



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tuan Nguyen

E-COMMERCE WEB SHOP

ABSTRACT

Author	Tuan Nguyen
Title	E-Commerce WebShop
Year	2021
Language	English
Pages	75
Name of Supervisor	Anna-Kaisa Saari

The research background related to the advantages of digitalization and E-commerce supplying to end customers in the integration of technology era.

The theoretical review introduces the concept of e-commerce, introducing Spring, its outstanding features and advantages in creating applications that require modularization and high reusability. At the same time, some technologies such as MySQL, Spring Boot, ReactJS, Bootstrap are currently being used by software companies to create an enterprise web application with Spring Framework.

After studying, the results will be applied to build an application for the purpose of illustrating the presented theory. A web-based commerce management application will be built. The application will be designed into two main modules, the client module and the server module. In this thesis, the concentration will emphasize the design and construction of the client module.

By implementing and testing the web-based, the research study will provide the understanding how the communication mechanism between client and server in modern RESTful service-oriented web model works and how to communication between components of a system.

Keywords E-commerce web-based, technology and architecture

CONTENTS

ABSTRACT

1	INTRODUCTION	6
1.1	Research Background	6
1.2	E-Commerce	6
1.2.1	Four Types of E-commerce.....	7
1.2.2	Advantages of E-commerce Website	8
1.2.3	Drawbacks of E-commerce Website	9
1.3	PayPal	9
2	REQUIREMENTS	10
2.1	Requirements for end-user perspective are the following:	10
2.2	Requirements for admin perspective are the following:	10
3	USED TECHNOLOGY	12
3.1	Front-end Side:.....	12
3.1.1	HTML5	12
3.1.2	Creating Responsive User Interface with Bootstrap	12
3.1.3	JavaScript/ES6	13
3.1.4	ReactJs Framework	13
3.2	Back-end side.....	14
3.2.1	Spring Framework and Spring Boot	15
3.2.2	RESTful API	16
3.2.3	MySQL.....	17
3.2.4	NGINX.....	18
4	SYSTEM ARCHITECTURE MODEL.....	20
4.1	Back-end Side	20
4.2	Front-end Side.....	20
4.3	Data Flow Model of the Web Page.....	23
5	DESIGN	25
5.1	Class Diagram.....	25
5.2	Sequence Diagram	26
5.2.1	Login	26
5.2.2	Logout	27

5.2.3	Changing Password.....	28
5.2.4	Add Product	28
5.2.5	Update Product.....	29
5.2.6	Add Categories.....	30
5.2.7	Update Categories	30
5.2.8	Add User	31
5.2.9	Assign Role.	32
5.2.10	Chat Room	32
5.2.11	Order	33
6	MODEL	34
6.1	Functional Hierarchy Chart.....	34
6.2	Use Case Diagram.....	34
6.3	Use Case Specification	35
6.3.1	Login Use Case Specification	35
6.3.2	Logout Use Case Specification	36
6.3.3	Changing Password Use Case Specification.....	37
6.3.4	Product Management Use Case Specification	37
6.3.5	Categories Management Use Case Specification.....	40
6.3.6	User Management Use Case Specification	42
7	DESIGN THE MOCKUP.....	45
7.1	List of Pages.....	45
7.2	Detailed Description of Pages.....	46
7.2.1	Login	46
7.2.2	Admin Page.....	47
7.2.3	Register Page.....	47
7.2.4	Add/ Update Product Page.....	48
7.2.5	Categories Admin Page.....	49
7.2.6	Staff Page	49
7.2.7	Room Page	50
7.2.8	Categories List Page.....	51
7.2.9	Best Deal Page	52
7.2.10	Change Password Page	52

7.2.11	Checkout Page.....	53
7.2.12	Compare Page	55
7.2.13	Wish Page.....	56
7.2.14	Message Form	56
8	IMPLEMENTATION	57
8.1	Deployment.....	57
8.1.1	Back-end and Front-end Side.....	57
8.2	Implementation	58
9	TESTING	69
10	CONCLUSION	73
11	REFERENCES	74

1 INTRODUCTION

1.1 Research Background

Nowadays, the world of e-commerce is developing very strongly. Digitalization helps people to save more costs transported through intermediaries, transaction costs and it significantly helps to save time to invest in other activities. Moreover, e-commerce also allows people to search for many different purposes, provide information according to human needs and preferences. Customers can sit at home and buy everything they want using online sales websites. Open-source technologies are very important in building sales websites efficiently with low costs. Usually open-source technologies are so widely used that there is excellent support available on the Internet. The provided support enables building user-friendly sales websites quickly. Therefore, the topic: “Building an E-commerce Website” was selected for this thesis. An e-commerce website is a simple but powerful enough system which allows rapid construction of sales applications on the Internet.

1.2 E-Commerce

Over the past decade, consumers have continuously changed the way they want to shop, especially in terms of shopping behavior on e-commerce channels and websites, forcing businesses to find ways to adapt to this trend. With the support of newest technology, it is easier for consumers to find, compare and purchase from online websites, e-commerce markets, mobile applications, and physical stores and social sites, rather than following traditional commercial models.

According to research by Get App (Factory 2021), the leading review software in the world, they have given the following Figures about consumer needs and habits of customers:

- Customers like to research and buy products online with 53.1%.
- Customers prefer to research online and buy offline with 28.9%.

- Only 18% of customers say they prefer to going to stores.



Figure 1. E-commerce Research (Factory 2021).

1.2.1 Four Types of E-commerce

B2C: BUSINESS TO CONSUMERS

B2C e-commerce includes transactions made between businesses and consumers. This is one of the most widely used sales models in the e-commerce landscape. For example, when customers buy shoes online from the manufacturer Nike, this is a business-to-consumer transaction.

B2B: BUSINESS TO BUSINESS

B2B e-commerce involves commercial activities carried out between businesses, such as between manufacturers and dealers or retailers. Typically, business-to-business sales will often focus on raw materials, or packaged products.

C2C: CONSUMER TO CONSUMER

One of the earliest established e-commerce business models is the C2C e-commerce business model (The-Future-Of Customer Engagement and Experience 2021; Bloom-idea, 2021). This includes customer-to-customer relationships, for example on Shopee, Amazon, and Lazada, Alibaba.

C2B: CONSUMER TO ENTERPRISE

C2B is a business model that reverses the traditional e-commerce model. C2B means the individual consumer produces the product or service, the business will be the one

who buys it.(The-Future-Of Customer Engagement and Experience 2021; Bloomidea, 2021).

1.2.2 Advantages of E-commerce Website

- **Increased profits**

E-commerce websites biggest advantage is that E-commerce is not limited in time and space like a traditional form of sales. Via e-commerce websites businesses can actively look for customers to their products and services. In addition, in order to increase profits, businesses must provide products with guaranteed quality, in accordance with the needs of users, reasonable prices with enthusiastic support.

- **Cost savings**

Cost savings are the benefits that attract many businesses to invest in owning an e-commerce website. Because businesses do not need to spend a large amount of money to hire and train staff but it still gets high profits if they know how to combine with some online marketing strategies.

- **Increased interactivity with customers**

When customers visit an e-commerce website, it is easier for them to update prices, product information and services. Besides, any time the customer needs advice, the customer care team will give advice on time, customer does not have to wait for support. This will show the respect, professionalism and reputation of the company in the heart of customers, helping to increase interaction with customers.

- **Improved competitiveness with competitors**

The competition in the technology sales market is getting fiercer. If businesses know how to take advantage of an own e-commerce websites with a unique and easy-to-see interface, it will help to earn money and to attract more customers.

- **Brand promotion**

In the digital age, customers can use social media, so it will be very convenient to promote the brand in the international market. Therefore, the design of an e-commerce websites will help businesses easily reach potential customers more quickly and effectively (The-Future-Of Customer Engagement and Experience 2021).

1.2.3 Drawbacks of E-commerce Website

Sometimes E-commerce cannot create a relationship between brands and buyers like physical stores. This lack of communication can be a disadvantage for many types of services and products, such as interior design or jewelry business. Also, from the security point of view, there have been many attacks of security breach in the world. Customer or credit information and identities can be stolen, . (The-Future-Of Customer Engagement and Experience 2021).

