HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

GRADUATION THESIS

Develop an efficient Delivery System

Major: Computer science

COUNCIL: COMPUTER SCIENCE
SUPERVISOR: PhD. PHAN TRONG NHAN
REVIEWER: Assoc. Prof. NGUYEN THANH BINH

—o0o—

STU: NGUYEN NGOC GIA VAN - 1614058

HO CHI MINH CITY, 12/2021

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

GRADUATION THESIS

Develop an efficient Delivery System

Major: Computer science

COUNCIL: COMPUTER SCIENCE
SUPERVISOR: PhD. PHAN TRONG NHAN
REVIEWER: Assoc. Prof. NGUYEN THANH BINH

—o0o—

STU: NGUYEN NGOC GIA VAN - 1614058

HO CHI MINH CITY, 12/2021

ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

KHOA:KH & KT Máy tính____
BỘ MÔN:HTTT _____

NHIỆM VỤ LUẬN ÁN TỐT NGHIỆP
Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: Nguyễn Ngọc Gia Văn_____ MSSV: 1614058 _____
NGÀNH: Khoa học Máy tính_____ LỚP:_____

1. Đầu đề luận án:
Develop an efficient delivery system _____

2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):
✔ Survey and analyze related systems.
✔ Propose features that make the delivery system more efficient.
✔ Identify main academic problems to be solved.
✔ Research relevant theory and practical approaches for those features implementation as well as for those issues.
✔ Research in-need technologies.
✔ Design the system with the proposed features and solutions.
✔ Develop a system prototype.
✔ Implement and perfect the system.
✔ Conduct empirical experiments.

3. Ngày giao nhiệm vụ luận án: 23/08/2021

4. Ngày hoàn thành nhiệm vụ: 13/12/2021

5. Họ tên giảng viên hướng dẫn: **Phần hướng dẫn:**

1) TS. Phan Trọng Nhân_____

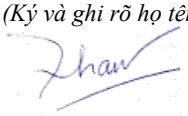
Nội dung và yêu cầu LVTN đã được thông qua Bộ môn.

Ngày 23. tháng 08. năm 2021

CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)

PGS. TS. Trần Minh Quang

GIẢNG VIÊN HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)



TS. Phan Trọng Nhân

PHẦN DÀNH CHO KHOA, BỘ MÔN:
Người duyệt (chấm sơ bộ):_____
Đơn vị: _____
Ngày bảo vệ: _____
Điểm tổng kết: _____
Nơi lưu trữ luận án: _____

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 24 tháng 12 năm 2021

PHIẾU CHẤM BẢO VỆ LVTN

(Dành cho người hướng dẫn)

1. Họ và tên SV: Nguyễn Ngọc Gia Văn
   MSSV: 1614058 Ngành (chuyên ngành): Khoa học Máy tính
2. Đề tài: Develop an efficient delivery system
3. Họ tên người hướng dẫn: TS. Phan Trọng Nhân
4. Tổng quát về bản thuyết minh:
   Số trang: Số chương:
   Số bảng số liệu Số hình vẽ:
   Số tài liệu tham khảo: Phần mềm tính toán:
   Hiện vật (sản phẩm)
5. Tổng quát về các bản vẽ:
   - Số bản vẽ: Bản A1: Bản A2: Khổ khác:
   - Số bản vẽ vẽ tay Số bản vẽ trên máy tính:
6. Những ưu điểm chính của LVTN:

- The student developed a system that covers the basic features of a delivery system and supports the efficiency of its operations for delivery. More specifically, the thesis work solves the capcacity-aware vehicle routing problem with auto and manual modes based on Google OR-tools.
- The student analyzed problem statement as well as its related work and challenges. In addition, the complete flow of delivery has been handled with several input constraints before the solver engine starts running.
- Moreover, the student applied modern technologies such as VueJS, MongoDB, and python to implement the system and integrated it with Telegram bot API and Google Map API. Furthermore, the system was successfully deployed on Heroku platform.
- Students were very proactive and open to novel knowledge and technology.

7. Những thiếu sót chính của LVTN:

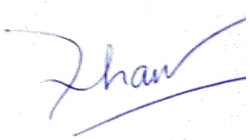
- The evaluation of system efficiency is still limited.
- Some other constraints about the order collecting phase (e.g., what if the promise time of some orders is close to one another) need thorough analysis and management.

8. Đề nghị: Được bảo vệ ☑ Bổ sung thêm để bảo vệ ☐ Không được bảo vệ ☐
9. 3 câu hỏi SV phải trả lời trước Hội đồng:

a.
b.
c.

10. Đánh giá chung (bằng chữ: giỏi, khá, TB): Very good Điểm : 8.9/10

Ký tên (ghi rõ họ tên)



Phan Trọng Nhân

Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom – Happiness

REVIEW OF GRADUATION THESIS

HCMC, 28th December 2021

1. Students name:

Nguyễn Ngọc Gia Văn (1614058)

2. Major: Computer Science

3. Thesis title:

4. Reviewer: Assoc.Prof. Nguyen Thanh Binh, PhD

5. Contents

- In this thesis, the author built a website in which customers can order food products online. This website will also contain a delivery system for owners to get notifications and the system will collect the orders automatically and optimize the route depending on the owner's requirement. The tasks in this thesis:
  + Technologies to build websites.
  + Learning google map api (application programing interface) service.
  + Learning vehicle routing problems (VRP) and its variants.
  + Learning constraints of vehicle routing problems and technology to solve them.
- The authors should explain more about the system which the authors built. The system has disadvantages for users.
- Some spelling errors in the essay.
- The author should clarify the advantages of the system which was built compared to existing systems.

6. General assessment: Marks 8.5/10

Reviewer

Nguyen Thanh Binh

INSURANCE

I hereby declare that the subject is my own personal and take full responsibility for the honesty of my project. All the materials I have used in my thesis all have their name, source and presented in ”REFERENCES”.

ACKNOWLEDGEMENTS

First, I would like to thank my instructors, Dr Phan Trong Nhan, for wholeheartly guiding me in this thesis proposal. Thanks to Mr. Nhan's suggestions and commenting for me, i am able to eliminate all unnecessary tasks and unusable ideas that slowing us down. His replying and commenting each time I commit my work really helps me to improve gradually and have a more overall view about the thesis. Next, I would like to send my gratitude to the teachers, professors and staffs in Bach Khoa university, especially the ones in Computer Science and Computer Engineering department for teaching new knowledge to me and supporting me through my time in the university. Thanks to the knowledge which you have taught me that I were able to finish this thesis. I also would like to thank my parents, my friends who not only supported us physically through the time of this thesis but also mentally. Thank you for being there when I needed. Finally, I wish everyone the best. The time I have had here at Bach Khoa university will always remain one of the most beautiful memories in my heart. Once again, I would like to thanks everyone for your support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The demand

COVID-19 first appeared in Wuhan city of Hubei Province in China in December 2019. It has a substantial impact on human life all around the world, especially for citizens. The threat of COVID-19 has resulted in people shopping online to get fresh food and reduce outdoor trips. The results suggest that more citizens will buy fresh food online increased. The experience of online shopping for fresh food during the lock-down days has promoted more online shopping.

Fresh food is important for people's daily lives, such as vegetables, fruit, meat, fish, milk, etc., which are consumed every day. Therefore, the shopping frequency for fresh food is much higher than other goods. With the improvement of the living standard, people have higher requirements for the quality of fresh food.

In the new post-COVID world, that demand for food convenience has increased – both by necessity and because so many brands are jumping on the fresh food delivery service bandwagon. To make this service available to customers, fresh food supply chains are either relying on third-party delivery providers or creating their own with an online ordering system.

Since the demand for fresh food convenience has increase, lead to the number of orders is large, the delivery location changes every day along with different constraint require- ments placed on the delivery process, so it takes a lot of time to arrange the orders every day, as well as difficult ensure that vehicles are used properly, delivery routes are short... causing in high transportation cost.

1.2 Why I choose this topic

This is the reason why the topic "develop an efficient delivery system" was created with the desire to apply the knowledge to solve vehicle routing problem to contribute to reducing transportation costs. In addition, better coordination of vehicle routing problem can affect positively on ecological aspects by reducing pollution, which is also one of important issues that need attention today.

This topic will give multiple techniques that not only build up on the web for ordering fresh food online but other fields like develop an efficient delivery system for food supply chains.

1.3 Objective and Content

1.3.1 Objectives

In this project, I wants to build a fully functional website in which customers can order food products online. This website will also contain an delivery system for owner get notifications and system will collect the orders automatically and optimize route depends on the owner's requirement such as the number of vehicles or the capacity of orders.

- **E-commerce Website**
  The website will have an online food store system that allows users to order and buy products pay directly or through momo e-wallet.
- **Admin Dashboard**
  The dashboard admin page will support staff to check orders placed, add new products or edit product information as well as track information and delivery vehicle status. And some other settings that assist employees in the management are as convenient as possible.
- **Optimize System**
  An optimize system that supports efficient automatic distribution and consolidation of orders.

  It also integrates a route optimization algorithm with given constraints.

  Moreover, it includes a bot that guides delivery drivers.

In order to achieve the objectives of this project, here are the tasks I need to accomplish before working on the development:

- Research the distributed database with various systems.
- Research technologies to build our website.
- Research business logic to build our website.
- Research and learn google map api (application programing interface) service.
- Research and learn vehicle routing problems (VRP) and it variants.
- Research and learn bot api (application programing interface).
- Research technology related to our project (extended library or framework supproted the system).
- Research and learn constrains of vehicle routing problems and technology to solve them.
- Research related problems and find solutions for them.

1.4 Boundary of the project

In this project, I focuses on recognising English and responding to them also by English.

The project focuses on researching the distribution network within the inner city of Ho Chi Minh City.

1.5 Structure of the thesis

- **Chapter 1: Introduction**
  In this chapter, I introduce the project with its goals and contents.
- **Chapter 2: Related work**
  This chapter show what I have prepared and researched before making this project
- **Chapter 3: System analysis**
  I analyse the requirement for this project and from there I design the use-cases and activity diagrams
- **Chapter 4: System design**
  I design the architecture of each component, the database system and UI for this project based on the analysis that I have made
- **Chapter 5: Implementation**
  The content of this chapter is how I implement the project into a website based on my system design
- **Chapter 6: Deployment and evaluation**
  This chapter contains how I deploy and test the result of our development
- **Chapter 7: Conclusion**
  This is the conclusion on what I have accomplished on the project, its result and future development plan.

Chapter 2

Related work

2.1 Grocery stores online ordering statistics

Online ordering is now the primary source of revenue for many glocery stores. For your business to succeed with takeout and online ordering, it's important to understand the shift in guest behaviors and how to translate your in-house hospitality to delivery and takeout sales.

2.1.1 General statistics and statistics in the US

a. Consumer behavior[1]

- 60% of U.S. consumers order delivery or takeout once a week.
- 31% say they use these third-party delivery services at least twice a week.
- 34% of consumers spend at least 50$ per order when ordering food online.
- 20% of consumers say they spend more on off-premise orders compared to a regular dine-in experience.
- Digital ordering and delivery have grown 300% faster than dine-in traffic since 2014.
- 87% of Americans who use third-party food delivery services agree that it makes their lives easier.
- 45% of consumers say that offering mobile ordering or loyalty programs would encourage them to use online ordering services more often.
- 63% of consumers agree that it is more convenient to get delivery than dining out with a family.

b. Impact of Covid-19[2]

- During the pandemic, 79% of consumers have bought groceries online, a jump from 19% a year ago.

[1]Stephanie Resendes. (2020, November 12). *Online Ordering Statistics Every Restaurateur Should Know in 2021.* Retrieved January 3, 2021, from https://upserve.com/restaurant-insider/online-ordering-statistics/

[2]Blake Morgan. (2020, Oct 19). *Statistics Showing The Lasting Impact Of COVID-19 On Consumers.* Retrieved January 3, 2021, from https://www.forbes.com/sites/blakemorgan/2020/10/19/50-statistics-showing-the-lasting-impact-of-covid-19-on-consumers/?sh=5f93c02e261f

- U.S. online grocery sales increased from $1.2 billion in August 2019 to $7.2 billion in June 2020.
- The average U.S. household monthly grocery bill was $525 in March 2020, an increase of 30% from March 2019.
- 10% of Americans now shop for groceries at least three times a week, down from 19% pre-COVID-19.
- 78% of consumers have made a change in where they shop for food.
- 15% of consumers say they now avoid the grocery stores where they typically shop.
- 36% of consumers have made a change in who shops for food because of COVID-19.

2.1.2 Statistics in Vietnam

a. Consumer behavior

- As surveyed in Vietnam in 2020, 56 percent of Vietnamese respondents stated that they preferred to purchase packaged and fresh food at traditional trade channels. Meanwhile, 60 percent of them indicated their preference for modern trade channels for canned food purchases[3]
- Revenue from online food delivery in Vietnam stood at 302 million U.S. dollars in 2020. By 2024, revenue in this segment is estimated to reach 557 million U.S. dollars.[4]
- The penetration rate of platform to consumer online food delivery in Vietnam stood at 2.3 percent in 2020.By 2024, the penetration rate in this segment is estimated to reach 4.6 percent of the population. [5]

b. Impact of Covid-19

- As of the first quarter of 2021, the average monthly traffic of online grocery retailers reached 2.47 million visits in Vietnam. Compared to the previous quarter, the traffic to these retailers increased by 13 percent. Online grocers have increased in popularity in Vietnam due to the ongoing COVID-19 pandemic and its related lockdowns.[6]

[3]Minh Ngoc Nguyen. (2021, Mar 19). *Preference of retail trade channels for basic necessities among consumers in Vietnam in 2020, by category.* Retrieved November 5, 2021, from https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer

[4]Statista Research Department. (2021, Jul 5). *Preference of retail trade channels for basic necessities among consumers in Vietnam in 2020, by category.* Retrieved November 5, 2021, from https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer

[5]Statista Research Department. (2021, Jul 5). *Penetration rate of online food delivery platforms in Vietnam from 2017 to 2024.* Retrieved November 5, 2021, from https://www.statista.com/forecasts/1230477/penetration-rate-online-food-platform-delivery-vietnam

[6]Minh Ngoc Nguyen (2021, Sep 22). *Average monthly web traffic of online grocery retailers in Vietnam from 4th quarter 2020 to 1st quarter 2021.* Retrieved November 5, 2021, from https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer

- As surveyed in Vietnam in 2020, 20 percent of Vietnamese respondents stated that they were much more willing to purchase basic necessities and fresh products online because of the COVID-19 pandemic. A further 38 percent of the respondents claimed to be somewhat more willing to do so. Meanwhile, according to 34 percent of the surveyed consumers, there was no change in their willingness to shop for these products online during the pandemic.[7]

- Vietnam's e-commerce market is projected to grow to VND 399.5 trillion ($17.3 billion) in 2023 from VND 218.3 trillion ($9.4 billion) last year, according to U.K. consulting firm GlobalData. [8]

2.1.3 Similar system surveillance[9]

Before starting on this project, I have conducted a survey of similar applications and websites in order to learn about their pros and cons and also set a benchmark for my project to improve upon.

When it comes to grocery shopping in Ho Chi Minh City, if you're not one of those who can patiently wait in a long queue or win a fight over toilet paper and hand sanitizer, online grocery stores are for you.

Check out these online grocery stores and supermarkets available in town that will make your food purchasing experience from the comfort of your own home faster and easier.

a. coopmart.vn

It's not just fresh produce and packaged foods you can get here – Coopmart's website also sells basic necessities such as homewares, cosmetics, and beverages. However, items here sell out fast so you might want to wake up early to fill your cart.

Your ordered items will be delivered within 24 hours, and you can make payment upon delivery or with a credit or ATM card online. If you have a family member who's not well-versed in shopping online, they can phone Coopmart directly to order.

b. lottemart.com.vn

One of the largest supermarket chains in Vietnam, LOTTE Mart has now made grocery shopping more convenient with the Speed LOTTE App. Here, you can choose from a wide variety of meat, seafood, vegetables, and packaged foods. It also has an abundant selection of powdered milk, beverages, and cosmetics imported from Korea.

LOTTE Mart makes deliveries within 15KM from any of its stores, so you know you'll get the goods fast. Shipping is free as long as your total purchase is over VND 150,000 (USD 6.45). Any purchases under VND 150,000 will be charged VND 5,000 (USD 0.21)

[7]Minh Ngoc Nguyen. (2021, Mar 19). *Willingness to buy necessities and fresh products online during COVID-19 Vietnam 2020.* Retrieved November 5, 2021, from https://www.statista.com/statistics/1222566/vietnam-willingness-to-buy-necessities-and-fresh-products-online-during-covid-19/

[8]Vien Thong (2020, May 14). *Has Covid-19 subverted the traditional Vietnamese consumer?.* Retrieved November 5, 2021, from https://e.vnexpress.net/news/business/industries/has-covid-19-subverted-the-traditional-vietnamese-consumer-4098697.html

[9]Josee Ng (2020, Mar 20). *Online Grocery Stores & Supermarkets In HCMC With Everything From Fresh Produce To Canned Goods.* Retrieved Aug 20, 2021, from https://thesmartlocal.com/vietnam/online-groceries-hcmc/

per kilometer for shipping. But with LOTTE Mart's gamut of offerings, it's not that hard to hit the VND 150,000 mark if you plan to buy basic necessities for a whole week.

