delivery fee right after he/she confirms the order placement.
* Researching and implementing the “Return goods” policy for customers to return the products that they are not happy with.
* Developing a fully functional mobile app for company’s drivers to update their location and route their way to the consignee location.

REFERENCES

1. Giao hàng tiết kiệm, “Dịch vụ Giao hàng tiết kiệm” from <https://giaohangtietkiem.vn/dich-vu-giao-hang-ghtk>.
2. Giao hàng nhanh, “Dịch vụ giao hàng nhanh” from <https://ghn.vn/pages/dich-vu-giao-hang>.
3. Andrew Smith, “Building Management System (BMS)”, *City of Melbourne*.
4. MDN Web Docs October 8th 2021, accessed November 1st 2021, <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>.
5. Rohan Vats, “Introduction To Django REST Framework: Development with Examples”, *upGrad*, 10/05/2021.
6. Vuejs.org, “Vue.js Introduction” from <https://vuejs.org/v2/guide/>.
7. Wikipedia December 5th 2021, accessed October 20th 2021, <https://en.wikipedia.org/wiki/Leaflet_(software)>.
8. Per Liedman, Leaflet Routing Machine 2015, accessed October 20th 2021, <http://www.liedman.net/leaflet-routing-machine/>.
9. Open Route Service, accessed October 20th 2021, <https://openrouteservice.org/>.