1.3 PayPal

PayPal is understood to be an intermediary service used to make international payments and money transfers over the Internet. With PayPal, customers can pay when they shop online.

PayPal can be seen as an electronic wallet or similar to Internet Banking of banks: customers can transfer, withdraw to another PayPal account, pay for online purchases if the place of sale supports PayPal, or receive international payment.

PayPal was founded in 1998, is headquartered in San Jose, California, USA. PayPal was founded by Peter Thiel. PayPal works on e-commerce via the Internet, and it charges fees when money is transfer or withdraw. (Henderson, R.2021).

2 REQUIREMENTS

Requirements for an e-commerce website to meet the needs of customers in terms of support as well as ensure the basic criteria of buying and selling products on an e-commerce site. The requirements of this website are presents next.

2.1 Requirements for end-user perspective are the following:

- Product information :
 - o Product is always updated, introducing the latest generation.
 - o Information about a product must be detailed so that customers can grasp information about the product they choose. Especially the items that many customers are interested in.
 - o Allows customers to search quickly and accurately according to many criteria.
- The interface is easy to use and highly aesthetically pleasing.
- Allow customers to register for membership and ensure confidentiality of information
- View and change account information.
- Payment methods must be accurate.

2.2 Requirements for admin perspective are the following:

- General management: related information of employees, customers, items.
- Updating items information online.
- Easily update and regularly change pictures, details, prices of the items for sale regularly changed.
- Manage online orders.
- Admin page store activities associated with customers of the store. All the activities related to customers and customers can be done remotely, regardless of geographical location.
- Statistics of which items are sold out.

- Synthesizing storing feedback from customers so that customers can respond to customers quickly and accurately.
- Products: Can add, edit and delete information, categories
- Receive and respond to customer requests.

3 USED TECHNOLOGY

3.1 Front-end Side:

The front end (also known as the client-side) is all about what a user sees when visit a website, including design and languages like HTML or CSS.

3.1.1 HTML5

HTML (Hypertext Markup Language) is a platform similar to Microsoft Word that helps users to design websites elements, structure pages, categories or design applications. The main function of HTML platform is to create websites layout and format (Tutorialspoint 2021).

3.1.2 Creating Responsive User Interface with Bootstrap

Bootstrap is the front-end framework, which is a free collection of tools for creating websites and web applications. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, and other interface elements, as well as extended JavaScript options (W3Schools 2021).

Bootstrap defines CSS classes available to help websites designers to save a lot of time. Bootstrap libraries have code ready to apply to websites without having to spend too much time writing it ourselves. With Bootstrap, developing websites interfaces to fit multiple devices becomes easier than ever. Bootstrap is the trend of developing websites interface which is very popular today. The websites is self-scalable to be compatible with all devices, from mobile phones to tablets, laptops, desktops. Another aspect is that responsive web design makes the websites a great experience for users across different screen sizes and devices (W3Schools 2021).

Responsive web design is very important since the devices and screen sizes that will be used to access websites cannot be predicted. A page that can perform well regardless of variation will provide a better and more consistent user experience than a page designed for a particular type of device and screen size (W3Schools 2021).

3.1.3 JavaScript/ES6

JavaScript is a programming language with the ability to bring life to websites design. The JavaScript scripting language is based on a built-in development object, or self-defined. JavaScript is the convenient and efficient because it is a dynamic programming language. Besides, JavaScript works based on HTML and CSS to a create dynamic content on the website. Moreover, JavaScript is mainly used in client-side programming without installing environment settings whereas with Java or Python environment settings would have to be installed. Therefore, JavaScript programming language is trusted more and more and widely applied on effective websites (Developer Mozilla 2021).

The use of the JavaScript language can be applied to all different browsers, now commonly used as Chrome, or Firefox. Moreover, JavaScript is also an effective programming language, fully supported on mobile device browsers. Therefore, the use is diversified and can well meet many different needs and requirements of users (Developer Mozilla 2021).

JavaScript is simple, easy to learn and use. With the syntax quite similar to English, the use of JavaScript becomes much easier and more accessible. Through the used DOM model, it provides many useful features, which are pre-written to better respond to different needs and requirements of the user. With many useful features that the JavaScript programming language brings, it becomes easier to be able to develop scripts to solve certain requirements or purposes (Developer Mozilla 2021).

3.1.4 ReactJs Framework

ReactJS is a library which contains a lot of open source JavaScript and it is maintained by Facebook and a community of individual developers and companies. The purpose of ReactJS was to create compelling websites applications with high speed and efficiency with minimal coding. The key purpose of ReactJS is that every website, when using ReactJS, has to run smoothly, fast and is highly scalable and simple to implement.

In general, ReactJS allows functional developers to separate the user interface from a complex way and turn it into simpler parts which means that rendering data is not only done at a server location but also at the client using ReactJS.

The basic components of React are called components. The React components are easier to write because using JSX, JavaScript's optional syntax extension. JSX allows HTML to be combined with JavaScript. JSX is a blend of JavaScript and HTML. ReactJS clarifies the whole process of writing websites structure. In addition, the extension also makes rendering more options easier. JSX may not be the most popular syntax extension, ReactJS can be effective for developing special components or high- volume applications (Facebook Inc 2021).

Redux Thunk

Redux is a library that acts as a container of states and helps manage application data flow. Redux was introduced back in 2005 at the React Europe conference.

Redux Thunk is a middleware that allows a user to write Actions that return a function instead of a plain JavaScript object by delaying the action to be sent to a reducer. Redux Thunk is used to handle complex asynchronous logic that requires access to the Store or simply retrieving data like an Ajax request.

In Redux, actions can also be sent from other parts of the system. This action is sent directly to a Store. This is the biggest difference between Redux and Flux. The logic that determines how directly the data changes in functions is called Reducers. When the Store receives an Action, it asks the Reducers the latest State information to be updated by sending the current state and Action information. Reducers need to return a new State with the format Immutable. The Store will perform its internal State update. The last step, React Component will do the render again (Reduxjs 2021).

3.2 Back-end side

Backend programming is the handling of all complex business, hidden behind a website, application, system to help the system operate smoothly. The backend usually consists of three parts: the server, the application, and the database. Any product has two places where the code works to make things run smoothly: the client side and the server side.

3.2.1 Spring Framework and Spring Boot

Spring is a framework that was created to help developers build systems and run applications on the JVM conveniently, simply and quickly. Spring Framework is open source developed and used widely in developing websites. Spring framework is a collection of many different small projects, such as Spring MVC (used to build web-based applications), Spring Data and Spring Boot.

Spring Boot is the fastest way to create a standalone REST service. Spring Boot simplifies configuration, in particular, Spring Boot configures all by itself by providing specific behaviors. Spring Boot simplifies deploying, packing application into a jar package and it can be easily integrated into web containers. Earlier the initialization of a Spring project was hard when dependencies needed to be declared in pom.xml file using XML or other complex annotations. With Spring Boot Spring application creation is quick and the configuration is also simpler.

The benefits of Spring Boot are:

- Easy to create Spring-based applications with Java or Groovy.
- Spring Boot does not write a huge of standard Code, Annotations, and XML configuration.
- With Spring Boot it is simple to communicate applications with Spring ecosystems, for example, Spring JDBC, Spring ORM, Spring Data, Spring Security and so forth
- Spring Boot gives Embedded HTTP like Tomcat to create and test web applications rapidly and without any problem.
- Spring Boot gives a CLI (Command Line Interface) device to create and test Spring Boot (Java or Groovy) applications from order brief effectively and rapidly.
- Spring Boot gives a great deal of modules to create and test Spring Boot applications rapidly utilizing build instruments like Maven.
- Based on Annotations to make beans rather than XML.
- Tomcat can be installed in the JAR fabricate record and can be run anyplace java environment(Spring Boot. 2021)

3.2.2 RESTful API

RESTful API is a standard used in designing API for web applications (designing Web services) to facilitate the management of resources. RESTful focuses on system resources, such as text files, images, audio, video or dynamic data, including resource states which are formatted and transmitted over HTTP. Figure 2 below shows the REST API (Maxoffsky 2021).