After confirming your order on the app, your order will be delivered to your door within 3 hours. You can make payment online or upon delivery.

c. bachhoaxanh.com

Bach Hoa Xanh carries over 600 online products that are restocked daily, including meat, cooking ingredients, canned foods, beverages, and homeware. The only item it doesn't have for now are fresh vegetables.

Shipping is only VND 15,000 (USD 0.65) for any purchase, no matter the total price and weight. You can specify what time you want your order delivered, and in addition, you'll be entitled to a VND 100,000 compensation fee (USD 4.30) if your order arrives late.

All these websites/applications all have their own advantages and disadvantages. In general, their ease of access and the large amount of products are the main advantages. However, when I actually surveyed the employees of the above stores about the shipping system, the order collection process is still manual, Delivery route depends on driver experience, there is no optimal routing system. In summary, about user interface, the interaction between the website and the user is quite efficient. However, the cost for staff is still high because of manual work, besides, the transportation cost if not using the route optimization algorithm is also high.

2.2 Technologies

2.2.1 WebSocket

a. Definition

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol enables interaction between a web browser (or other client application) and a web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server.[10]

b. Boundary

WebSocket protocol is supported by most of the web browsers including Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari and Opera. WebSocket is usually implemented in web servers and browsers to handle WebSocket messages.

- Github and StackOverflow[11] are two primary example of WebSocket implementation.
- Socket.IO is a library which support WebSocket for NodeJs

[10] Wikipedia. *WebSocket*. Retrieved Oct 27, 2020 from https://en.wikipedia.org/wiki/WebSocket

[11] Nick Craver. (2016 Feb 17). *Stack Overflow: The Architecture - 2016 Edition*. Retrieved Oct 27, 2020 from https://nickcraver.com/blog/2016/02/17/stack-overflow-the-architecture-2016-edition/

- Nginx has supported WebSockets since 2013
- Apache HTTP Server has supported WebSockets since July, 2013
- Internet Information Services added support for WebSockets in version 8 which was released with Windows Server 2012

c. Limitation

WebSockets don't automatically recover when connections are terminated – this is something you need to implement yourself, and is part of the reason why there are many client-side libraries in existence.

2.2.2 ExpressJS Framework

a. Environment

For developing the back end environment, I research 2 different development environments based on my own experience and how handy they are in managing systems like this project which are Python and NodeJS. I have compared and analysed their properties, pros and cons with the following tables

So after analysing both the environment and their pros/cons, I decide to choose Node.js as its performance is fast and it is more suitable than Python. Also I have much more experience in developing using Node.js and third party libraries are actually very good at supporting our website (socket.io, ...).

	Python	Node.js
Definition	Python is a multi-paradigm, general-purpose, interpreted, high-level programming language	Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side
Usage	Python can be used in web programming, desktop application, gaming	Server-side scripting
Benefits	Python enables developers to get the job done with fewer lines of code than similar object-oriented languages. Further, most developers are also comfortable with switching between Java and Python.	Node.js is a server-side JavaScript environment. It uses an asynchronous event-driven model and is designed for writing scalable Internet applications, notably web servers. Thus, Node.js gets excellent performance based on the architectures of many Internet applications.
Real-time Usage	Python is used in Real-time data processing. PyRealtime package simplified to the development of real-time applications.	Maintenance and Handling of large volumes of customer data
Scalability	Not easy scalable	It is highly scalable as Nodejs is microservices-based that best suits distributed systems.
Best suitable for	Data Science app voice & face recognition Image processing software 3D modeling software game	IoT Solutions Real-time chats & chatbots complex single-page apps Real-time collaboration apps and Streaming platforms

Table 2.1: Python and NodeJs comparison table

For their pros and cons: Python is an ideal platform to do multiple things – web applications, integration with back-end applications, numerical computations, machine learning, and network programming. Node.js is a better choice if your focus is on web applications and website development.

Python is suited for developing larger projects. Node.js is best suited for small projects to enable functionality that needs less amount of scripting.

Node.js has superb performance and speed, is a perfect solution for applications featuring real-time messaging or chatting, as well as for heavy-load applications, content management solutions, multi-vendor marketplaces, e-commerce, and other applications largely depending on the speed of processing.

b. ExpressJS Framework

For back-end development, I have researched frameworks in NodeJs that are popular, as they provide more management. After research, I find that there are multiple frameworks in NodeJs that can help develop the back-end of my website. So based on our knowledge, testing and reviewing, I minimize my options to 3 frameworks: Express.Js, Nest.Js and Feathers.JS

1. **Express.Js**: a web application framework for Node.js. It provides various features

that make web application development fast and easy which otherwise takes more time using only Node.js. It's a set of routing libraries that provides a thin layer of fundamental web application features that add to the lovely existing Node.js features. It focuses on high performance and supports robust routing, and HTTP helpers (redirection, caching, etc). It comes with a view system supporting 14+ template engines, content negotiation, and an executable for generating applications quickly. Express.js is based on the Node.js middleware module called connect which in turn uses http module. So, any middleware which is based on connect will also work with Express.js.

Advantages:

- Easy to connect with databases such as MongoDB, Redis, MySQL
- Includes various middleware modules which you can use to perform additional tasks on request and response.
- Easy to serve static files and resources of your application.
- Flexible Plugins
- Simple and easy to set up and customization.
- It lets you identify your application routes based on HTTP methods and URLs.
- It helps you to identify a middleware handling error.

Disadvantages:

- Express doesn't dictate how security is implemented either - that's entirely up to your app.
- Its error messages are usually unhelpful.

2. **Nest.Js**: a framework for building efficient, scalable Node.js server-side applications. It uses progressive JavaScript, is built with and fully supports TypeScript (yet still enables developers to code in pure JavaScript)

   Advantages:

   - The framework is very annotation-driven, with everything from endpoints to Swagger documentation being generated from them. The endpoints are clean and simple, and the annotations make developing simpler all around.
   - The folder structure in Nest.js is heavily based on Angular. This allows for minimal downtime when first designing a Nest service.
   - Because Nest.js is a module-based framework, it's easyto externalize general-purpose modules and reuse code in multiple projects
   - Components get their own folders, with an application module and main file residing in the root. This simple structure allows more attention to be paid to the design of endpoints and their consumers, instead of application structure.
   - Nest.js uses the latest version of TypeScript, which helps ensure that it will remain relevant in the rapidly changing JavaScript landscape and gives developers less context switching. The transition from Angular code to Nest is relatively easy.

   Disadvantages:

- The largest risk facing Nest users is the lack of documentation. The framework has great integrations with other frameworks but the documentation is minimal and doesn't cover any issues that may arise.
- Nest does hold an edge in its use of TypeScript and relation to Angular, but it doesn't have the backing power of a large enterprise behind it.
- Overall, Nest has a smaller community compared to other frameworks.

3. **Next.js**: Next is the most popular framework compared to the other two. It has more npm weekly downloads, GitHub stars and number of contributors.

   Next.js is a React framework that lets you build server-side rendering and static web applications using React.

   Advantages:

   - Every component is server-rendered by default
   - Automatic code splitting for faster page loads
   - Webpack-based dev environment which supports Hot Module Replacement (HMR)
   - Fetching data is very simple
   - It's possible to customize with your own Babel and Webpack configurations

   Disadvantages:

   - Next is powerful, but If you're creating a simple app, it can be overkill
   - All data needs to be loadable from both the client and server
   - Migrating a server-side app to Next.js is not a quick process, and depending on your project it may be too much work

Overall, these frameworks all have their own advantages and limitations. Express is usually used for quick and easy websites while Next.js is a small JavaScript framework primarily used as a back-end for developing applications in React framework. NestJs is used for its ability to scale the back-end and ease of management. So in the end, I see that ExpressJs is the most suitable framework for building back-end in the long term.

The reason why I choose ExpressJS[12].

- Improves scalability of the application:

  I can easily scale up my application. With the presence of Node.JS, I can easily scale my application in every way by adding nodes and adding additional resources to it.

- Same language can be used for both backend and frontend:

  With the aid of JavaScript, I can do both frontend and backend coding. While there are many similar frameworks, they provide different frontend and backend language support, which makes it very difficult to deal with. But with Express. JS, I'll rarely encounter any such issues.

[12]Lucas White. (2020, Nov 6). *What Are The Reasons To Learn Express.Js In 2021?*. Retrieved Mar 1, 2021 from *https://codersera.com/blog/learn-express-js/*

- Express.JS supports caching feature:

  The caching function is provided by Express.js, and the benefit of it is that I will not have to re-execute the codes over and over again. In addition, it will allow the web site to load quicker than ever before.

- It requires less development and maintenance cost:

  Express. JS is a full-stack JavaScript that does not require me to employ numerous developers to manage a web application's frontend and backend. Thus, it allows me to cut a lot of costs on the development and maintenance of the app.

- It increase the efficiency and performance:

  It was noted previously that the JavaScript code is interpreted by Node.js through Google's V8 JavaScript engine. The JavaScript code is complied with by this engine straight into the machine code. This makes applying the code in an efficient way simpler and quicker.

  As it facilitates non-blocking I/O operations, the speed of code execution is also improved by the runtime environment.

- It has a support of large and active community:

  Node.js is privileged to have a wide and committed group of developers who continue to contribute to its ongoing growth and progress on a regular basis.

  In fact, the developer groups are well assisted by JavaScript developers who provide GitHub with ready-made and easy solutions and codes. The developers are expected to initiate several more developers in the future.

- Manages the requests concurrently:

  Since Node.js provides the choice of non-blocking I/O systems, it allows me to process multiple requests simultaneously.

  The framework can manage the efficient handling of concurrent requests better than others, like Ruby or Python. The incoming requests are positioned and are launched rapidly and systematically.

- Node.js is highly extensible:

  Node.js is considered to be highly extensible, which implies that Node.js can be modified and expanded according to its specifications.

  I may also use JSON to offer the scope for data exchange between the client and the webserver. Integrated APIs for the creation of HTTP, TCP, and DNS servers, etc., are also supported.

Installation

1. Install NodeJS environment
2. Create a directory to hold your application, and make that your working directory.

```
mkdir myapp
cd myapp
```

3. Use the npm init command to create a package.json file for your application

```
npm init
```

4. Install Express in the myapp directory and save it in the dependencies list

```
npm install express --save
```

2.2.3 VueJS Framework

For front-end developing, I research 3 popular developing frameworks and how well they perform against each other, provide a structure to help judge front-end JavaScript frameworks in general and how can I choose the best fit for my project. The three frameworks are: React, Vue, Angular. They are all highly popular JavaScript libraries and frameworks that help developers build complex, reactive and modern user interfaces for the web.

Because the rendering pages will happen a lot in my project. That means the performance and the frameworks size of the framework is the thing I need to base on when choosing the front-end framework.

Angular uses a real Document Object Model (DOM), therefore it's best suited for the single-page-applications where content is updated from time to time. It makes the process of updating much slower and in case of losing the flow, it will take a lot of time to find out the issue. Thankfully to the two-way data binding process, all the changes made in the Model are replicated into the views in a secure and efficient way. Due to the wide range of features available, the application is much heavier (approximately 500KB) in comparison to Vue and React that slows down the performance a little.

Contrary to Angular, React uses a virtual DOM that enhances the performance of all-sizes applications that need regular content updates. Single-direction data allows better control over the project. The disadvantage might be the need of developers to constantly upgrade their skills as to the constantly evolving nature of React. As React doesn't provide a wide range of libraries, its size is much smaller than the size of Angular (approximately 100KB).

Vue also uses a virtual DOM, so the changes within a project are made without affecting the DOM properly. Vue possesses the smallest size of the three (approximately 80KB) which significantly speeds up its performance.

Next thing I need to consider is the popularity of the 3 frameworks and how complex to learn the frameworks.According to Stack Overflow Developer Survey Results 2019, React is the most loved by developers (74,5%) followed by Vue.js (73,6%) and only then Angular.js (57,6%)[13]. Because of Vue's popularity, finding input components and ready-to-use elements is extremely easy. They're all just a Google or GitHub search away.

After the research, I feel that Vue would be the most fit for our project, since it has:

- CLI
  Vue provides you with a CLI where you can pick what features you like when you are creating a project. Which makes the process of creating a project smooth and easy. React has the Create React App to set up your project. But when you want to add features, such as router, Sass, linter and more, you have to do it manually afterward.

[13]https://insights.stackoverflow.com/survey/2019

- Directives vs Components
  Vue has a clearer separation between directives and components. Directives are meant to encapsulate DOM manipulations only, while components are self-contained units that have their own view and data logic. In AngularJS, directives do everything and components are just a specific kind of directive.

- Vue also has single file components, where all component code (HTML, JavaScript, and CSS) are in the same file if wanted

- Vue comes with helpful developer toolset

2.2.4 MongoDB Atlas

MongoDB Atlas is a multi-cloud database service by the same people that build MongoDB. Atlas simplifies deploying and managing your databases while offering the versatility you need to build resilient and performant global applications on the cloud providers of your choice.

The reason why I choose MongoDB Atlas[14]

- Work with your data as code:
  Documents in MongoDB map directly to objects in your programming language. Modify your schema as your apps grow over time.

- Focus on building, not managing
  Let MongoDB Atlas take care of the infrastructure operations you need for performance at scale, from always-on security to point-in-time recovery.

- Simplify your data dependencies
  Leverage application data for full-text search, real-time analytics, rich visualizations and more with a single API and minimal data movement.

2.2.5 Google Maps Javascript API[15]

The Maps JavaScript API lets you customize maps with your own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite, hybrid, and terrain) which you can modify using layers and styles, controls and events, and various services and libraries

Direction Service:
You can calculate directions (using a variety of methods of transportation) by using the DirectionsService object. This object communicates with the Google Maps API Directions Service which receives direction requests and returns an efficient path. Travel time is the primary factor which is optimized, but other factors such as distance, number of turns and many more may be taken into account. You may either handle these directions results yourself or use the DirectionsRenderer object to render these results.

I decided to use this service to calculate the actual path for route optimization applying the vehicle routing problem algorithm as well as calculate the actual time to solve automatic order collection problem.

[14]MongoDB. *What is MongoDB Atlas?*.
Retrieve September 15, 2021 from https://docs.atlas.mongodb.com/

[15]Google Maps Platform.
Retrieved Mar 5, 2021 from https://developers.google.com/maps/documentation/javascript/overview

2.2.6 Google OR-tools[16]

OR-Tools is open source software for combinatorial optimization, which seeks to find the best solution to a problem out of a very large set of possible solutions. Here are some examples of problems that OR-Tools solves:

- Vehicle routing: Find optimal routes for vehicle fleets that pick up and deliver packages given constraints (e.g., "this truck can't hold more than 20,000 pounds" or "all deliveries must be made within a two-hour window").
- Scheduling: Find the optimal schedule for a complex set of tasks, some of which need to be performed before others, on a fixed set of machines, or other resources.
- Bin packing: Pack as many objects of various sizes as possible into a fixed number of bins with maximum capacities.

In most cases, problems like these have a vast number of possible solutions—too many for a computer to search them all. To overcome this, OR-Tools uses state-of-the-art algorithms to narrow down the search set, in order to find an optimal (or close to optimal) solution.
I will choose OR-tools to solve vehicle routing (optimize) in my system.

2.2.7 ElementUI [17]

Element is a Vue UI component library with a large community. It's not only for front-end developers but also provides a full UI kit that designers and product managers can work with. It's specifically tailored for creating desktop UIs, but it does support some responsive features such as hiding elements based on the window size and creating grids.

This library is developed mostly by Chinese contributors and the original documentation was written in Chinese. It's highly suggested to check out the internationalization guide before starting a project, as the default language of the project is Chinese. Element is a great kit for developing complex desktop UIs by teams of developers, designers, and product managers.

Installation

```
npm i element-ui -S
```

2.2.8 Python-shell library

A simple way to run Python scripts from Node.js with basic but efficient inter-process communication and better error handling.[18]. I use this library to run scripts (OR-tools algorithm use python) from Node.js.

Features

- Reliably spawn Python scripts in a child process

[16]Google OR-tools *Google OR-tools*. Retrieved March 15, 2021 https://developers.google.com/optimization

[17]ElementUI documentation. Retrieved Mar 5, 2021 from https://element.eleme.io/#/en-US

[18]Python-shell documentation. Retrieved Sep 12, 2021 from https://www.npmjs.com/package/python-shell

- Built-in text, JSON and binary modes
- Custom parsers and formatters
- Simple and efficient data transfers through stdin and stdout streams
- Extended stack traces when an error is thrown

Installation

```
npm install python-shell
```

2.2.9 Telegram bot API[19]

For chatbot API using:

API	API Features	Pricing	Ease of Use
Facebook Messenger API	Communicate with people in Facebook groups and chats, make posts, answer questions, complete other specified actions	Free	Easy
Slack API	Interact with users automatically, send direct messages, get mentioned, embed bots on external platforms	Free	Easy
Telegram API	Get customized notifications, accept payments, integrate with external platforms	Free	Easy
ApiAI API	Create text-based and voice conversational interfaces and deploy on various platforms	Free and paid plans	Easy
ChatBot API	Create bot stories, interactions, rich messages, get performance reports	Free and monthly paid plan from $50	Easy

Figure 2.1: Chatbot API comparison

This API allows you to connect bots to our system. Telegram Bots are special accounts that do not require an additional phone number to set up. These accounts serve as an interface for code running somewhere on your server.

To use this, you don't need to know anything about how our MTProto encryption protocol works — our intermediary server will handle all encryption and communication with the Telegram API for you. You communicate with this server via a simple HTTPS-interface that offers a simplified version of the Telegram API.

I decided to use telegram bot API to inform and give route instructions to the driver

[19]Telegram bot APIs documentation. Retrieved Mar 5, 2021 from https://core.telegram.org/bots

Chapter 3

System design

3.1 Application architecture

Our website will contain 2 main parts:

- Front-end application for user
- Back-end server

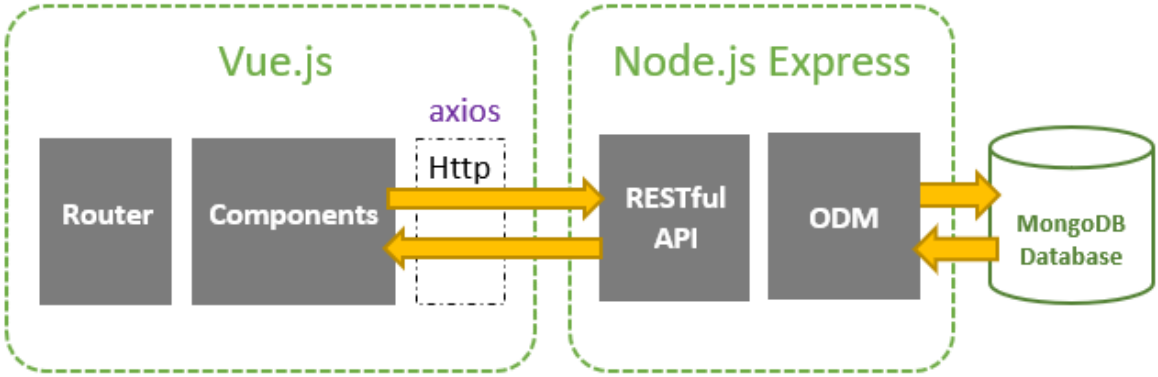


Figure 3.1: Website architecture

3.1.1 Front-end application's architecture

The front-end will display views to users while also taking user requests through interactive user interface and sending them to either back-end for requests or messages in web-socket.

Modern front-end frameworks and libraries make it easy to create reusable UI components. This is a step in a good direction to create maintainable front-end applications. Vue.js is a progressive JS framework used for developing single-page applications and user interfaces (UIs). This is an open-source framework that employs 'high decoupling', permitting developers to create web interfaces progressively. Vue is lightweight and easy to write. Owing to its use of components and familiar templating syntax, migrating or integrating into current projects to Vue.js is smoother and faster. You have an API, a Store, and a View. The API is the service for getting or setting data. The Store is where the data is stored in our app. The view is where the store data is combined with markup to output the rendered page. In the stack that my project is using the View is Vue. The store is Vuex

and there is a lot of glue between them to make it work, like vue-router, and others,...

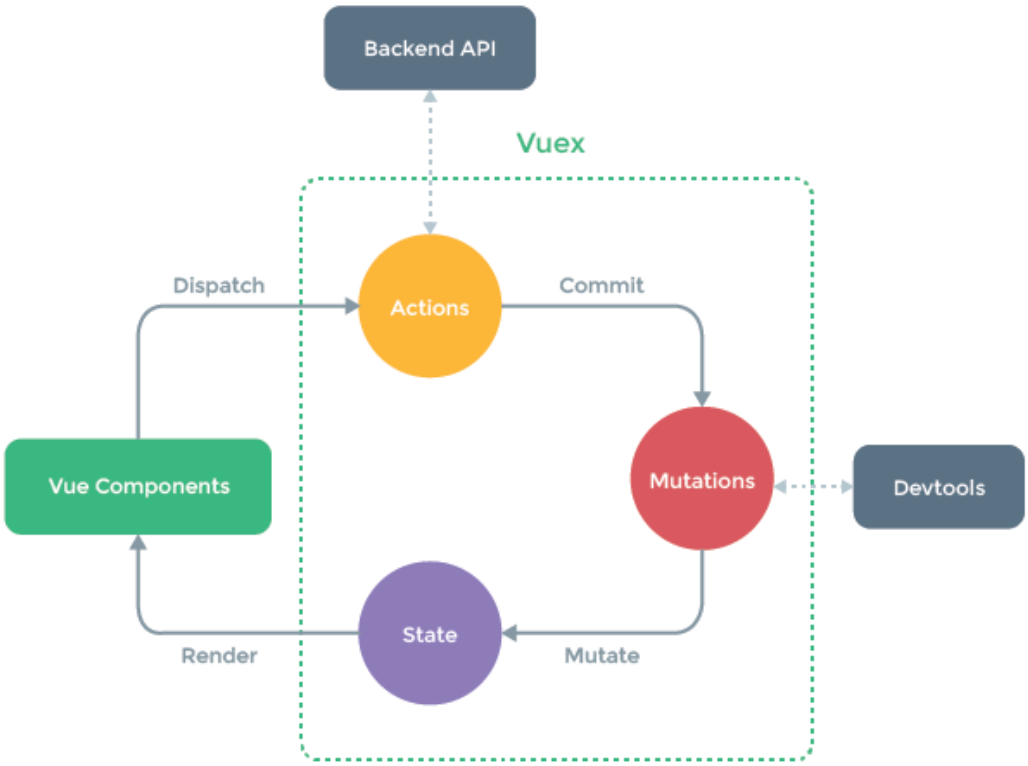


Figure 3.2: Front-end architecture[1]

The Vue components that compose the user interface dispatch actions. These may check certain conditions, access a backend api, and even dispatch other actions. And at some point they may well need to change the application state — this is done by committing a mutation. Only mutations can change the application state.

a. State

Vuex uses a single state tree - that is, this single object contains all application level state and serves as the "single source of truth." This also means usually each application have only one store. A single state tree makes it straightforward to locate a specific piece of state, and allows us to easily take snapshots of the current app state for debugging purposes.
The data that store in Vuex follows the same rules as the data in a Vue instance.
Whenever state of store changes, it will cause the computed property to re-evaluate, and trigger associated DOM updates. However, this pattern causes the component to rely on the global store singleton. When using a module system, it requires importing the store in every component that uses store state, and also requires mocking when testing the component.

[1]Colin Fallon (2017, Mar 25) *Uni-directional data flow with Vuex.* Retrieved Nov 10, 2021 from https://medium.com/@softwarecf/uni-directional-data-flow-with-vuex-4f31d7ac8c83

b. Mutations

The only way to actually change state in a Vuex store is by committing a mutation. Vuex mutations are very similar to events: each mutation has a string type and a handler. The handler function is where we perform actual state modifications, and it will receive the state as the first argument.

Since a Vuex store's state is made reactive by Vue, when we mutate the state, Vue components observing the state will update automatically. This also means Vuex mutations are subject to the same reactivity caveats when working with plain Vue.

c. Actions

Actions can be synchronous or asynchronous and shouldn't change the state directly but actions commit mutations. Actions can invoke multiple mutations.

Also here's some more info and best practices for mutations and actions:

Mutations

- It's convention to uppercase mutation names
- Called via

```
commit('MUTATION_NAME', payload)
```

- Payload is optional
- Should not contain any logic of whether to mutate the data or not
- It's best practice to only call them from your actions (even though you do have access to calling them in components)

Actions

- Can be called from within other actions in the store
- Are the primary way to change state from within components called via

```
dispatch('actionName', payload)
```

- Payload is optional
- Can contain logic of what to mutate or not mutate
- Good practice to define as async to begin with so that if the logic changes from synchronous to asynchronous the places where you dispatch the actions don't have to change.

3.1.2 Back-end server

My back-end server will contain a ExpressJs server, a DBMS to store system data. The HTTP and socket request sent from user client will be caught in ExpressJs server and handle based on the following architecture:

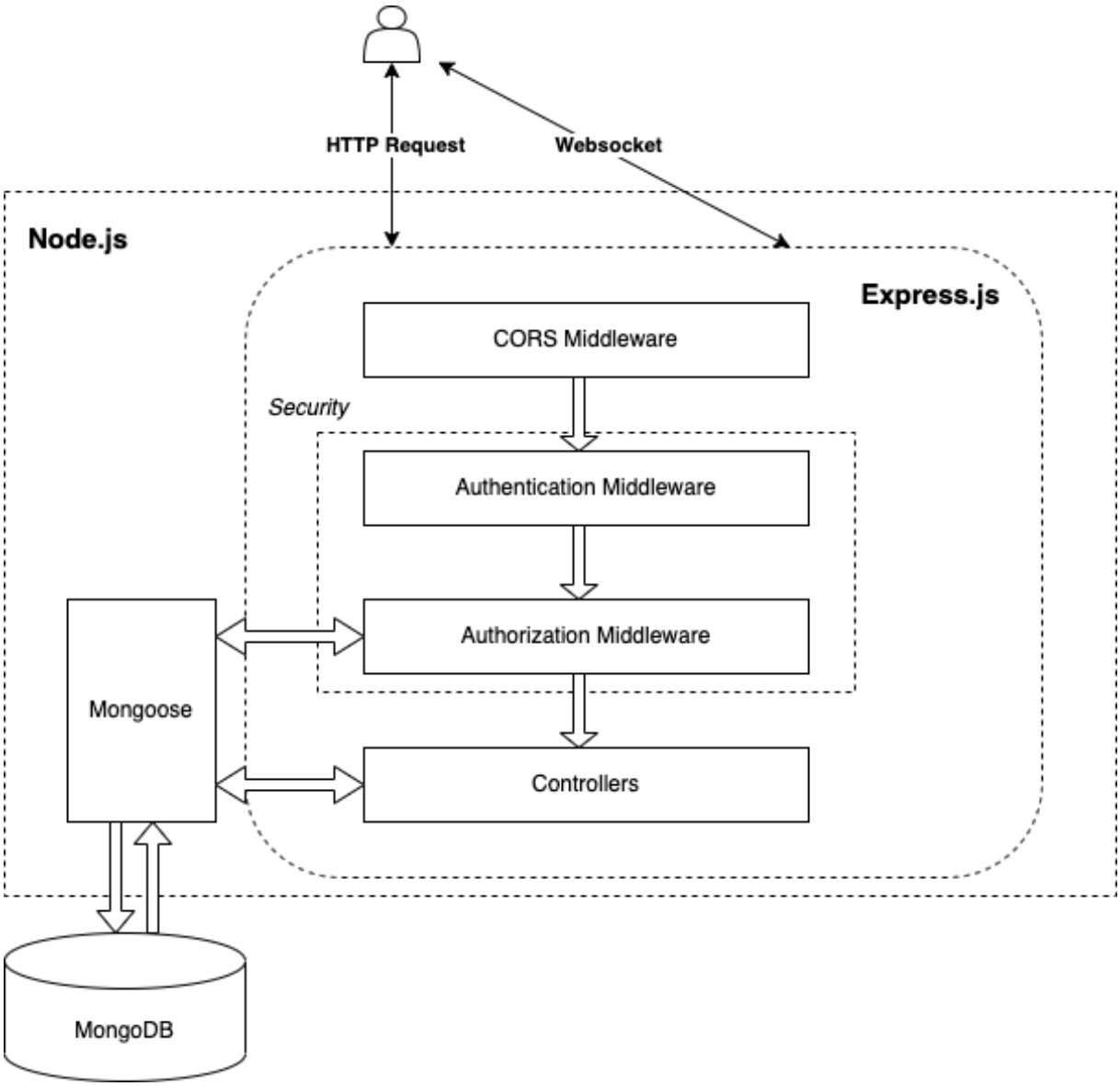


Figure 3.3: Back-end architecture

Via Express routes, HTTP request that matches a route will be checked by CORS Middleware before coming to Security layer.

Security layer includes:

- JWT Authentication Middleware: verify SignUp, verify token
- Authorization Middleware: check User's roles with record in database

An error message will be sent as HTTP response to Client when the middlewares throw any error.

Controllers interact with MongoDB Database via Mongoose library and send HTTP response (token, user information, data based on roles...) to Client.

The system automatically updates the collection order every time a new order is

placed from the customer, I need a real time notification for employee to manage and prepare product for each order. By using sockets to send messages, it allows me to update collection and responding speed which is one of the most important aspects of my website. As mentioned in the survey, I have chosen NodeJs as my development environment so I will be using the Socket.IO library in order to implement this communication service as Socket.io is the best library which supports socket based connection.

To authenticate an user in HTTP requests, I decided to use JWT(JSon Web Token) in order to store the user's authentication information on the browser and from them identify the user for action through API call. The token will be passed in the authorization field of Header and will be verified when the request goes through our guards.

3.2 Route Optimization

3.2.1 Vehicle Routing Problem

Vehicle routing problem (VRP) belongs to the class of combinatorial optimization problems, concerned with transporting goods from one or more depot to a set of customers. The VRP problem was first proposed by Dantzig and Ramser in 1959[2], in which the author introduced a practical problem of moving gasoline to gas stations. With the number of petrol trucks and their storage capacity known in advance, the author wants to find a way to move the gas trucks to the gas stations so that each gas station is visited exactly once, the total amount of gasoline delivered to the gas station visited by a vehicle does not exceed its capacity, and at the same time the distance traveled by the vehicles is as small as possible. This problem is the simplest form of the class of VRP problems.

The vehicle routing problem (VRP) is defined as follows. Vehicle routing problem is a set of combination optimization problems to determine a set of optimal routes, each route taken by a transport vehicle starting and ending at the depot, which satisfying customer requirements with all constraints so that costs are minimal. The delivery of products and services related to a set of customers by a set of means of transport over a period of time and the execution of the movement according to an appropriate route.

Some concepts of the components related to the VRP problems:

- **Vehicle**
  Delivery vehicle is a vehicle that moves between customers to make deliveries. Each vehicle has a number of attributes associated with it, such as the maximum amount of goods it can carry, or the maximum number of customers it can visit.
- **Customers**
  Can pick up the products delivered by the vehicle or transfer the products to the vehicle for transporting to the depot, and this is also the destination point of vehicle.
- **Depot**
  Is a place to store products and it may also be a place of departure / return of some vehicles.
- **Network traffic**
  Vehicles travel between customers through this transport network. The traffic network will normally be represented by a graph, in which the vertex of the graph is

[2]G. B. Dantzig, J. H. Ramser. (1959, Oct 1). *The Truck Dispatching Problem.* Retrieved June 19, 2021, from https://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.80

the customer and the warehouse, the edges connecting the two vertices will have a corresponding weight of the cost.

- **Route**
  Each trip starts from the departure point and returns to the initial point of a vehicle.

- **Fleet**
  Fleet corresponds to the set of available vehicles that can be used for delivery. Depot has can be homogeneous (all vehicles are the same, have the same attributes) or not.

- **Objective Routing**
  The objective function of a VRP problem is usually to minimize/maximize some quantity. The most basic and common objective is to minimize the total distance traveled by the vehicle. Another possible objective function is to maximize the profits obtained by serving a set of customers.

- **A VRP solution**
  A solution to the VRP problem is usually a set of routes along with the corresponding trucks that will move along that route. Some other complex situations of the VRP problem require some additional information to represent the solution, such as timetables for the vehicles. Each solution can be evaluated by its goodness through the objective function.

- **Constraint**
  Constraints are conditions that the solution to the VRP needs to be satisfied. Some common constraints are: maximum number of bus routes, maximum distance a vehicle can travel,...

3.2.2 Some variants of Vehicle Routing Problem[3]

- Capacitated of vehicle routing problem (CVRP):
  The basic capacitated Vehicle Routing Problem (CVRP) can be described in the following way. Given a set of homogeneous vehicles each of capacity Q, located at a central depot and a set of customers with known locations and demands to be satisfied by deliveries from the central depot, each vehicle route must start and end at the central depot and the total customer demand satisfied by deliveries on each route must not exceed the vehicle capacity, Q. The objective is to determine a set of routes for the vehicles that will minimize the total cost. The total cost is usually proportional to the total distance traveled if the number of vehicles is fixed and may also include an additional term proportional to the number of vehicles used if the number of routes may vary.

- Vehicle routing problem with time window (VRPTW):
  In a Vehicle Routing Problem with Time Windows (VRPTW) the capacity constraint still holds and each customer i is associated with a time interval [ai, bi], called the time window, and with time duration, si , the service time. These constraints restrict the times at which a customer is available to receive a delivery. This problem is often common in real-world applications since the assumption of complete availability overtime of the customers made in CVRP is often unrealistic.