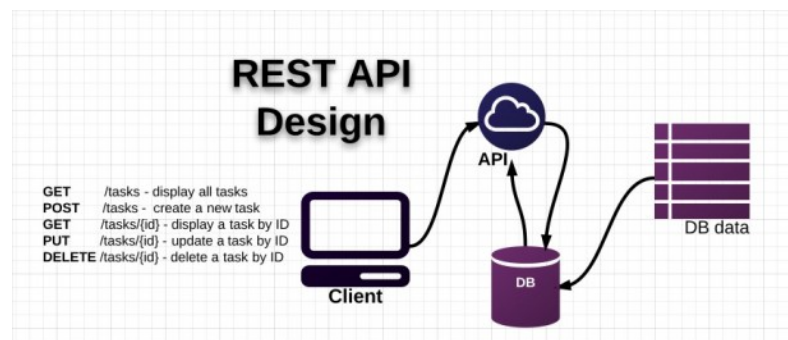


Figure 2. Rest API (Maxoffsky 2021).

APIs are methods and protocols to connect with other libraries and applications. API stands for Application Programming Interface. Moreover, the API provides the ability to provide access to a set of frequently used functions and exchange data between applications. The API can return the data for application in common data types such as JSON or XML.

REST (Representational State Transfer) is a standard of web service. RESTful can use JSON, XML, plain text, html as the structure of the returned data, with clear convention on how to write a URL and HTTP method. But RESTful does not provide information protection mechanism in endpoints like SOAP, instead it uses Json Web Token in combination with RESTful to solve the problem. Besides, REST sends an HTTP request like GET, POST, DELETE to a URL to process the data. RESTful does not specify application code logic and is not limited to an application programming language. Any language or structure can be utilized to plan a RESTful API. Figure 3 below shows the flowing of REST API (Tram 2021).

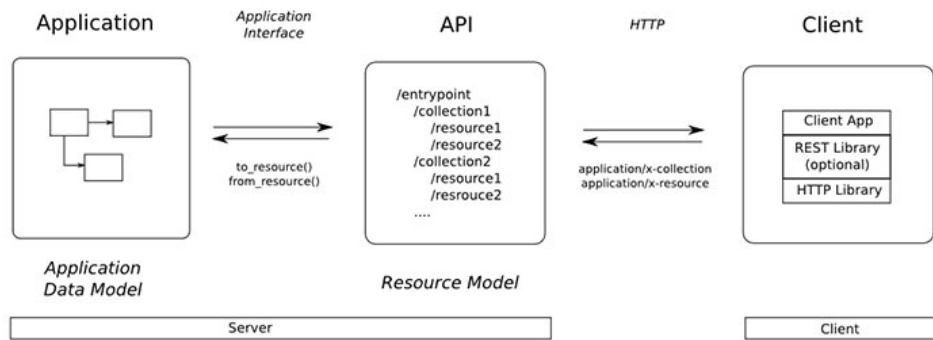


Figure 3. Flowing of REST API (Tram.2021).

3.2.3 MySQL

MySQL is an open source database management system (referred to as RDBMS) that operates in a client-server model. RDBMS stands for Relational Database Management System. MySQL is combined with Apache or Php MySQL. MySQL manages data through databases. Each database can have many relational tables containing data. MySQL also has the same access and code as the SQL language.

As shown in Figure 5, How MySQL running.

- MySQL creates tables to store data, and also defines the relationships between those tables.
- Client sends SQL requests with a special command on MySQL.
- The application on the server receives and responds to the information and returns the results to the client (Herawan Dwika, P.2021).

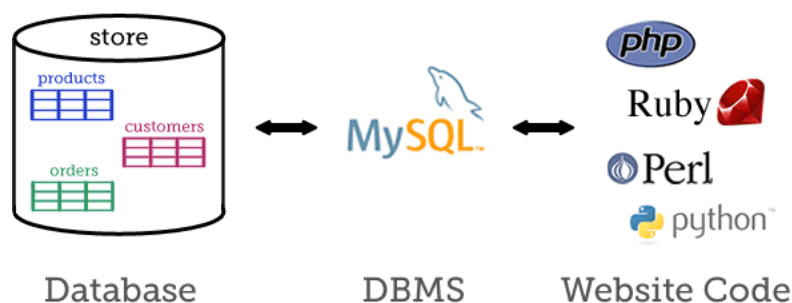


Figure 4. MySQL Flowing (Elated, 2021).

Benefits of MySQL:

- Ease of Use: MySQL is simple, easy to use. In addition, this software can operate on many operating systems to provide many powerful utility functions.
- High security: MySQL has a lot of security features, including high-level security types.
- Multi-function: MySQL provides many features that any relational database management system should expect.
- Powerful and easily scalable: MySQL is capable of handling large amounts of data. Besides, users can expand it if the need arises.
- Fast: MySQL is faster than other software thanks to built-in standards.
- Data recovery possible: MySQL allows users to recover data from the effects of crashes.

Drawbacks of MySQL:

- Restricted: MySQL is limited to a few features that applications may need
- Reliability is not too high: Compared with other relational database management systems, the reliability of MySQL is not too high.
- Limited capacity: The more records in MySQL, the more difficult it becomes to retrieve data due to capacity limitation (Blue Clawdb, 2021).

3.2.4 NGINX

NGINX is a powerful open source web server. NGINX uses a single threaded, event-driven architecture, so it is more efficient than the Apache server. NGINX can also do other important things, such as load balancing and HTTP caching. NGINX can also be used as a reverse proxy. Moreover, Nginx can be deployed to serve dynamic HTTP content on the network using FastCGI, SCGI for scripting, WSGI application server or Phusion Passenger module and it can act as a part load balancer. NGINX uses an asyn-

chronous event-driven approach, rather than threads to handle requests. Nginx's modular event-driven architecture can provide more predictable performance under high load. Nginx's default configuration file is `nginx.conf` (F5 2021).

4 SYSTEM ARCHITECTURE MODEL

4.1 Back-end Side

The Three Tier Architecture model is very popular in Spring Boot. The presentation layer is the communication layer with the end user. All data sent from the server to the client parallelly here only checks the correctness of the data. In REST, three tier has the HTTP Method like GET, POST, PUT, DELETE with the purpose of getting data, adding data, updating data, deleting. The Controller contains all the logic of the program. Moreover, **the** Data access layer interacts with the database, returning the results to the business logic layer (Controller). Specifically, the project is divided into 3 floors (tier or layer) as shown in Figure 5 (Yaghini 2021).

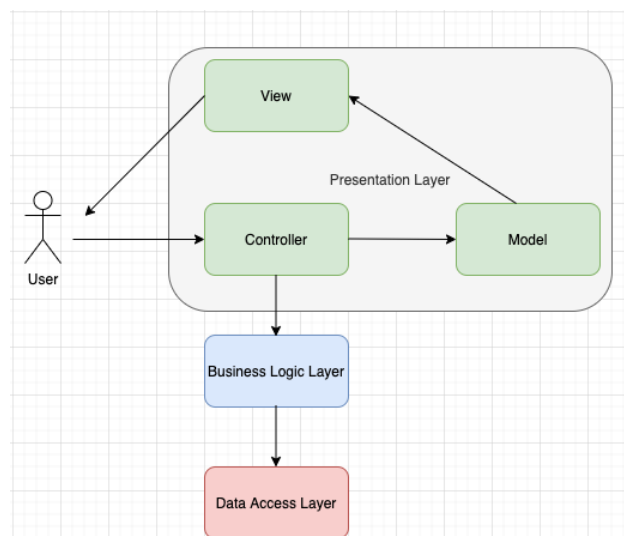


Figure 5. Three Tier Architecture.