[3]Sajaykumar J (2020, April 24). *Vehicle Routing Problem and its variants.* Retrieved Mar 15, 2021, from https://www.linkedin.com/pulse/vehicle-routing-problem-its-variants-sajaykumar-j

Time windows can be set to any width, from days to minutes, but their width is often empirically bound to the width of the planning horizon. The presence of time windows imposes a series of precedence on visits, which makes the problem asymmetric, even if the distance and time matrices were originally symmetric. VRPTW is also NP-hard and even finding a feasible solution to VRPTW is an NP-hard problem.

- Vehicle Routing Problem with Pickup and Delivery (VRPPD):
  The pickup and delivery problem (PDP) is a problem of finding a set of optimal routes, for a fleet of vehicles, in order to serve a set of transportation requests. Each vehicle from the fleet of vehicles has a given capacity, a start location, and an end location. Each transportation request is specified by a load to be transported, an origin, and a destination location. In other words, the pickup and delivery problem deals with the construction of optimal routes in order to visit all pickup and delivery locations, and satisfy precedence and pairing constraints. Precedence constraints deal with the restriction, in which each pickup location has to be visited prior to visiting the corresponding delivery location. Pairing constraints restrict the set of admissible routes so that one vehicle has to do both the pickup and the delivery of the load of one transportation request.

- Vehicle Routing Problem with Multi Depot (VRPMD):
  Multi Depot Vehicle Routing Problem is one type of Vehicle Routing Problem that has more than one depot that serving customer demand. Multi Depot Vehicle Routing Problem is aimed to establish the route of delivery at each depot so obtained a route with a minimum distance. It should be noted that each customer is served by one vehicle once. In addition, each route starts and ends at the same depot. The total number of customer demands in one route is not greater than the vehicle's capacity. Routes formed between the customer and the depot are obtained based on the minimum distance from the depot to the customer and back to the depot, without violating the existing constraints.

3.2.3 CVRP Problem

CVRP requires delivery to customers, each with a given deman, using vehicles with limited storage capacity. Cargo trucks are parked at a depot, routes will start and end at that depot. The objective is to define routes so that every customer is visited, the vehicle does not exceed its capacity, and the total distance traveled by the vehicles is minimal.

The CVRP problem is defined on a graph G = $\left(\text{V,E}\right)$ representing the traffic network. The vertex set V = $\left\{0,1,...,\text{n}\right\}$ represents different locations, while the edge set E represents the paths in the transport network. The vertex i = 1,2,...,n is the customers and the vertex 0 corresponds to the depot.

Each edge (i,j) ϵ E has a weight d_{ij} that represents the distance between the location corresponding to vertex i and the location corresponding to vertex j. The distances are assumed to be symmetric ($d_{ij} = d_{ij}$ with $\forall i, j \epsilon$ V) and satisfy the triangle inequality ($d_{ik} + d_{kj} \geq d_{ij}$ with $\forall$ i, j, k ϵ V).

Each customer i ϵ V $\setminus$ $\left\{0\right\}$ will be assigned an integer q_i that is the demand of customer i, i.e. the quantity of goods that this customer needs. There are K vehicles, each of which can carry a maximum of Q quantity of goods. Each vehicle can only perform a maximum of one trip.

Each route starts from the depot, visits a number of customers, and then returns to

the depot. Each route r can be represented by an ordered sequence of customers visited by the route: r = $(c_1, c_2, ..., c_t)$. The symbol r_{size} is the number of customers that route r visits. The customer s visited by route r, denoted by r[s], is c_s. Since a bus route must start and end at depot 0, for convenience, we default r[0] = r[r_{size} + 1] = 0.

The objective of the CVRP problem is to find a solution to the vehicle routing Sol, which is a set of vehicle routes, Sol = $\left\{r_1, r_2, ..., r_m\right\}$. This solution must satisfy with the following constraint:

1. $|Sol| \leq$ K.
   The number of routes must not exceed the number of existing vehicles, which is K.

2. $r_i \cap r_j = \emptyset$ with $\forall r_i, r_j \in$ Sol, $r_i \neq r_j$ and $\cup_{r_i \epsilon Sol}\ r_i = V \setminus \left\{0\right\}$.
   Each customer must be visited exactly once.

3. $\Sigma_{c \epsilon r}\ q_c \leq Q\ \forall r \in$ Sol.
   The total demand of customers visited on a route should not exceed vehicle capacity.

4. Dist(Sol) = $\Sigma_{r \epsilon Sol}(\Sigma_{i=0}^{r_{size}} d_{r[i]r[i+1]})$ is minimum.
   The total distance traveled by the vehicles is the minimum.

A solution that satisfies all conditions from 1 to 3 above is called a valid solution. Otherwise, the solution is said to be invalid.

The quality of a Sol solution is evaluated by Dist(Sol). This value is smaller, the solution is better. The solution Sol $*$ is called to be optimal if and only if it is a valid solution and there are no other valid Sol 'solutions such that Dist(Sol ') $<$ Dist(Sol $*$). So that mean there can be multiple optimal solutions to a CVRP problem.

Note, when K = 1, Q = ∞, the CVRP problem becomes the TSP (Traveling Salesman Problem) postman problem. Therefore, the CVRP problem is more difficult than the TSP problem.

3.2.4 Survey relevant theory

1. **Exact algorithm**

   - **Brand and bound**
     Branch and bound is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as mathematical optimization. A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root. The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm.
   - **Dynamic programming**
     Solve the complex problem by simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner.
   - **Set partitioning**
     This method presents the CVRP problem as a set division problem, and

then uses the existing methods to solve the set division problem. [4]

2. **Heuristic algorithm**

- **Saving algorithm**

  The Saving algorithm (SA) is the most famous and oldest heuristic. This algorithm is very often used to find initial solutions because of its fast speed and good results.[5]

  The algorithm will calculate saving(i, j) for every pair of customers (i, j) as the cost saved when connecting the two routes, one is r_1 to visit the last customer i, the other is r_2 to visit the first j customer. We have the saving formula (i,j)=$d_{0i} + d_{0j}$ - d_{ij}.

  An initial solution is created in which each route only visits one customer. Then, each pair of customers (i, j) that are traversed in descending saving(i, j) will be processed as follows. If two routes r1, r2 can be "connected" to each other (ie, after connecting, the condition is still satisfied that the total demand does not exceed the vehicle capacity) then connect them to form a new route r = $r_1 \times r_2$. The solution is updated by removing r_1, r_2 and adding a new r: Sol = Sol $\setminus \{r_1, r_2\} \cup \{r\}$. The algorithm terminates when the solution cannot be changed anymore.

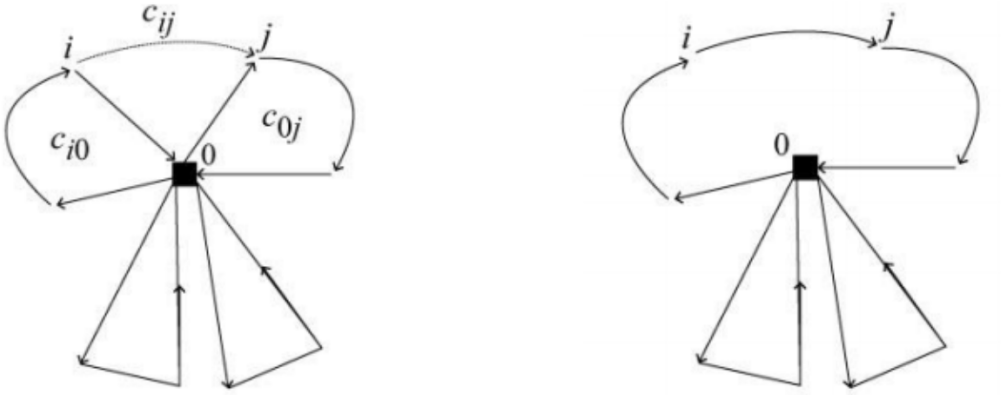


Figure 3.4: Saving algorithm

- **Sweep algorithm**

  The Sweep algorithm [6] is imagine that a line (often a vertical line) is swept or moved across the plane, stopping at some points. Geometric operations are restricted to geometric objects that either intersect or are in the immediate vicinity of the sweep line whenever it stops, and the complete solution is available once the line has passed over all objects.

[4]Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi (2006, Aug 1). *An exact algorithm for the vehicle routing problembased on the set partitioning formulation with additional cuts.* Retrieved Sep 13, 2021, from https://link.springer.com/article/10.1007/s10107-007-0178-5

[5]G. Clarke, J. W. Wright (1964, Aug 1). *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points.* Retrieved Sep 13, 2021, from https://pubsonline.informs.org/doi/abs/10.1287/opre.12.4.568

[6]Renaud, Jacques & Boctor, Fayez F (2002, Aug). *A sweep-based algorithm for the fleet size and mix vehicle routing problem.* Retrieved Sep 13, 2021, from https://ideas.repec.org/a/eee/ejores/v140y2002i3p618-628.html

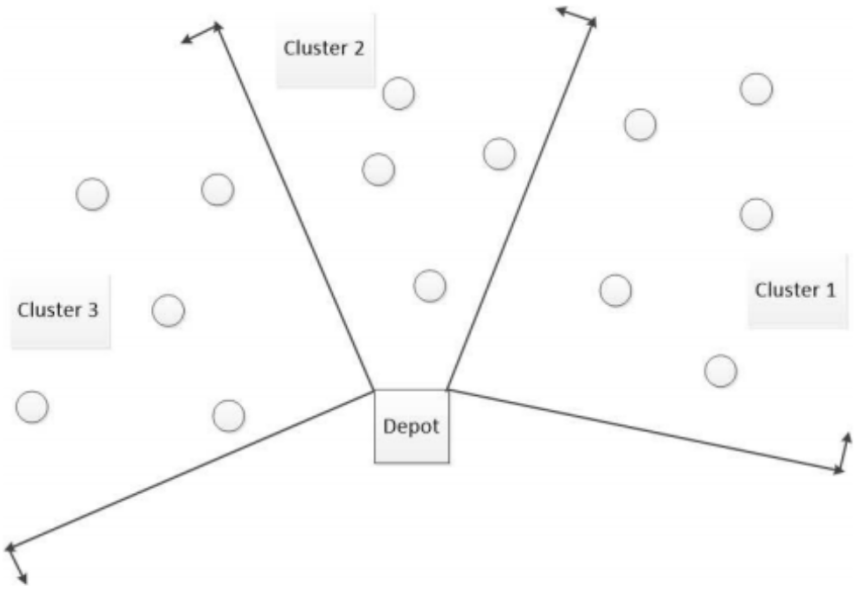


Figure 3.5: Sweep algorithm

3. **Meta-heuristic algorithm**

   In computer science and mathematical optimization, a metaheuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. Meta-heuristics may make relatively few assumptions about the optimization problem being solved and so may be usable for a variety of problems.

   Some known algorithms to optimize vehicle routing problem are tabu search algorithm, genetic algorithm (GA), ...

3.3 Design algorithm for automatic order collection

I define these type of duration and time as follow:

- d_o: duration to run optimization route tool
- d_p: duration to prepare product for order.
- t_p: promise time.
- $d_{deliver}$: duration to delivery from depot to pick-up point of the order

The order delivery process conduct as follows: The staff prepare for orders. System will notify the time to run optimization tool. After proceeding optimization tool, the vehicle will deliver.
I divide the timeline for promise time from 9am to 10pm with interval is 15 minutes: 9.00am, 9.15am, 9.30am ..., 8.30pm, 8.45pm, 9.00om, 9.15pm, 9.30pm, 9.45pm, 10.00pm. And my system collect orders according to the promise time satisfied with:

1. All orders has the same promise time will be collected.

2. Time (run optimization tool) = MIN(t_p - d_o - $d_{deliver}$)

3. If customer place order (same promise time) at the time later than time (run optimization), this orders will be collected in the next collection.

I define f_c(h) is the least of set $\left\{9.00am, 9.15am, ..., 9.30pm, 10.00pm\right\}$ greater than or equal to h.

So $t_p \geq f_c$ (Now + d_o + d_p + $d_{deliver}$)

3.4 Database design

1. **User documentation**: contains user data

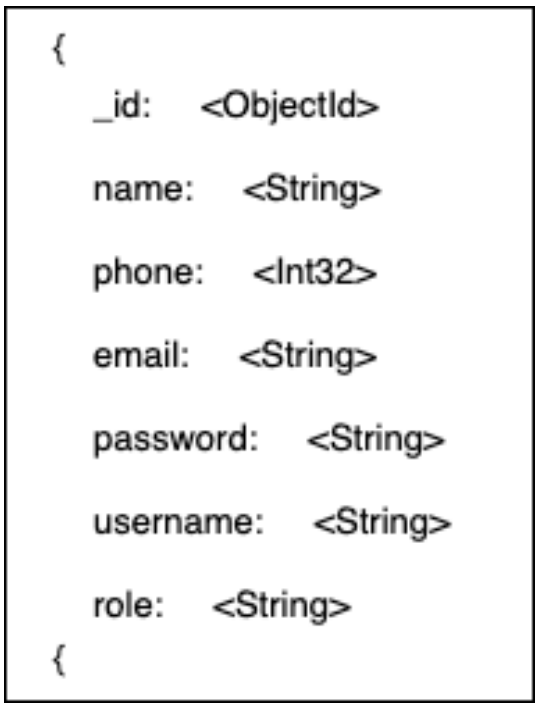


Figure 3.6: User Documentation

Description:

- `_id`: Id of user
- name: Name of the user
- username: Username of the user
- phone: Phone of the user
- email: Email of the user
- role: Role of the user (admin or employee)
- password: Password of the user.

2. **Customer documentation**: contains customer data

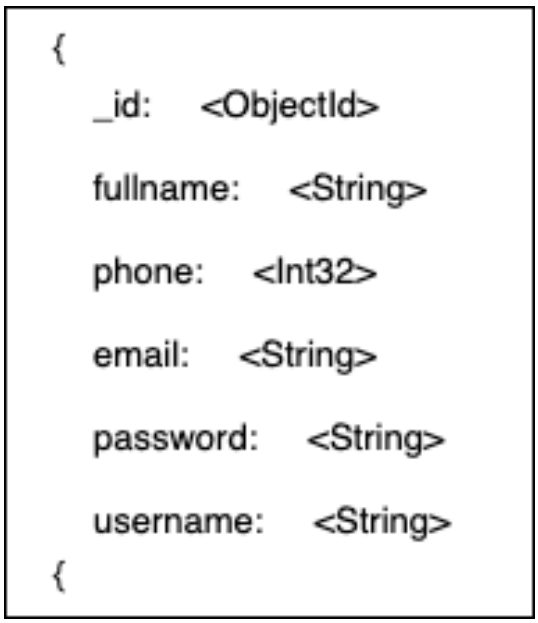


Figure 3.7: Customer Documentation

Description:

- `_id`: Id of customer
- fullname: Fullname of the customer
- username: Username of the customer
- phone: Phone of the customer
- email: Email of the customer
- password: Password of the customer

3. **Product documentation**: contains product data

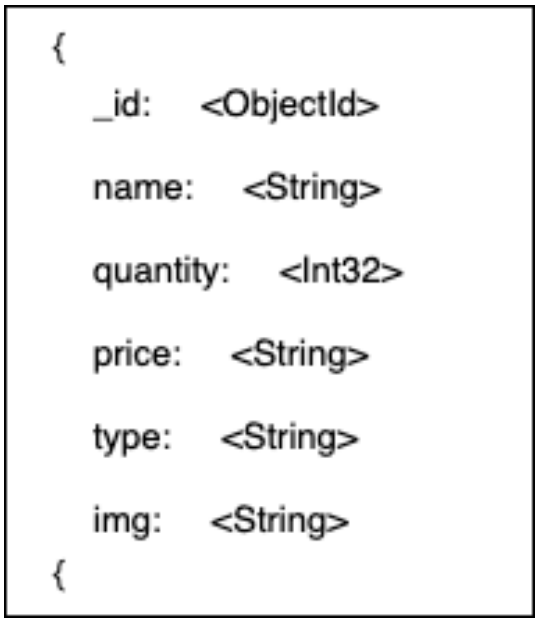


Figure 3.8: Product Documentation

Description:

- `_id`: Id of product

- name: Name of the product
- quantity: Quantity of the product
- price: Price of the product
- type: Type of the product
- img: Image of the product

4. **Vehicle documentation**: contains vehicle data

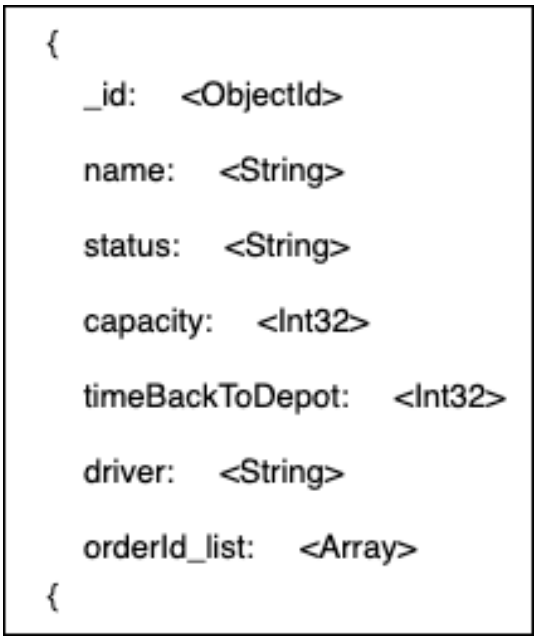


Figure 3.9: Vehicle Documentation

Description:

- `_id`: Id of vehicle
- name: Name of the vehicle
- capacity: Maximum capacity of vehicle
- status: Status of the vehicle. If the vehicle has not been delivered, the status is available. On the contrary, status of vehicle is unvailable.
- `orderId_list`: Include the id of all the orders that the vehicle is about to deliver.
- timeBackToDepot: The time the will delivery and return to the depot.