4.2 Front-end Side

React is a JavaScript library. React has no constraints in the project directory layout and structure. React gives the freedom to refer to different methods and to apply appropriate. Figure 6 illustrates the structure of ReactJS used in the thesis.

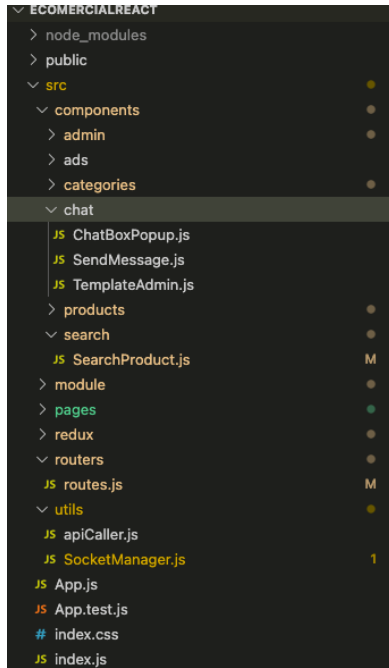


Figure 6. ReactJS Structure.

The operating principle of Redux is illustrated in Figure 7,

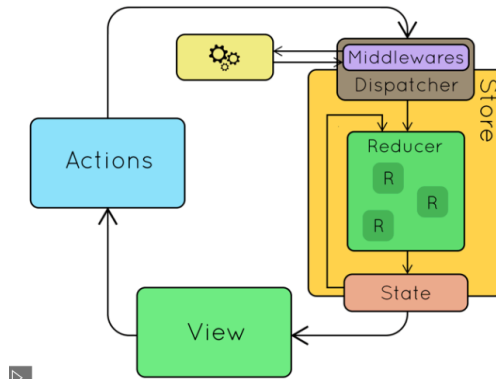


Figure 7. Redux Architecture (Reduxjs 2021).

Figure 8 shows the Action file: the action sends information from the application to the Store, describing what jobs will be done with this store. This information is an object describing what happened. The action consists of two parts, the type which describes the action and the value of the parameter passed.

```

import callAPI, { uploadAPI } from '../../utils/apiCaller';
import * as Types from '../../const/AdminActionTypes';
import qs from 'qs';

// fetch product
export const fetchproduct = ({list, currentPage, total}) => {
  return {
    type: Types.FETCH_PRODUCT_ADMIN,
    products : list,
    page : currentPage,
    total
  }
}

export const fetchProductRequest = (params, callback) => {
  return (dispatch) => {
    return callAPI('api/product/list?' + qs.stringify(params), 'GET', null).then(res => {
      if (res.data && res.data.success) {
        dispatch(fetchproduct(res.data));
        callback(res.data);
      } else {
        callback();
      }
    })
  }
}
}

```

Figure 8. Action File.

Figure 9 shows the Reducer file: Action describes what happened but does not specify which part of the response state has changed and how this will be done by the Reducer. The Reducer takes two parameters: the old state and action information is sent.

```

export default (state = initialState, action) => {
  const copyState = { ...state }
  switch (action.type) {
    case FETCH_PRODUCT_ADMIN: {
      copyState.productAll = action.products;
      copyState.page = action.page;
      copyState.total = action.total;
      return copyState;
    }
    case FETCH_CATEGORY_ADMIN: {
      copyState.categories = action.categories;
      return copyState;
    }
    case FETCH_BRAND_ADMIN: {
      copyState.brands = action.brands;
      return copyState;
    }
    case FETCH_ROOM: {
      copyState.getRoomMessage = action.rooms;
      return copyState;
    }
    case LOG_IN_ADMIN: {
      const dataLogin = jwt_decode(action.data.accessToken);
      localStorage.setItem('tokenAdmin', action.data.accessToken);
      copyState.userAdmin = dataLogin;
      return copyState;
    }
    case USER_INIT_ADMIN: {
      const data = jwt_decode(action.token);
      copyState.userAdmin = data;
      return copyState;
    }
    case ADD_NEW_ROOM: {
      copyState.getRoomMessage = [action.room, ...copyState.getRoomMessage];
      return copyState;
    }
  }
}

```

Figure 9. Reducer File.

Figure 10 shows that the store is an object that stores all state of the application, access state via `getState ()`, update state via `dispatch (action)`. The store contains a dispatcher which is responsible for implementing actions inside the reducer; the reducer is responsible for handling incoming actions. When an action is executed, the dispatcher is implemented and sends an action to the reducer. The reducer now takes action based on the action sent. At the same time, the value of the new state is saved in the store and that new state returned. The dispatcher is the middleware manager, usually used to call APIs, logs.

```
const composeEnhancer = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;
const store = createStore(
  rootReducer,
  composeEnhancer(applyMiddleware(thunk))
);
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);
reportWebVitals();
```

Figure 10. Store.

4.3 Data Flow Model of the Web Page

The data flow model of the web site is shown in Figure 11. First the user will go to View to view and interact on the page. When the user starts to load data, for example, by clicking the Reload button, a request from View is sent to the Controller. The Controller receives the request and begins to ask service (in the code is called the method of Service). The Service receives the request from the Controller, for a simple code that can be calculated and returned. But for operations that need to touch the database, the Service must call the Repository to get data in the database. The Repository receives a request from the Service, will manipulate DB. The data retrieved from the database is mapped by the Object Relational Mapping system (like JPA or Hibernate) into objects (in Java). These objects are called Entities.

The Service receives Entities returned by the Repository. The transformation here is able to perform calculations, add or remove fields, and finally return Entity via the Model. The Model will be returned to the Controller. The Controller receives the Model

and returns to the View. There are two ways, one is to use the template engine to pass Model data into the HTML page, and then return the HTML department (already with the data) to the client.

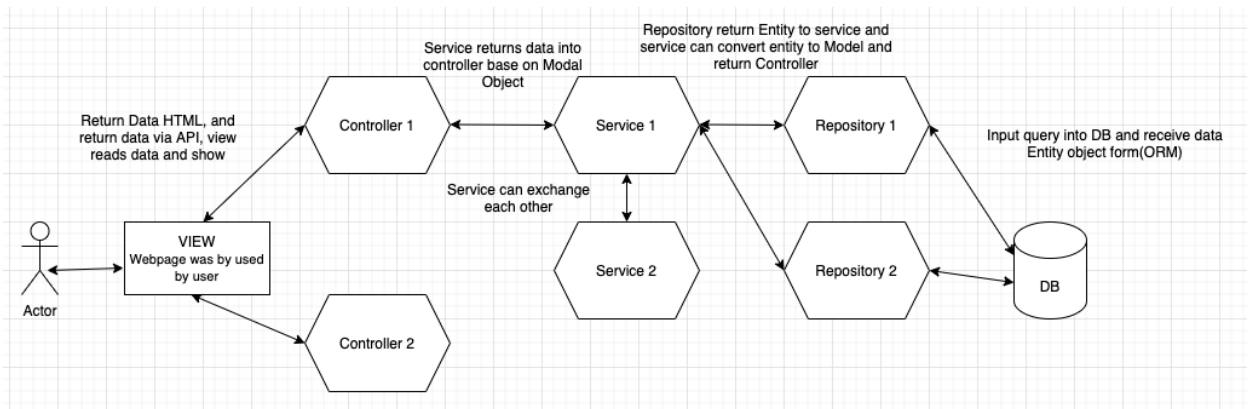


Figure 11. Data flow model.

5 DESIGN

5.1 Class Diagram

Figure 12 shows the class diagram of the application.

Users: The user table is responsible for containing the necessary information when starting to register a member of a website. The fields in the user table are suitable for each user's role.

Roles: The roles table is used to store role information and the roles table has a relationship with the users table to determine which role each user belongs to.

Order: The order table is used to store order information, including shipping information fields, such as name, address, and phone number when a user starts an order from the website.

Products: The products table is used to store information of a product including fields such as name, price, and quantity. The field information in the product table is relevant for the different product categories.

Rating: The rating table is used to contain user rating information about the product.

Categories: The product category table is used to contain product type information including field name and category type codes. All category type codes are unique.

Store: The store table is used to store all store information for each branch.

Brands: The product brand table is used to contain manufacturer information and all manufacturer-related information such as image and the manufacturer's name.

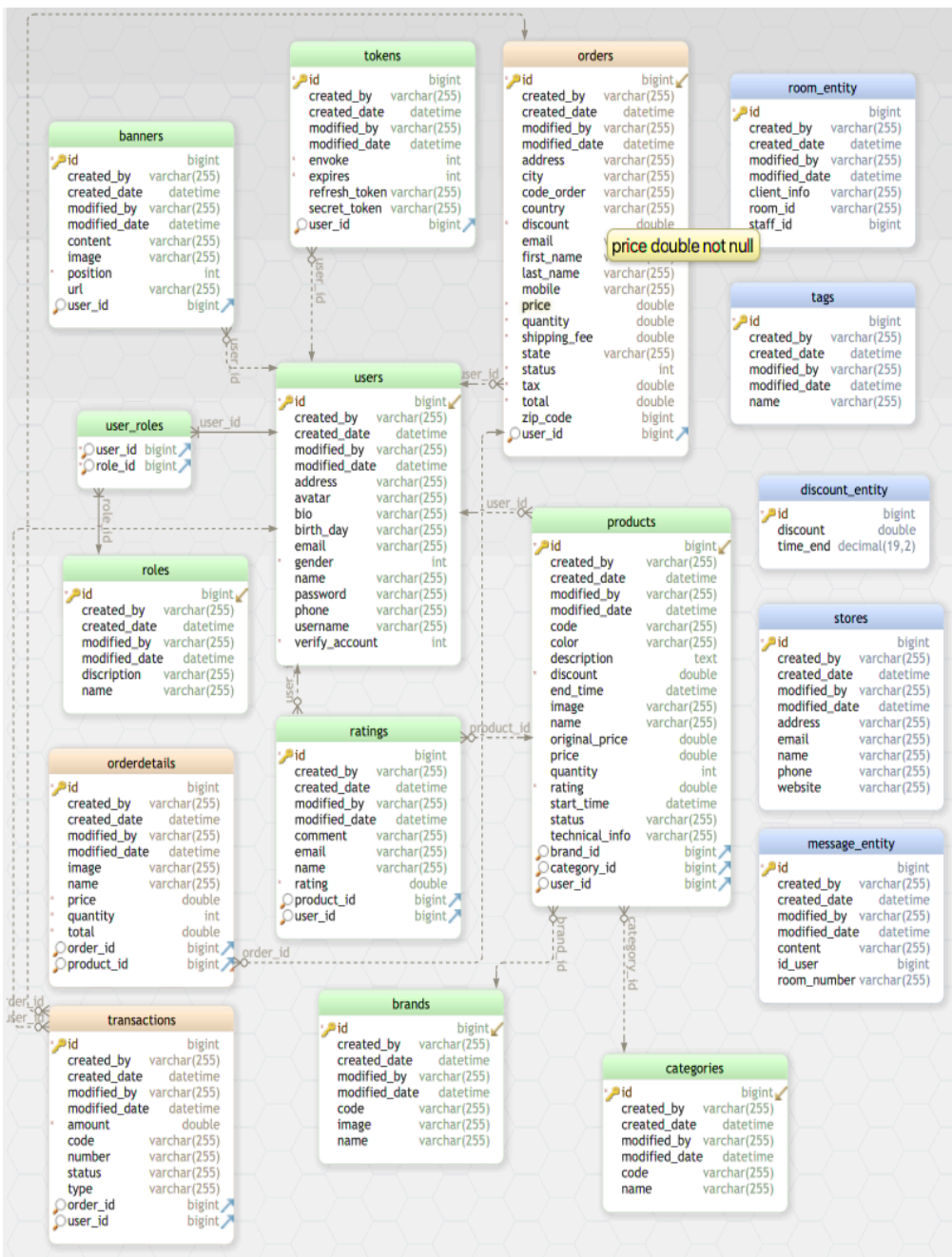


Figure 12. Class Diagram.

5.2 Sequence Diagram

5.2.1 Login

The sequence diagram in Figure 13 shows sequences in login-functionality. First of all, the user will enter the email and password, and then click on the login button. The system will check that the email and password in the database are correct. If the login is not

successful, the user will need to log in again. In case the user logs in successfully, the system grants permissions to access the feature of the website, the user can use the functions in the website.

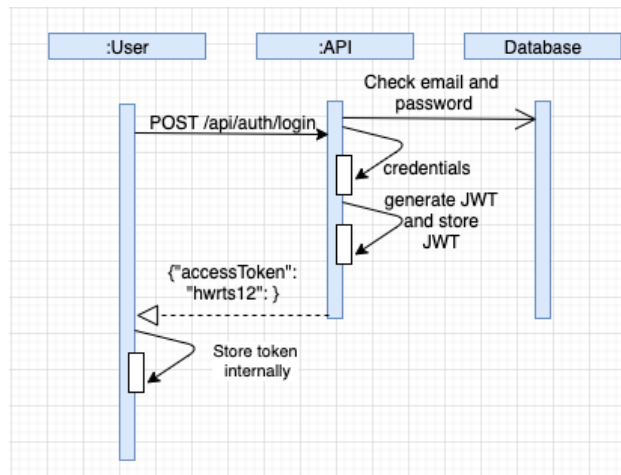


Figure 13. Login.

5.2.2 Logout

The sequence diagram in Figure 14 shows the sequence in logout functionality. The user can log out from the account by clicking the logout button. After the user logout, the system will automatically redirect the user to the home page.

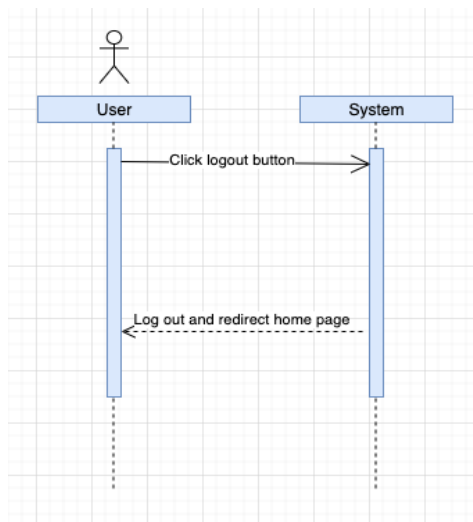


Figure 14. Logout.

5.2.3 Changing Password

The sequence diagram in Figure 15 shows the steps to change the password. The user must login to the system before the user is able to change the password. The web page has a link to change the password. The user clicks the link and fills in the old password, the new password and confirm password -fields. Finally, the user submits the form and the notifications will announce the successful change of the password successfully.

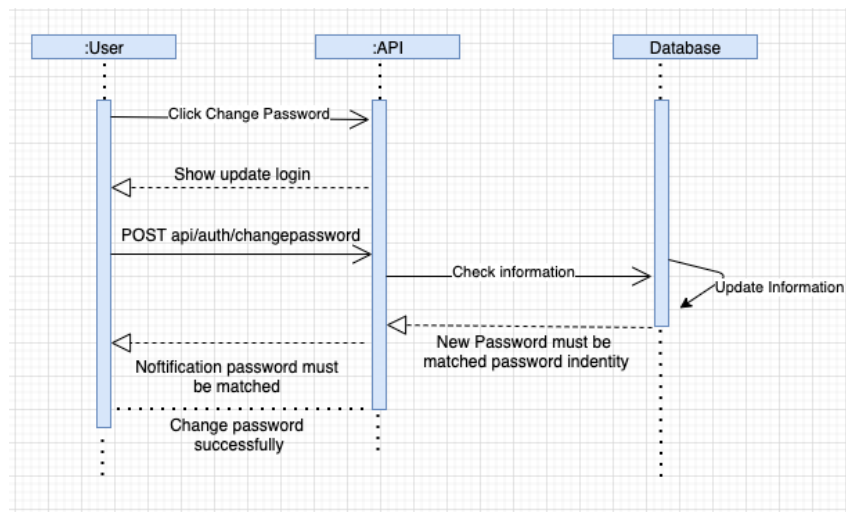


Figure 15. Change Password.