5. **Order documentation**: contains order data

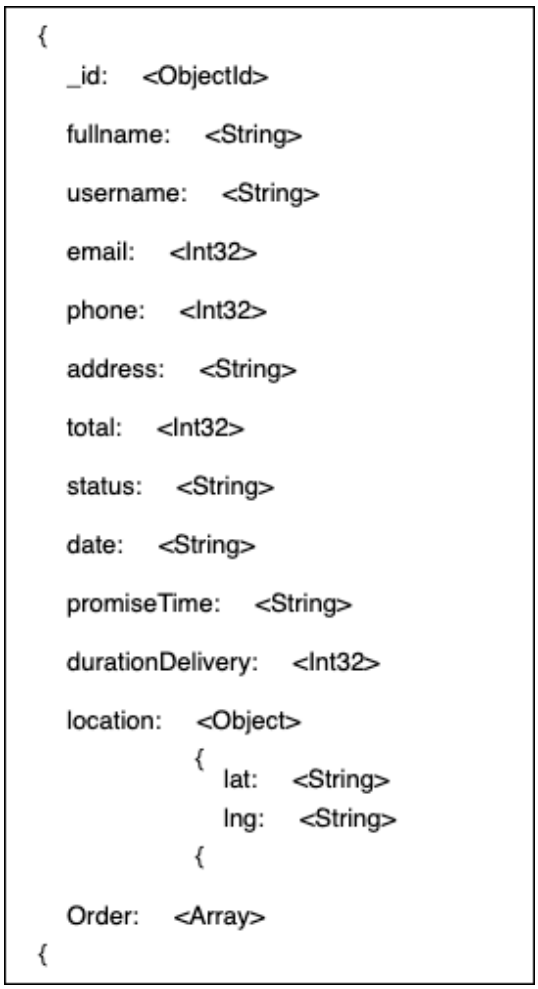


Figure 3.10: Order Documentation

Description:

- `_id`: Id of order
- fullname: Fullname of customer placed order
- username: Username of customer placed order
- email: Email of customer placed order
- phone: Phone of customer placed order
- address: Address of customer placed order
- total: Total price include taxes of the order
- status: Status of the order.
  - New: New order placed from customer. The system will automatically collect new orders before moving to processing status. Each group orders have timeline to change processing status and put into optimal system.
  - Processing: This is the process of running the optimization system to give the optimze route for each group processing order before delivering.
  - Delivering: This is the process of transporting goods to the receiving places.
  - Completed: The delivering process is completed, ie the vehicle goes to the depot.
- date: This is the time customer placed the order.

- promiseTime: This is the time customers want to receive the goods.
- durationDelivery: This is the duration from the depot to the place of receipt.
- order: The information of customer cart.

6. **Config documentation**: contains vehicle data

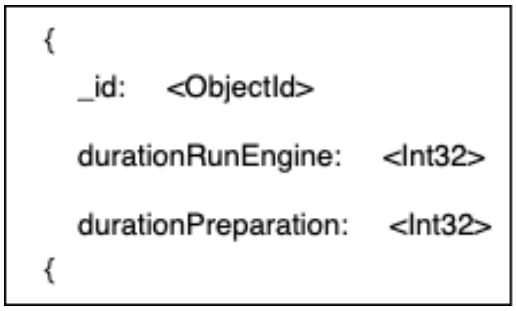


Figure 3.11: Config Documentation

Description:

- `_id`: Id of configuration
- durationRunEngine: Optimal system running time
- durationPreparation: Duration the staff prepares the order.

3.5 UI design

For the user interface of my website, I select some available interface templates for e-commerce website as well as admin website. In addition, I have create some mock-ups so that my final product can look as well as I pleased and for me to have references during development

3.5.1 E-commerce website

For e-commerce website, I select this template according to link below:

https://themes09.anvanto.com/super/themes/grain/en/#

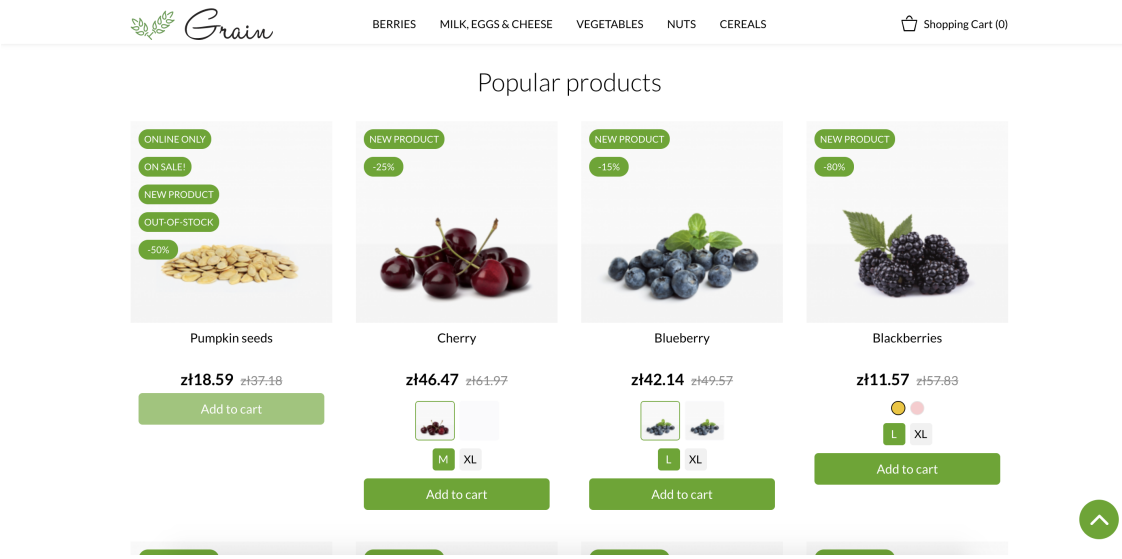


Figure 3.12: Home Page Mock-up

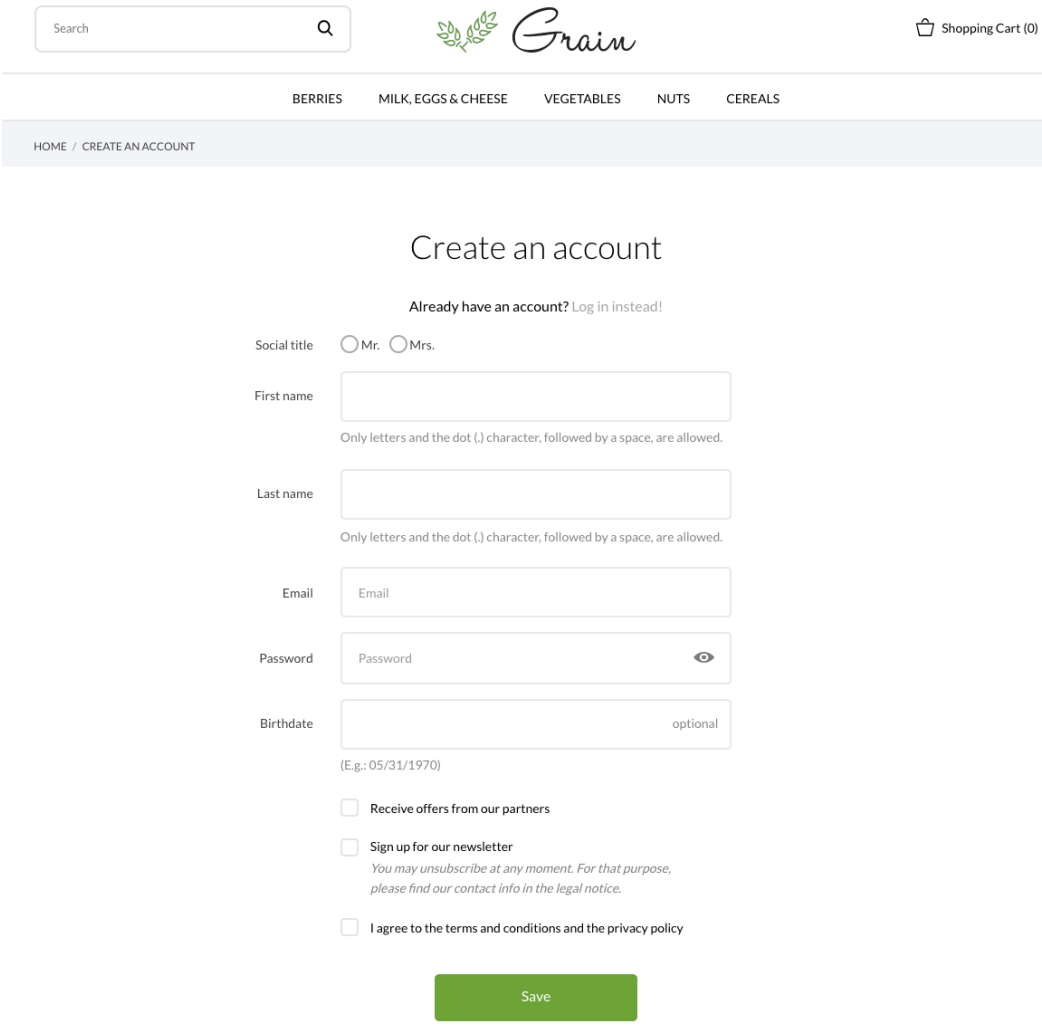


Figure 3.13: Sign Up Page Mock-up

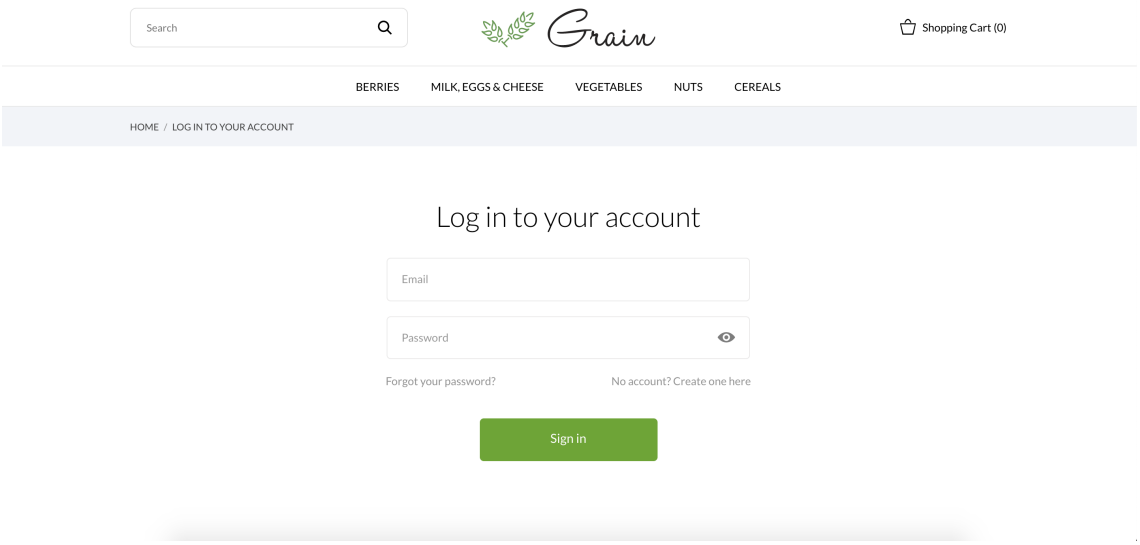


Figure 3.14: Login Page Mock-up

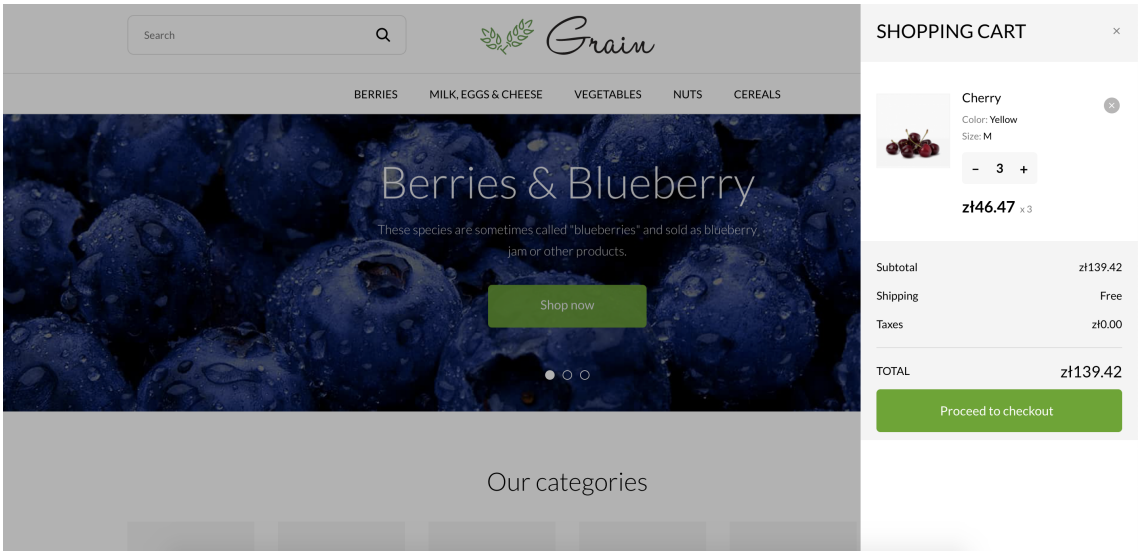


Figure 3.15: Add To Cart Mock-up

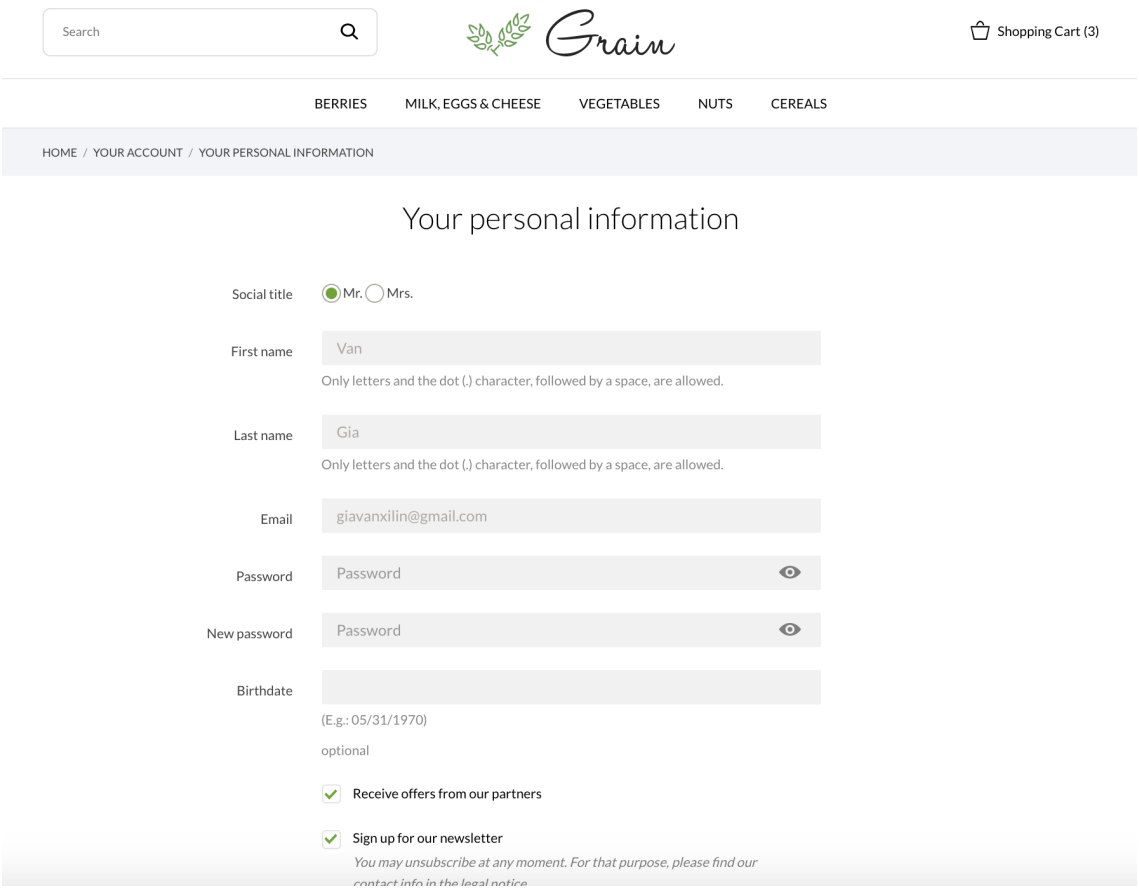


Figure 3.16: Customer Information Page Mock-up

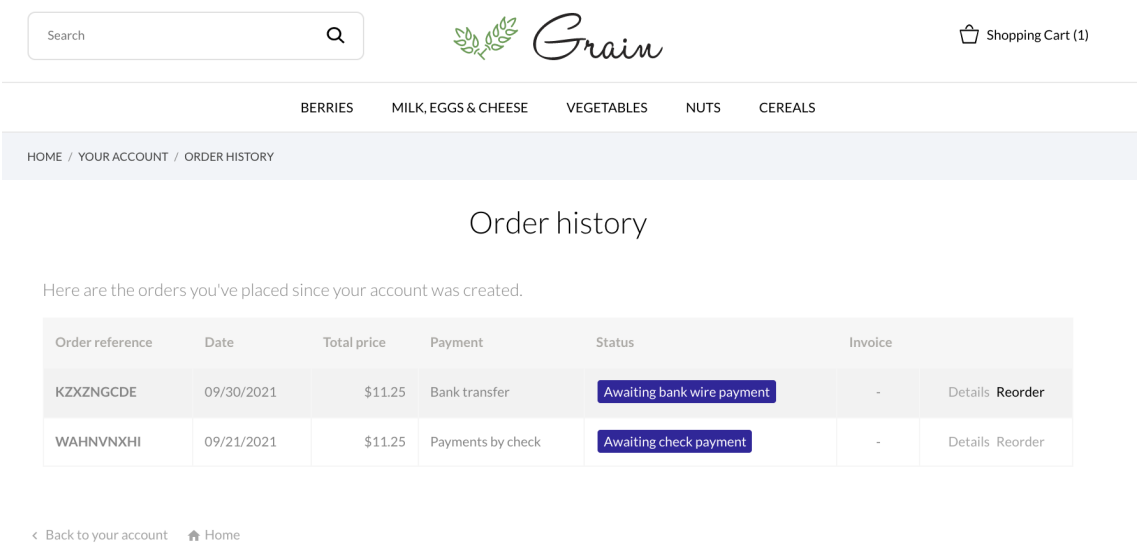


Figure 3.17: Order History page Mock-up

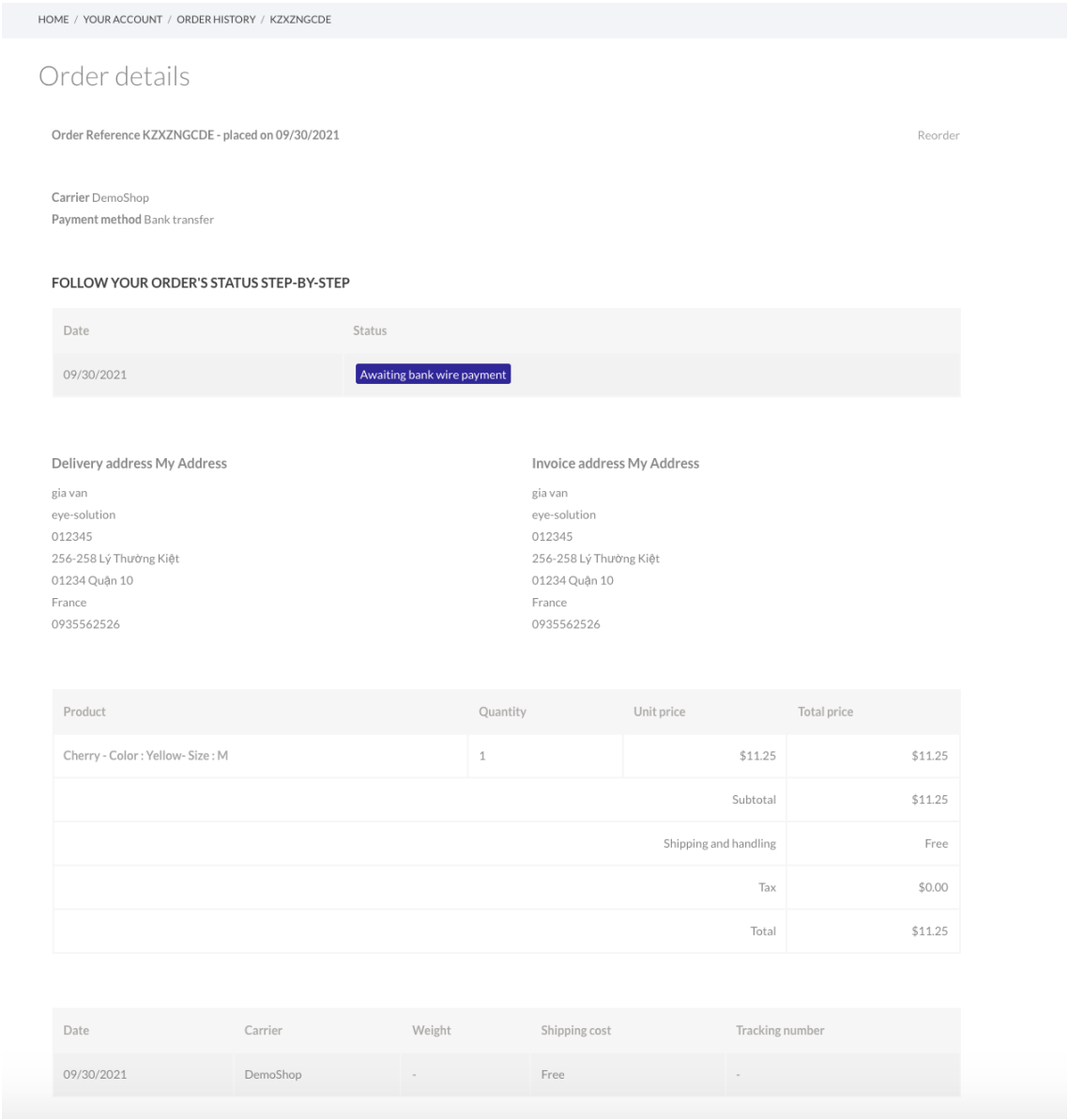


Figure 3.18: Order Details page Mock-up

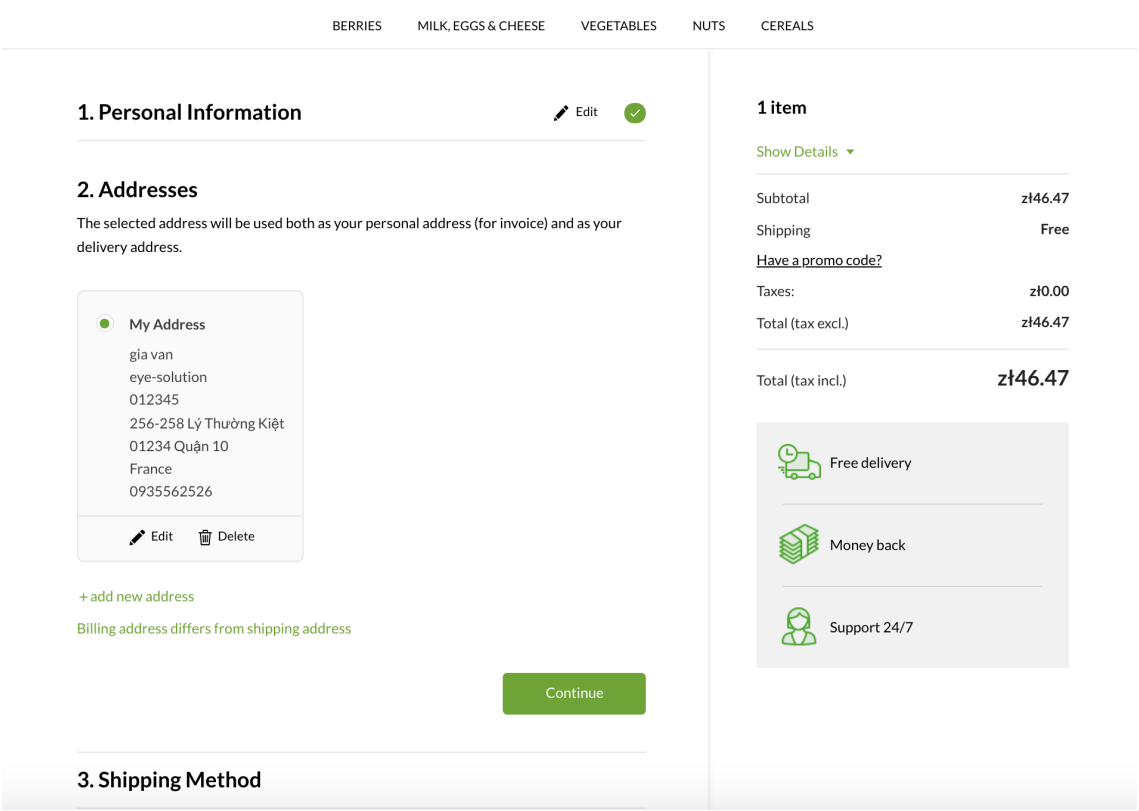


Figure 3.19: Checkout Order page Mock-up

3.5.2 Admin Dashboard

For admin dashboard, I select this template according to link below:

https://www.creative-tim.com/live/vue-material-dashboard-laravel

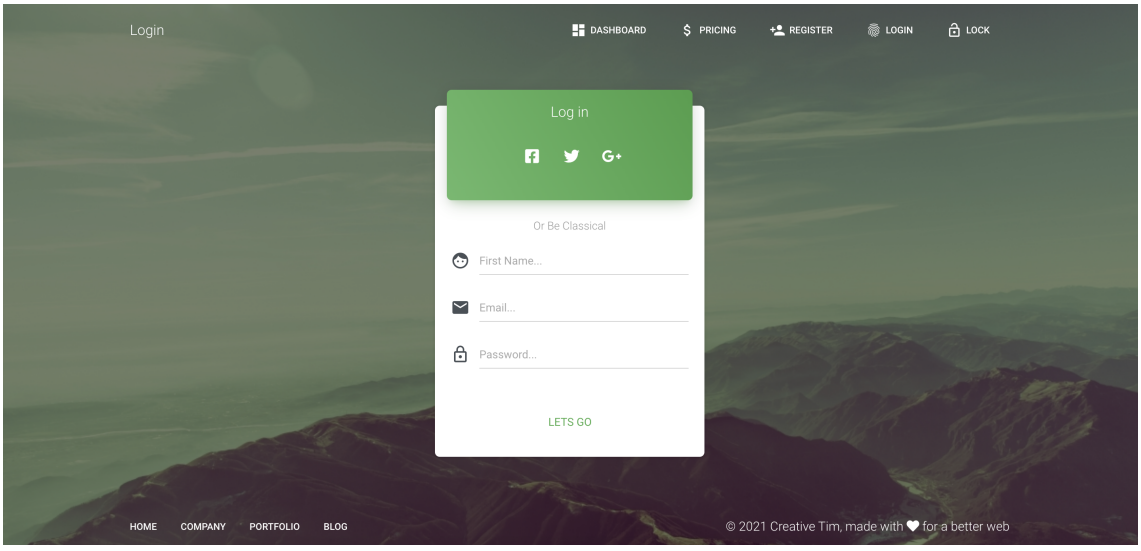


Figure 3.20: Admin Login page Mock-up

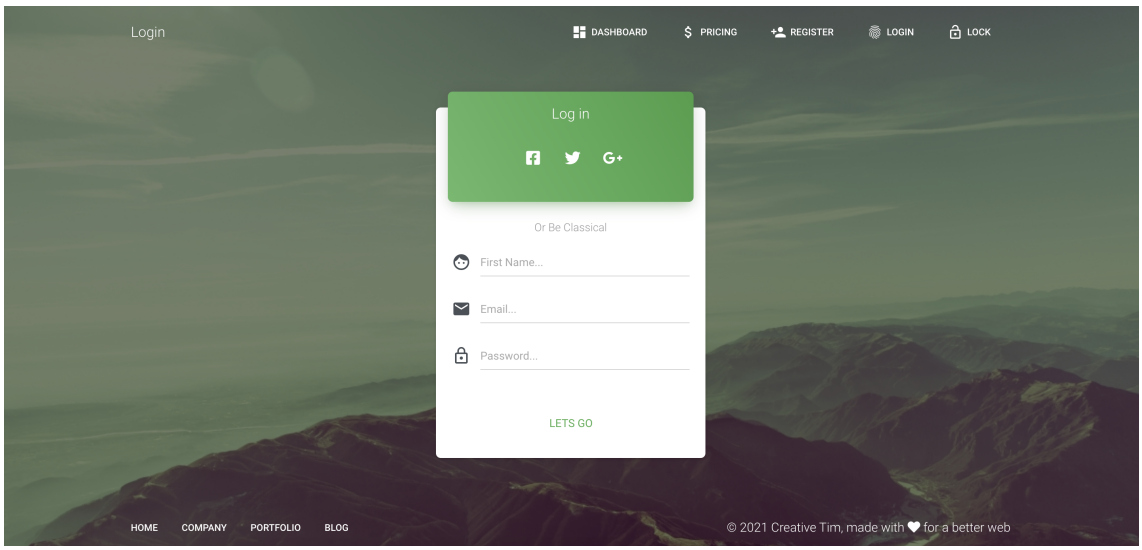


Figure 3.21: Admin Login page Mock-up

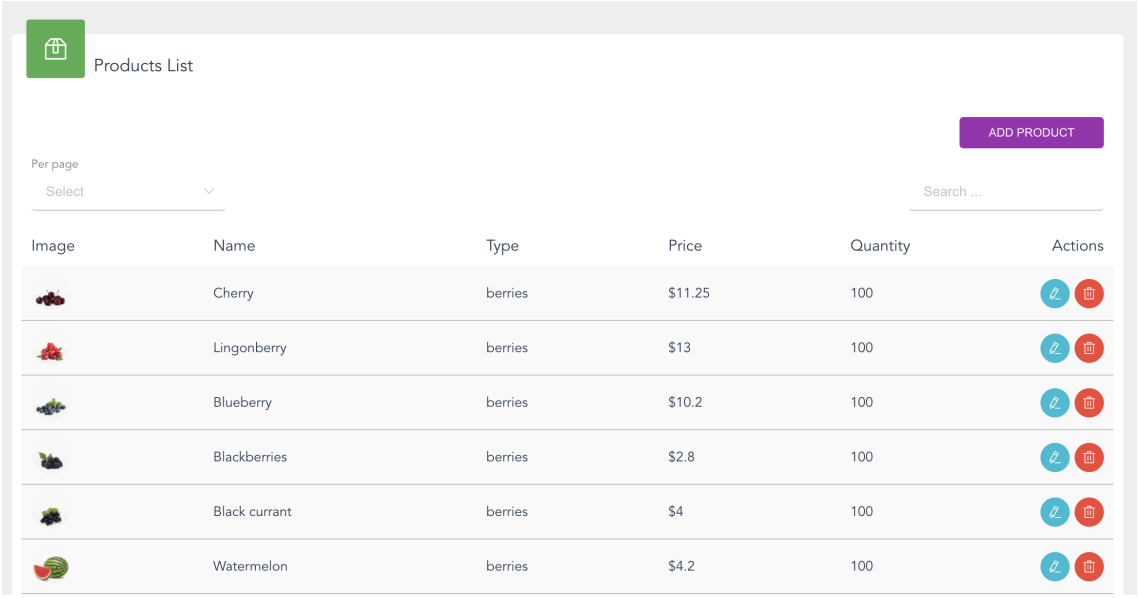


Figure 3.22: Product Management page Mock-up

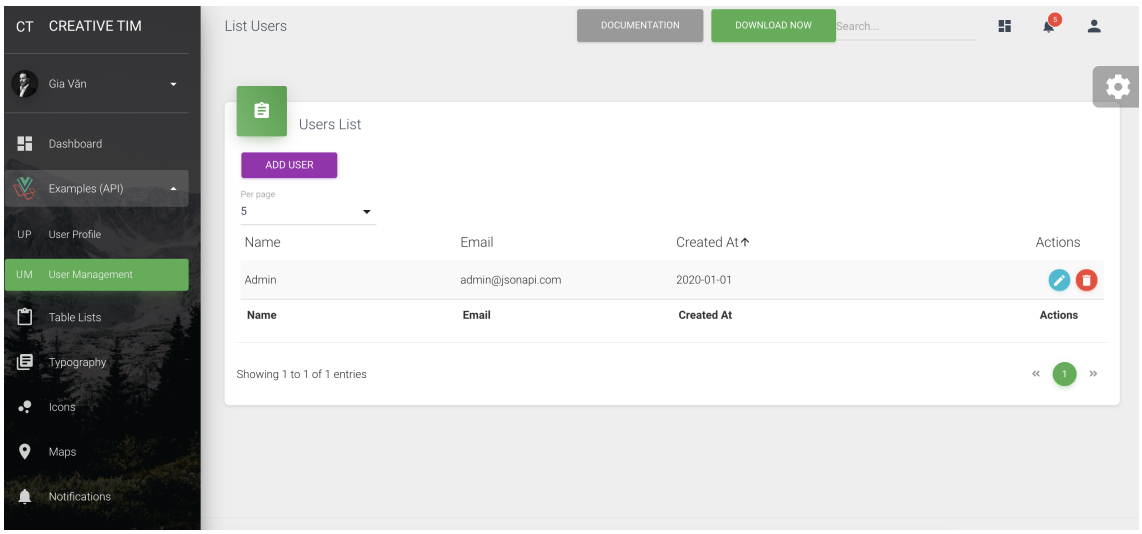


Figure 3.23: User Management page Mock-up

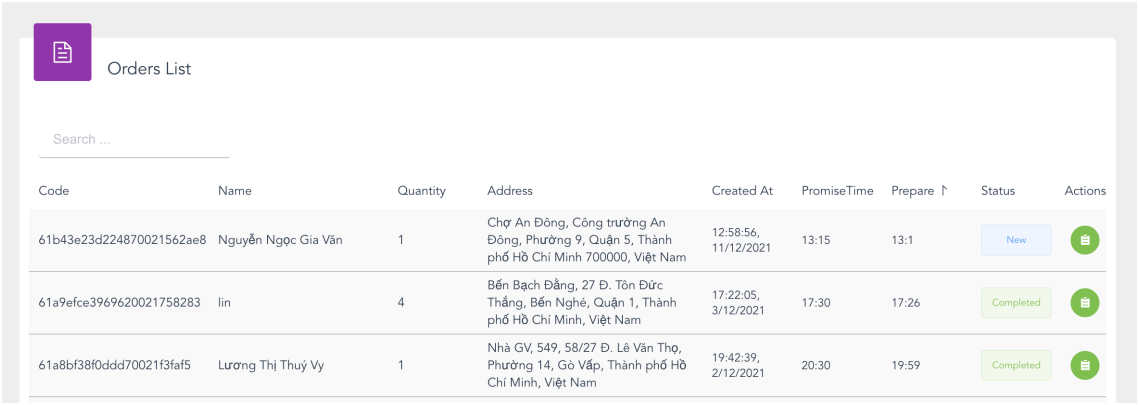


Figure 3.24: Order Management page Mock-up

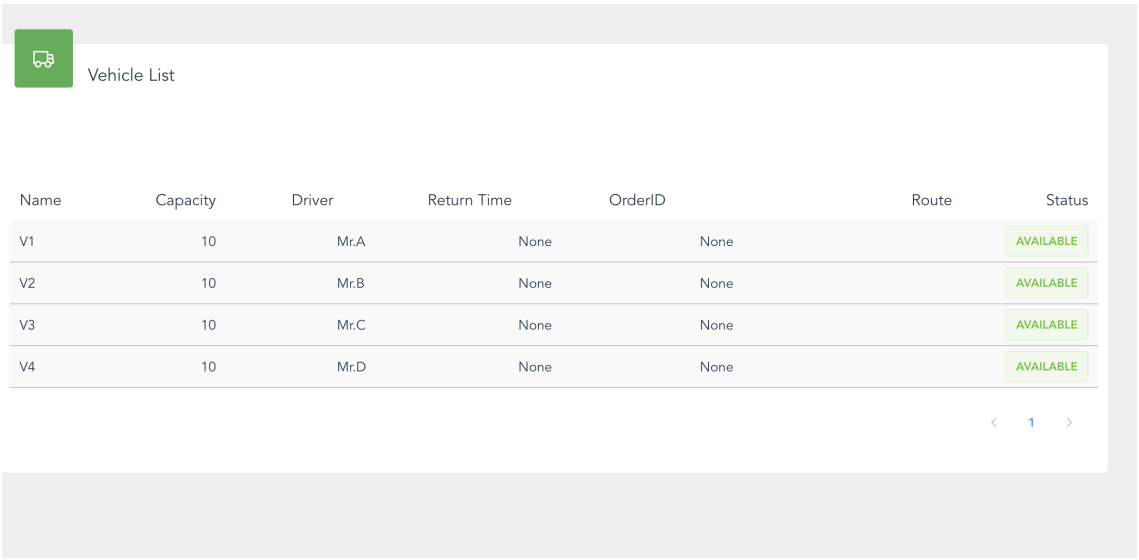


Figure 3.25: Vehicle Management page Mock-up

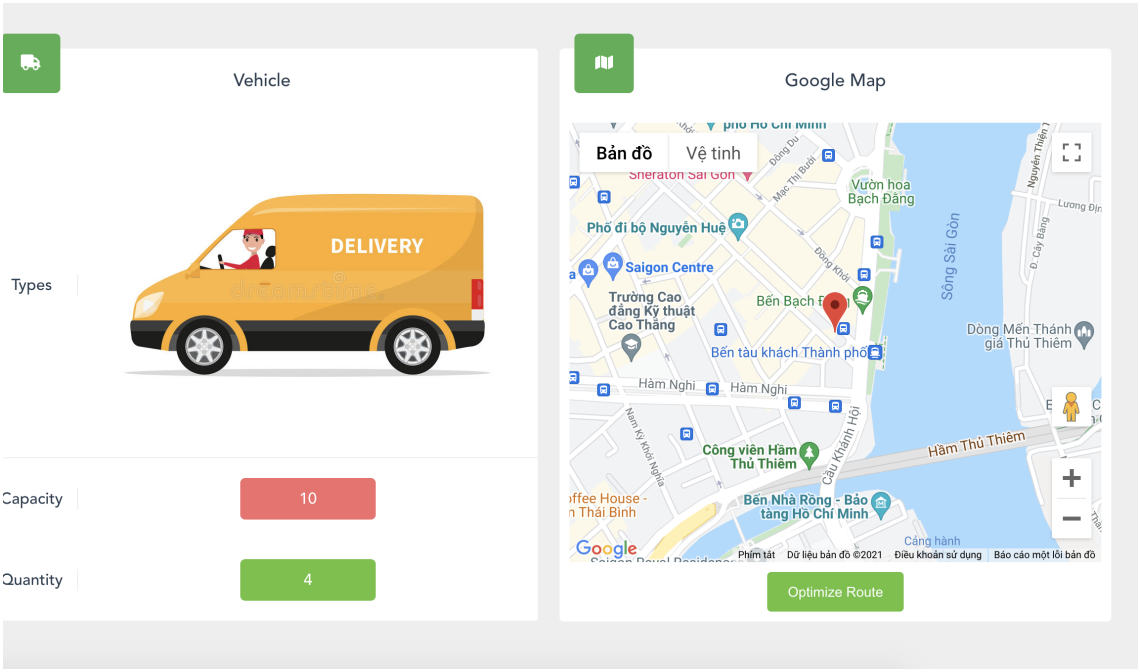


Figure 3.26: Route Optimization page Mock-up

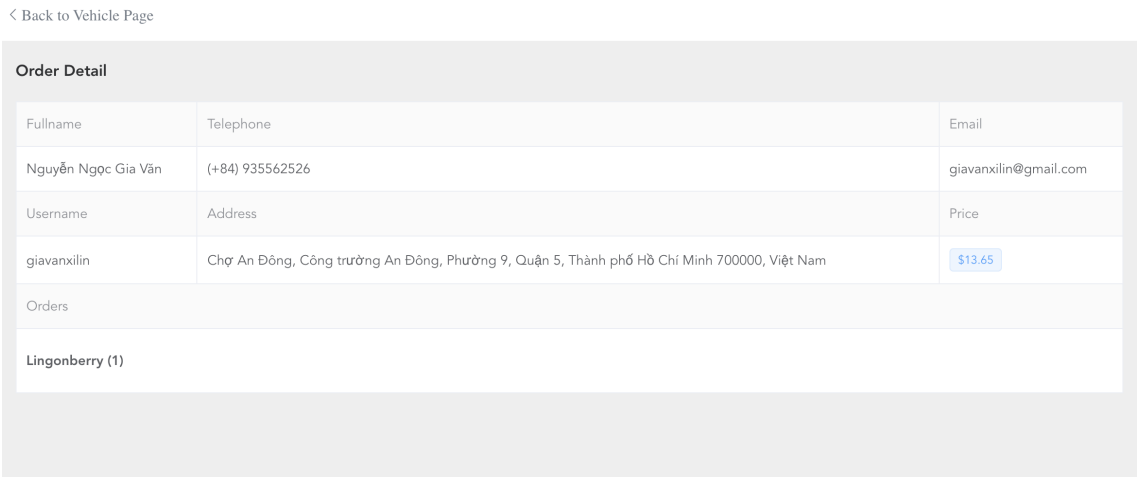


Figure 3.27: Admin Detail Order page Mock-up

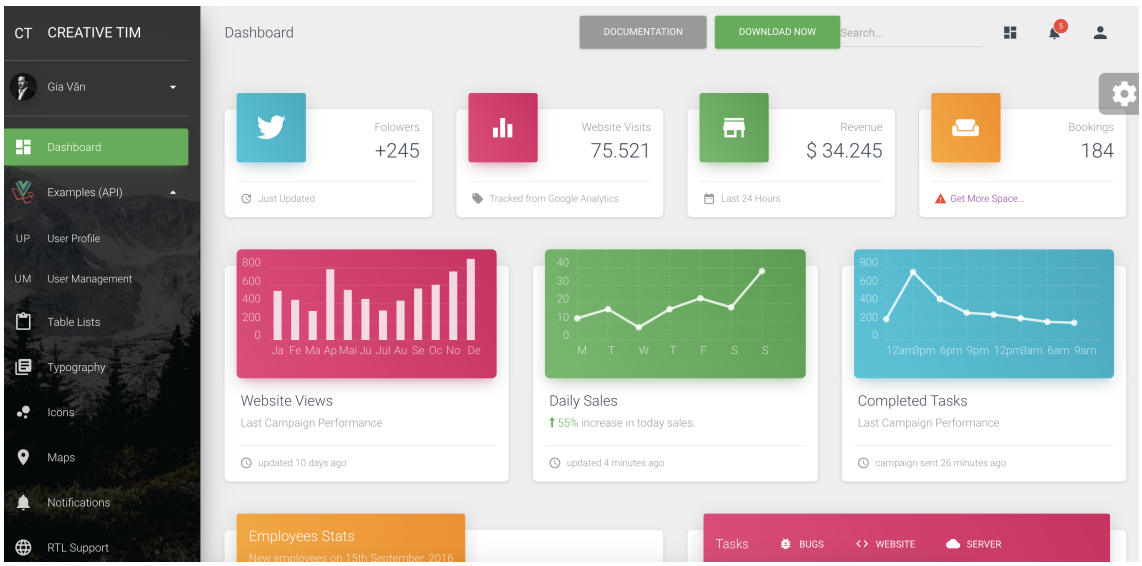


Figure 3.28: Admin Dashboard page Mock-up

Chapter 4

Deployment and evaluation

4.1 Deployment model

4.1.1 E-commerce website Front-end deployment

To deploy the front-end of the website, I will configure app to serve on express server locally.
Now install express and add a server.js file to serve Vue.js app

```
npm install express serve-static --save
```

After installing, I create a server.js file, and add following script to serve my app on express static server in server.js file:

```
const express = require('express')
const serveStatic = require('serve-static')
const path = require('path')

const app = express()

app.use('/', serveStatic(path.join(__dirname, '/dist')))

app.get(/.*/, function (req, res) {
  res.sendFile(path.join(__dirname, '/dist/index.html'))
})

const port = process.env.PORT || 8080
app.listen(port)
console.log(`app is listening on port: ${port}`)
```

After creating file server.js, start running the following compilation command: npm run build. My project will appear a folder called dist, because I just finished building the application, the dist folder is a compressed and shortened folder of my website
I edit my package.json file to tell heroku to serve app from my server.js file.

```
   "scripts": {
   "serve": "vue-cli-service serve",
   "build": "vue-cli-service build",
   "lint": "vue-cli-service lint",
   "start": "node server.js" <--- add this line under scripts block
 },
```

I create new app on heroku with the name ”giavan-store” to deploy my e-commerce website on heroku.
I install Heroku CLI using this link: https://devcenter.heroku.com/articles/heroku-cli and run this script on terminal:

```
$ heroku login
$ heroku git:clone -a giavan-store
$ git add .
$ git commit -am "deploy giavan-store"
$ git push heroku master
```

This script will automatically pull the newest code on my develop branch before to update my e-commerce website front-end.
My e-commerce website will be on server https://giavan-store.herokuapp.com

4.1.2 Admin website Front-end deployment

Similar to e-commerce website, I will configure app to serve on express server locally. Now install express and add a server.js file to serve Vue.js app

```
npm install express serve-static --save
```

After installing, I create a server.js file, and add following script to serve my app on express static server in server.js file:

```
const express = require('express')
const serveStatic = require('serve-static')
const path = require('path')

const app = express()

app.use('/', serveStatic(path.join(__dirname, '/dist')))

app.get(/.*/, function (req, res) {
  res.sendFile(path.join(__dirname, '/dist/index.html'))
})

const port = process.env.PORT || 9090
app.listen(port)
console.log(`app is listening on port: ${port}`)
```

After creating file server.js, start running the following compilation command: npm run build. My project will appear a folder called dist, because I just finished building the application, the dist folder is a compressed and shortened folder of my website
I edit my package.json file to tell heroku to serve app from my server.js file.

```
   "scripts": {
   "serve": "vue-cli-service serve",
   "build": "vue-cli-service build",
   "lint": "vue-cli-service lint",
   "start": "node server.js" <--- add this line under scripts block
 },
```

I create new app on heroku with the name "giavan-admin" to deploy my e-commerce website on heroku.
I install Heroku CLI using this link: https://devcenter.heroku.com/articles/heroku-cli and run this script on terminal:

```
$ heroku login
$ heroku git:clone -a giavan-admin
$ git add .
$ git commit -am "deploy giavan-admin"
$ git push heroku master
```

This script will automatically pull the newest code on my develop branch before to update my admin website front-end.
My admin website will be on server https://giavan-admin.herokuapp.com

4.1.3 Back-end deployment

Because my back-end server use express.js and there is small piece of code use python-shell library to run python code in `optimize_route.py` file below:

```
recordRoutes.get("/solving-route", async (req, res, next) => {
  await PythonShell.run('optimize_route.py', null, function (err, result) {
    if (err) console.log(err)
    let drop = result.shift()
    drop_nodes = drop.split(':')[1].split(' ').map(x => parseInt(x))
    drop_nodes.shift()
    doc = result.map(x => x.split(',').map(y => parseInt(y))).filter(x =>
        x.length != 2)
    doc.map(x => x.pop())
    doc.map(x => x.shift())
    res.send({route_legs: doc, drop_nodes: drop_nodes})
  });
});
```

So I will need a requirement file. Run this command:

```
pip freeze > requirements.txt
```

This will create a requirement.txt file for my application.
Inside the root of your folder, I create a Procfile (make sure that there is no extension in the file and "P" is capital). It is the Heroku file in which you will define the dynos setting like after uploading the app.

```
pipinstall: pip install -r requirements.txt
web: npm start
```

Heroku will start the Node.js web server and install python library in requirements.txt.
I have .env files for storing environment variables to keep API keys and other credentials in private mode. But these pose a problem while deploying to Heroku. So I have to configure Heroku so that its server can also use those hidden keys. Run the following commands on terminal:

```
heroku config:set <key=value>
```

Then I create new app on heroku with the name ”gv-grocery-api” on heroku and run this script on terminal:

```
$ heroku login
$ heroku git:clone -a gv-grocery-api
$ git add .
$ git commit -am "deploy gv-grocery-api"
$ git push heroku master
```

Now I can access my API through Heroku deployed website. My back-end server will be on server https://gv-grocery-api.herokuapp.com

4.2 Technical experiment

4.2.1 Testing environment

Before deploying, I use the local environment to test my application based on the test cases that I have designed. My local environment is run by using these following step:

- Start local Front end by running

```
npm run servr
```

- Start local Backend by running

```
npm start
```

Based on which port your front-end is running on, you can access your local website by using the URL: http://localhost:PORT. By using this way, I are able to reach each feature within ourselves when it is finished. I also conduct a small demo with some users asking them to use my website and give their feedback.

After I have deployed, I use the deployed site to test my feature again along with adding some cases which I can not test in the local development environment.

4.2.2 Functionality:

The **Appendix A** and **Appendix B** section contains my test cases table with the result of each case . Of 46 test cases that I have designed. All of my features work nearly perfectly except for some unusual cases and some cases I forgot to check. So overall the website is good enough to be used by users. However, I will constantly listen to users feedback for issue reports and further improvements.