5.2.4 Add Product

The sequence diagram in Figure 16 shows the steps to add a product. The staff must login to the management page and click the add product button to enter the product information. If the product information is wrong, the system will notify the user to enter the information again. The system will save the product to the database when the information is correct.

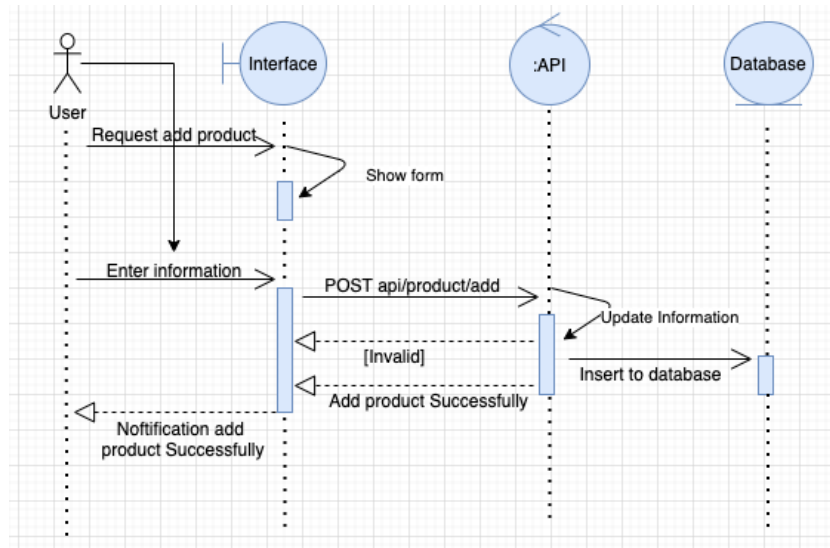


Figure 16. Add Product.

5.2.5 Update Product

The sequence diagram in Figure 17 shows the steps to update a product. The staff must login to the management page and click the update product button to be able to edit product information. If the update information is wrong, the system will notify the user to enter the information again. The system will update the product to the database when the information is correct.

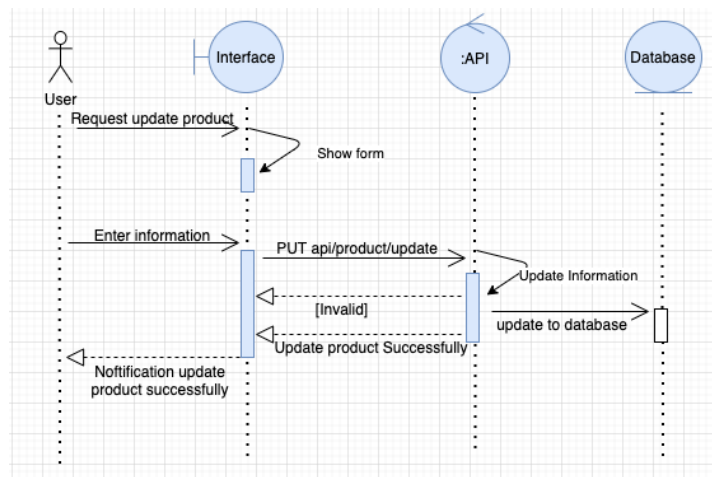


Figure 17. Update Product.

5.2.6 Add Categories

The sequence diagram in Figure 18 shows the steps to add a category. The staff must login to the management page and click the add category button to add the category information to be added. If the category information is wrong, the system will notify the user to enter the information again. The system will save the category to the database when the information is correct.

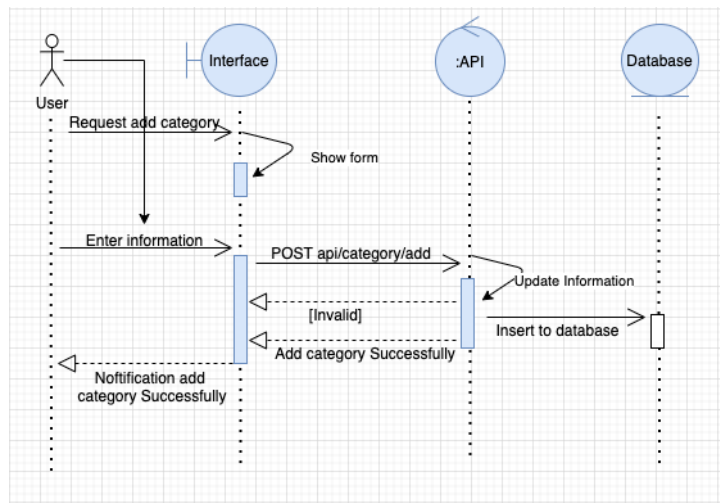


Figure 18. Add Categories.

5.2.7 Update Categories

The sequence diagram in Figure 19 shows the steps to update a category. The staff must login to the management page and click the update category button to edit the category information to be edited. If the category information is wrong, the system will notify the user to enter the information again. The system will update the category to the database when the information is correct.

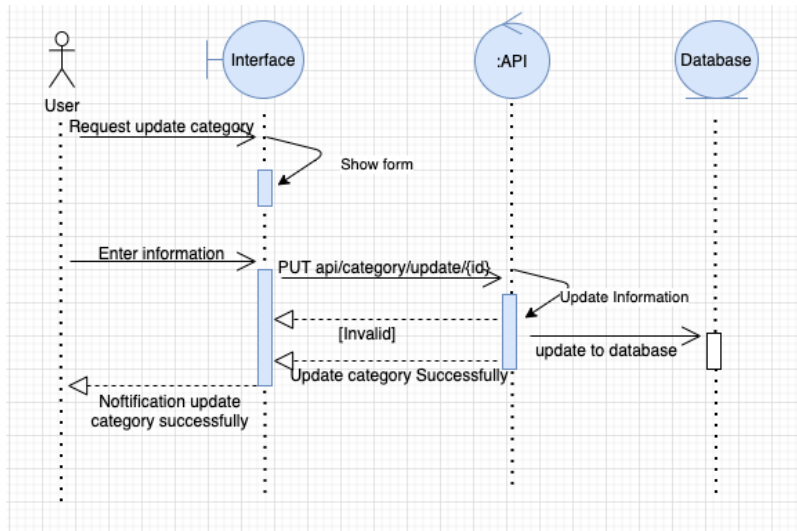


Figure 19. Update Categories.

5.2.8 Add User

The sequence diagram in Figure 20 shows the steps in adding a user. The user accesses the account registration page to fill in the account information. If the account information is valid, it is updated to the database. In case, the information is wrong, the system will request to review the account information.

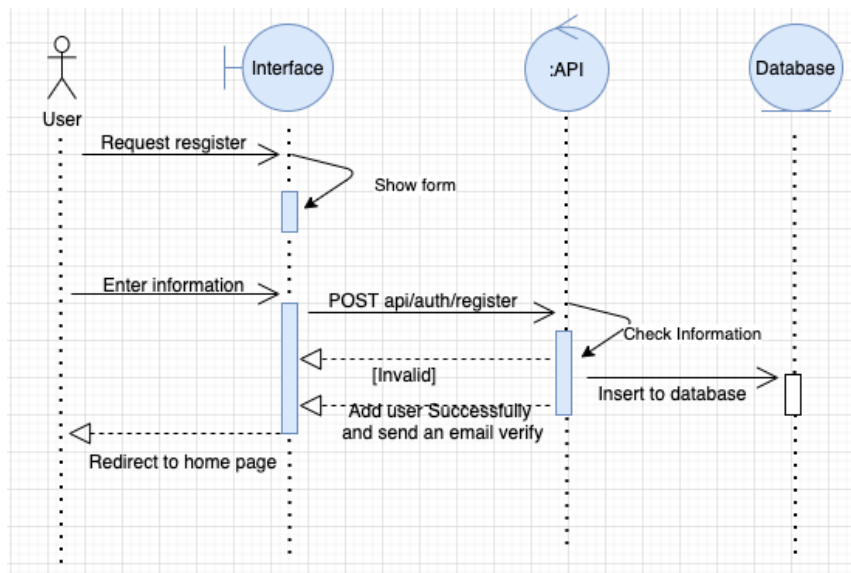


Figure 20. Add User.