4.3 Nontechnical experiment

4.3.1 Testing environment

To test the non-technical, I have to use the deployed website for the final result. TO test performance of the website, I will use tool: Google Lighthouse.

Lighthouse is an automated tool for improving the quality of your site. You give it a URL, and it provides a list of recommendations on how to improve page performance, make pages more accessible, adhere to best practices and more. You can run it from within Chrome DevTools, as a Chrome Extension, or even as a Node module, which is useful for continuous integration.

Lighthouse tool evaluate website based on 4 categories:

- Performance: Lighthouse analyzes how quickly a website or app loads and how quickly users can access or view the content
- Accessibility: Lighthouse's accessibility audits examine how well a website or app can be used by people with disabilities. This includes tests on important elements like buttons or links, to see whether they are sufficiently well described, or whether images have been assigned an alt-attribute so that the visual content can also be described by screen readers for visually impaired users.
- Best practice: Lighthouse analyzes whether HTTPS and HTTP/2 are used, checks to see whether resources come from secure sources and assesses the vulnerability of JavaScript libraries. Other best practices look at secure database connections and avoiding the use of non-secure commands, such as document.write(), or incorporating antiquated APIs.
- SEO: Lighthouse runs various tests to establish how well a website or app can be crawled by search engines and displayed in the search results.

4.3.2 Website performance

I test the performance of each page on my website to see their performance, accessibility, best practice score and SEO score:

Page	Performance	Accessibility	Best practice	SEO
Homepage	54	80	100	73
History Order	58	90	100	82
Detail Order	59	90	100	82
Confirm Order	58	78	100	74
Account Setting	56	88	100	82
Customer Profile	59	96	100	82

Table 4.1: Lighthouse performance result

My website has quite good scores on accessibility and SEO, best practice. However, the performance of my website is a little bit lower than remains, especially the loading speed is not good

But thanks to Lighthouse I have found some issue associated with the performance problem and also suggestion for them. These include

- A long cache lifetime can speed up repeat visits to your page.
- Consider reducing the time spent parsing, compiling and executing JS. Maybe delivering smaller JS payloads helps with this.

For other metrics, I can also improve the score by using the suggestion the Lighthouse provided to me.

Chapter 5

Conclusion

5.1 Accomplishment

In this project, I have been able to accomplish:

- Complete research and understand about algorithms for optimizing vehicle routing problem.
- Complete research of related frameworks in building the website and fully know how to use them.
- Understand the frameworks and technologies during the project.
- Survey the similar application and find solutions to the potential problems.
- Complete research how to use or-tool to solve VRP.
- Complete research how to use google maps Javascript API
- Complete research how to use telegram bot guide for driver.
- Deployed my website to the internet.
- Conduct tesing on my product.

5.2 Limitation

This application is built up from scratch, hence, there will be a lot of potential matters. As this idea is new and comes up by combining new techniques, new attractive entertainment.

- Furthermore, my assets are free on the internet then they have the limitation on resolution, capability, or usability.
- The websocket technique that I am using can crash sometimes and the load capacity of the websocket is not yet fully explored
- The UI/UX of admin website is not good enough to learn and use.
- Haven't tested the efficiency of the automatic ORDER COLLECTION algorithm because The number of people using the software is still small.
- Not clustering orders with close distance in automatic order collection.

5.3 Future work

The **Appendix C** section contains my timeline for this project. In the future, I will focus on these item.

- Improve the UI of the website
- Improve and design algorithm for optimization vehicle routing.
- Improve and design better algorithm for automatic order collection.
- Improve the loading speed of my website
- Improve the admin management features
- Research vehicle routing problem with multi-depot to apply into system with many branch stores.

Bibliography

[1] Stephanie Resendes. (2020, November 12). *Online Ordering Statistics Every Restaurateur Should Know in 2021.* Retrieved January 3, 2021, from *https://upserve.com/restaurant-insider/online-ordering-statistics/*

[2] Blake Morgan. (2020, Oct 19). *Statistics Showing The Lasting Impact Of COVID-19 On Consumers.*.Retrieved January 3, 2021, from *hhttps://www.forbes.com/sites/blakemorgan/2020/10/19/50-statistics- showing-the-lasting-impact-of-covid-19-on-consumers/?sh=5f93c02e261f*

[3] Minh Ngoc Nguyen. (2021, Mar 19) *Preference of retail trade channels for basic necessities among consumers in Vietnam in 2020, by category.* Retrieved Nov 5, 2021 from *https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer*

[4] Statista Research Department. (2021, Jul 5). *Preference of retail trade channels for basic necessities among consumers in Vietnam in 2020, by category.* Retrieved Nov 5, 2021 from *https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer*

[5] Statista Research Department. (2021, Jul 5). *Penetration rate of online food delivery platforms in Vietnam from 2017 to 2024.* Retrieved Nov 5, 2021 from *https://www.statista.com/forecasts/1230477/penetration-rate-online-food-platform-delivery-vietnam*

[6] Minh Ngoc Nguyen. (2021, Sep 22) *Average monthly web traffic of online grocery retailers in Vietnam from 4th quarter 2020 to 1st quarter 2021.* Retrieved Nov 5, 2021 from *https://www.statista.com/statistics/1263984/vietnam-monthly-web-traffic-of-ecommerce-grocers/#statisticContainer*

[7] Minh Ngoc Nguyen. (2021, Mar 19) *Willingness to buy necessities and fresh products online during COVID-19 Vietnam 2020.* Retrieved Nov 5, 2021 from *https://www.statista.com/statistics/1222566/vietnam-willingness-to-buy-necessities-and-fresh-products-online-during-covid-19/*

[8] Vien Thong. (2020, May 14) *Has Covid-19 subverted the traditional Vietnamese consumer?.* Retrieved Nov 5, 2021 from *https://e.vnexpress.net/news/business/industries/has-covid-19- subverted-the-traditional-vietnamese-consumer-4098697.html/*

[9] Josee Ng. (2020, Mar 20) *Online Grocery Stores Supermarkets In HCMC With Everything From Fresh Produce To Canned Goods.* Retrieved Aug 20, 2021 from *https://thesmartlocal.com/vietnam/online-groceries-hcmc/*

[10] Wikipedia. *WebSocket*. Retrieved Oct 27, 2020 from *https://en.wikipedia.org/wiki/WebSocket*

[11] Nick Craver. (2016 Feb 17). *Stack Overflow: The Architecture - 2016 Edition*. Retrieved Oct 27, 2020 from *https://nickcraver.com/blog/2016/02/17/stack-overflow-the-architecture-2016-edition/*

[12] Lucas White. (2020 Nov 6). *What Are The Reasons To Learn Express.Js In 2021?*. Retrieved Mar 1, 2021 from *https://codersera.com/blog/learn-express-js/*

[13] Stack Overflow. (2019). *Developer Survey Results 2019*. Retrieved Oct 28, 2020 from *https://insights.stackoverflow.com/survey/2019*

[14] MongoDB *What is MongoDB Atlas*. Retrieved Sep 15, 2021 from *https://docs.atlas.mongodb.com*

[15] Google Maps Platform. Retrieved Mar 5, 2021 from *https://developers.google.com/maps/documentation/javascript/overview*

[16] Google OR-tools. Retrieved March 15, 2021 *https://developers.google.com/optimization*

[17] ElementUI documentation. Retrieved Mar 5, 2021 from *https://element.eleme.io/#/en-US*

[18] Python-shell documentation. Documentation. Retrieved Sep 12, 2021 from *https://www.npmjs.com/package/python-shell*

[19] Telegram bot APIS documentation. Documentation. Retrieved Mar 5, 2021 from *https://core.telegram.org/bots*

[20] Colin Fallon (Mar 25, 2017). *Uni-directional data flow with Vuex*. Retrieved Nov 10, 2021 from *https://medium.com/@softwarecf/uni-directional-data-flow-with-vuex-4f31d7ac8c83*

[21] G. B. Dantzig, J. H. Ramser. (1959, Oct 1). *The Truck Dispatching Problem*. Retrieved June 19, 2021 from *https://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.80*

[22] Sajaykumar J (2020, April 24). *Vehicle Routing Problem and its variants*. Retrieved Mar 15, 2021 from *https://www.linkedin.com/pulse/vehicle-routing-problem-its-variants-sajaykumar-j*

[23] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi (2006, Aug 1). *An exact algorithm for the vehicle routing problembased on the set partitioning formulation with additional cuts*. Retrieved Sep 13, 2021 from *https://link.springer.com/article/10.1007/s10107-007-0178-5*

[24] G. Clarke, J. W. Wright (1964, Aug 1). *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*. Retrieved Sep 13, 2021 from *https://pubsonline.informs.org/doi/abs/10.1287/opre.12.4.568*

[25] Renaud, Jacques Boctor, Fayez F (2002, Aug). *A sweep-based algorithm for the fleet size and mix vehicle routing problem.*. Retrieved Dec 21, 2020 from *https://ideas.repec.org/a/eee/ejores/v140y2002i3p618-628.html/*

Appendices

Appendix A

Test case table for e-commerce website

This contains the test case table which I have design and tested for e-commerce website.

Id	Test case description	Expected result	Actual result	Status	Tester
1	Login customer	Able to be in homepage	Able to be in homepage	Pass	Nguyen Ngoc Gia Van
2	Login with invalid username/password	See warning incorrect username/password	See warning incorrect username/password	Pass	Nguyen Ngoc Gia Van
3	Valid register customer	Register successful, customer is redirected to setting page	Register successful, customer is redirected to setting page	Pass	Nguyen Ngoc Gia Va
4	Register customer with invalid user/password/email	See warning in invalid field	See warning in invalid field	Pass	Nguyen Ngoc Gia Van
5	Register customer with not match password	See warning not match password	See warning not match password	Pass	Nguyen Ngoc Gia Van
6	Add to cart	Add item to shipping cart and system display success message	Add item to shipping cart and system display success message	Pass	Nguyen Ngoc Gia Van
7	Update quantity items in cart	Number items in cart will change when customer update	Number items in cart will change when customer update	Pass	Nguyen Ngoc Gia Van
8	Remove item in cart	Customer see item will be removed	Customer see item will be removed	Pass	Nguyen Ngoc Gia Van
9	Remove all items in cart	Customer see no item in shipping cart and can not checkout	Customer see no item in shipping cart and can not checkout	Pass	Nguyen Ngoc Gia Van

10	Select product type	Customer see type of product they want	Customer see type of product they want	Pass	Nguyen Quang Manh
11	Checkout error when customer haven't logged yet	A error message appear to notifiy customer and navigae to login page.	A error message appear to notifiy customer and navigate to login page	Pass	Nguyen Ngoc Gia Van
12	Checkout and move confirm order page	System navigates to confirm order page	System navigates to confirm order page	Pass	Nguyen Ngoc Gia Van
13	Confirm order error	An error message appear to notify customer if customer doesn't enter shipping address	An error messgae appear to notify customer if customer doesn't enter shipping address	Pass	Nguyen Ngoc Gia Van
14	Confirm order error	An error message appear to notify customer if shipping address is not in Ho Chi Minh area	An error messgae appear to notify customer if shipping address is not in Ho Chi Minh area	Pass	Nguyen Ngoc Gia Van
15	Disable Promise Time	Promise time is disable if customer doesn't enter shipping address before	Promise time is disable if customer doesn't enter shipping address before	Pass	Nguyen Quang Manh
16	Change payment method	Customer sees payment method will be changed	Customer sees payment method will be changed	Pass	Nguyen Ngoc Gia Van
17	Confirm Order Success	Customer sees success message	Customer sees success message	Pass	Nguyen Ngoc Gia Van
18	Order history	User see list of orders	User see list of orders	Pass	Nguyen Ngoc Gia Van
19	Order detail	User see detail of order	User see detail of order	Pass	Nguyen Ngoc Gia Van
20	Change account setting	Can see message successfully update your password. User can now login with the new passsword	Can see message successfully update your password. User can now login with the new passsword	Pass	Nguyen Ngoc Gia Van
21	Change account setting with invalid old password	Can see message show cannot update new password	Can see message show cannot update new password	Pass	Nguyen Ngoc Gia Van
22	Change profile setting	Can see message show update successfully	Can see message show update successfully	Pass	Nguyen Ngoc Gia Van

23	Log out	Can see message log out successfully	Can see message log out successfully	Pass	Nguyen Ngoc Gia Van

Table A.1: Test cases table for e-commerce website

Appendix B

Test case table for admin website

This contains the test case table which I have design and tested for admin website.

Id	Test case description	Expected result	Actual result	Status	Tester
1	Login user	Able to be in dashboard	Able to be in dashboard	Pass	Nguyen Ngoc Gia Van
2	Login with invalid username/password	See warning incorrect username/password	See warning incorrect username/password	Pass	Nguyen Ngoc Gia Van
3	Add Product Invalid	See warning invalid field	See warning invalid field	Pass	Nguyen Ngoc Gia Van
4	Add Product valid	A success message appear	A success message appear	Pass	Nguyen Ngoc Gia Van
5	Remove product	See warning if user remove product	See warning if user remove product	Pass	Nguyen Ngoc Gia Van
6	Remove product	See success message if user confirm remove product	See success message if user confirm remove product	Pass	Nguyen Ngoc Gia Van
7	Update Product Invalid	See warning invalid field	See warning invalid field	Pass	Nguyen Ngoc Gia Van
8	Notify number of new orders	User sees notification new order if customer place order	User sees notification new order if customer place order	Pass	Nguyen Ngoc Gia Van
9	User sees detail order	Detail of order will be display if user click detail icon	Detail of order will be display if user click detail icon	Pass	Nguyen Ngoc Gia Van
10	User sees status change progressing	At the time equal to the preparation time, status order will change progressing status	At the time equal to the preparation time, status order will change progressing status	Pass	Nguyen Ngoc Gia Van

11	User sees status change delivering	About 2 minutes after changing to progressing, status will change to delivering	About 2 minutes after changing to progressing, status will change to delivering	Pass	Nguyen Ngoc Gia Van
12	User sees status change completed	When the vehicle back to depot, status will change into completed	When the vehicle back to depot, status will change into completed	Pass	Nguyen Ngoc Gia Van
13	Automatic order collection	User can see automatic order collection	User can see automatic order collection	Pass	Nguyen Ngoc Gia Van
14	Add new user Invalid	See warning incorrect field	See warning incorrect field	Pass	Nguyen Ngoc Gia Van
15	Update user Invalid	See warning incorrect field	See warning incorrect field	Pass	Nguyen Ngoc Gia Van
16	Update user Invalid	See warning incorrect field	See warning incorrect field	Pass	Nguyen Ngoc Gia Van
17	Notify vehicle active	When order status change delivering, system will notify vehicle active	When order status change delivering, system will notify vehicle active	Pass	Nguyen Ngoc Gia Van
18	Vehicle status	Can see change of vehicle status	Can see change of vehicle status	Pass	Nguyen Ngoc Gia Van
19	Optimzation Route	Can see optimization of route on google map	Can see optimization of route on google map	Pass	Nguyen Ngoc Gia Van
20	Notify telegram bot	When order status change delivering, notify telegram bot	When order status change delivering, notify telegram bot	Pass	Nguyen Ngoc Gia Van
21	Instruction panel	Can see instruction panel of optimize route	Can see instruction panel of optimize route	Pass	Nguyen Ngoc Gia Van
22	Change configuration	A success message appear when user change configuration	A success message appear when user change configuration	Pass	Nguyen Ngoc Gia Van
23	Log out	Can see message log out successfully	Can see message log out successfully	Pass	Nguyen Ngoc Gia Van

Table B.1: Test cases table for admin website

Appendix C

Progress table

This contains the progress table of this project

Stage	Task	Duration
1	Design use-cases	1/3/2021 - 20/3/2021
2	Take Surveys of related Application	21/3/2021 - 28/3/2021
3	Design Architecture	29/3/2021 - 11/4/2021
4	Select Libraries and Frameworks + Find solution to potential problems	12/4/2021 - 5/5/2021
5	Research algorithm solving vehicle problem	6/5/2021 - 21/5/2021
6	Research tools for solving vehicle problem	22/5/2021 - 5/6/2021
7	Research google maps Javascript API	6/6/2021 - 20/6/2021
8	Select template and design mockup for website	21/6/2021 - 21/7/2021
9	Implement authentication in back-end and front-end	22/7/2021 - 2/8/2021
10	Implement e-commerce website	3/8/2021 - 3/9/2021
11	Implement admin website	4/9/2021 - 15/9/2021
12	Integrate OR-tool and Google Map to website	16/9/2021 - 10/10/2021
13	Design and integrate automatic order collection in my app	11/10/2021 - 1/11/2021
15	Fixing bugs in backend and frontend, design testing	2/11/2021 - 10/11/2021
15	Fixing bugs. Deploy front-end and back-end	11/11/2021 - 18/11/2021
16	Fix bugs for whole website	19/11/2021 - 26/11/2021
17	Responsive mobile and continue testing	27/11/2021 - 5/12/2021
20	Receive feedback and improve website	6/12/2021 - future

Table C.1: Plan for thesis table