5.2.9 Assign Role.

The sequence diagram in Figure 21 shows the steps in assigning roles. The Admin logs in to the system to decentralize accounts for users. Admin is allowed to grant permissions for the role manager and the author permissions for the product management. After the admin adds, modifies and deletes accounts for the user and if that account information is valid, it is allowed to update the database.

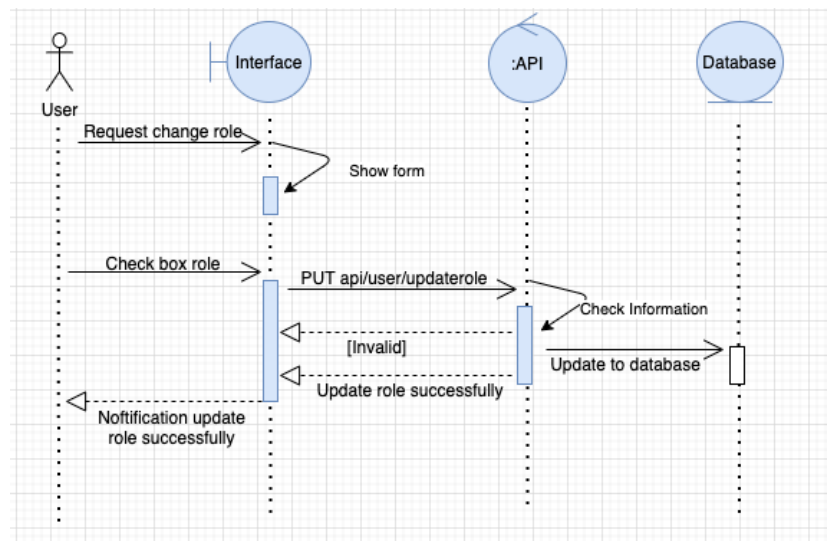


Figure 21. Assign Role.

5.2.10 Chat Room

The sequence diagram in Figure 22 shows the steps in adding a chat room. The user does not have to be logged into the system. The user clicks icon message and needs to fill the information before starting a chat with the staff. If the user information is wrong such as email, the system will notify the user to enter the information again with the correct email format. The system will send a message to the staff when the information is correct

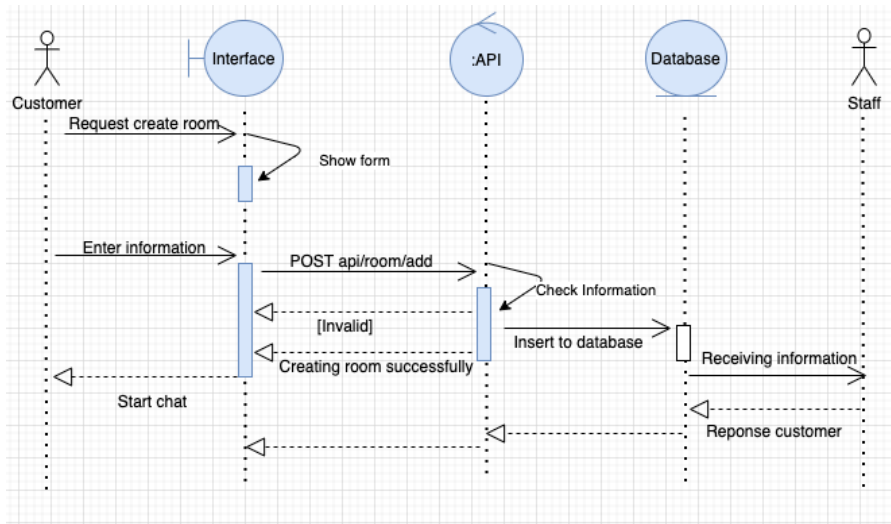


Figure 22. Add Room.

5.2.11 Order

The sequence diagram in Figure 23 shows the steps in order confirm. The user needs to login to the system to make the payment; the user clicks check out button and needs to fill the information before starting payment for the user’s order. If the user information is wrong, the system will notify the user to enter the information again. The system will send an email to the user when the information is correct.

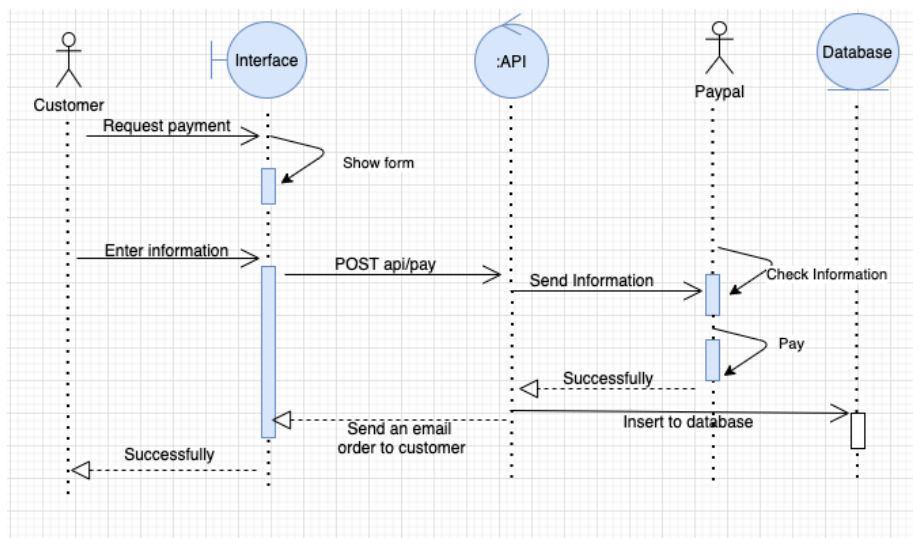


Figure 23. Order and Payment.

6 MODEL

6.1 Functional Hierarchy Chart

The functional hierarchy chart in Figure 24 is represents a simple hierarchical decomposition of tasks to be performed. Each job is divided into sub jobs, the number of division depends on the size and complexity of the system.

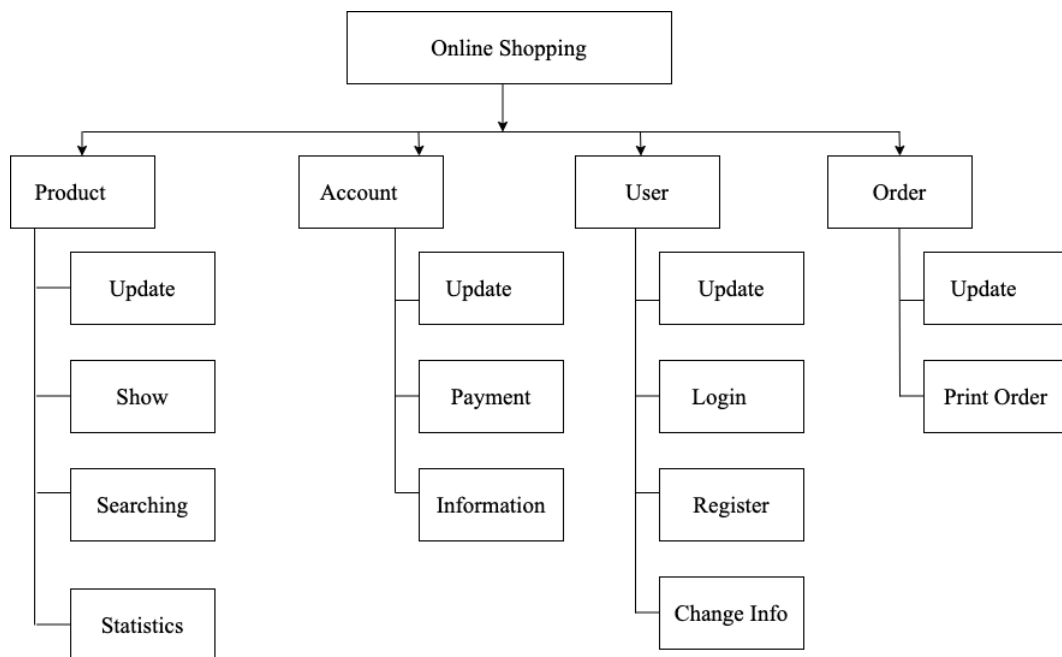


Figure 24. Functional hierarchy chart.

6.2 Use Case Diagram

The use case modal in Figure 25 shows the use case modal relationship diagram between actor and actor, actor with use case and use case.

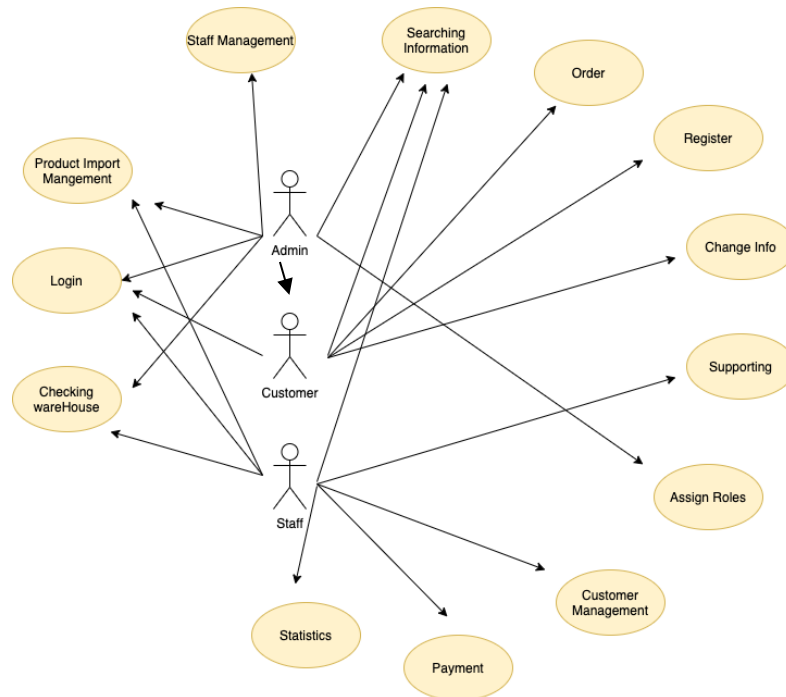


Figure 25. Use case modal.

6.3 Use Case Specification

6.3.1 Login Use Case Specification

The table 1 shows the use case specification for Login functionality

Table 1. Login.

Name	Login
Summary	Function log into the system.
Event	<ol style="list-style-type: none"> 1. The system displays the login form. 2. Enter your username and account (both fields are required) and click "Login". 3. The system checks the login information (Other event stream: Wrong credentials). 4. The system displays the main form.
	<ol style="list-style-type: none"> 1. Wrong password or Email

Other Event	The system displays a message that the login account is not valid.
Special Required	Not available
System state before use case execution	Actor: all of the actors Require: Not available
System state after use case execution	The user who logs into the system can use all the permission of the system allowed.

6.3.2 Logout Use Case Specification

The table 2 shows the use case specification for Logout functionality.

Table 2. Logout.

Name	Logout
Summary	Function log out to the system.
Event	1. The user clicks on log out 2. The system logs out and returns to the home page
Other Event	Not available
Special Required	Not available
System state before use case execution	Actor: all of actors Require: The user has logged on to the system.
System state after use case execution	User logs out of the system.

6.3.3 Changing Password Use Case Specification

The table 3 shows the use case specification for changing password functionality.

Table 3. Change Password.

Name	Change Password
Summary	Change the password for the user account.
Event	1. The user clicks on his or her account and selects "Change Password." 2. User enters the old password, new password and confirm new password
Other Event	Not available
Special Required	The new password must match the confirmation password.
System state before use case execution	Actor: all of the actors Require: The user has logged on to the system
System state after use case execution	User changes the password.

6.3.4 Product Management Use Case Specification

The table 4 shows the use case specification for adding functionality.

Table 4. Add Product.

Name	Add product
Summary	Add product information such as: product name, model name, manufacturer, supplier and other details.
Event	1. Go to product management, press the button "Add product".

	<p>2. The user enters the necessary information (including some required information) and presses "Save".</p> <p>3. The system checks the information, if the information is valid, the next step will be taken.</p> <p>(Other event stream: Invalid information).</p> <p>4. The system saves the data and reports it successfully.</p> <p>(Other event: Cannot add product to database).</p>
Other Event	<p>1. Invalid information:</p> <p>The system displays a red message on the spot of an error and asks to re-enter information.</p> <p>2. Cannot save to database: Error while adding => Ask user to re-enter information, if still error contact the development team.</p>
Special Required	Not available
System state before use case execution	<p>Actor: Admin, Staff</p> <p>Require : The user has logged into the system and has the right to use this function.</p>
System state after use case execution	User successfully added product information to the system.

The table 5 shows the use case specification for editing product functionality.

Table 5. Edit Product.

Name	Update product information
------	----------------------------

Summary	<p>Update product information</p> <p>The system allows updating most of the information.</p>
Event	<ol style="list-style-type: none"> 1. Enter the product list item, click the edit button for a product model 2. The user enters the necessary information (including some required information) and presses "Save". 3. The system checks the information, if the information is valid, the next step will be taken. <p>(Other event stream: Invalid information).</p> <ol style="list-style-type: none"> 4. The system saves the data and notify it of success. <p>(Other event: Cannot update to the database)</p>
Other Event	<ol style="list-style-type: none"> 1. Invalid information: The system displays a message asking for information re-input. 2. Cannot update the database: Error while updating => Request user to re-enter information; if still error, contact the development team.
Special Required	Not available
System state before use case execution	<p>Actor: Staff, Admin</p> <p>Require: The user has logged into the system and has the right to use this function.</p>
System state after use case execution	User successfully updated product information to the system.

6.3.5 Categories Management Use Case Specification

The table 6 shows the use case specification for adding category functionality.

Table 6. Add Categories.

Name	Add categories type
Summary	Add categories type information such as name, code.
Event	<ol style="list-style-type: none"> 1. Go to the product category manager, click the button “Add categories.” 2. The user enters the necessary information (including some required information) and presses "Save". 3. The system checks the information, if the information is valid, the next step will be taken. (Other event: Invalid information). 4. The system saves the data and notify it of success. (Other event stream: Cannot update to the database)
Other Event	<ol style="list-style-type: none"> 1. Invalid information: The system displays a message asking for information re-input. 2. Cannot update the database: Error while updating => Request user to re-enter information; if still error, contact development team.
Special Required	Not available
System state before use case execution	Actor: Staff, Admin

	Require: The user has logged into the system and has the right to use this function.
System state after use case execution	User successfully added the information to the system.

The table 7 shows the use case specification for editing category functionality.

Table 7. Edit Categories.

Name	Update Categories information
Summary	Update categories information The system allows updating most of the information.
Event	<ol style="list-style-type: none"> 1. Enter the categories list item, click the edit button for category model 2. The user enters the necessary information (including some required information) and presses "Save". 3. The system checks the information, if the information is valid, the next step will be taken. (Other event: Invalid information). 4. The system saves the data and notify it of success. (Other event: Cannot update to the database)
Other Event	<ol style="list-style-type: none"> 1. Invalid information: The system displays a message asking for information re-input.

	2. Cannot update the database: Error while updating => Request user to re-enter information; if still error, contact development team.
Special Required	Not available
System state before use case execution	Actor: Staff, Admin Require: The user has logged into the system and has the right to use this function.
System state after use case execution	The user successfully updated category information to the system.

6.3.6 User Management Use Case Specification

The table 8 shows the use case specification for adding staff functionality.

Table 8. Add Staff.

Name	Add Staff
Summary	Add employee Information
Event	<ol style="list-style-type: none"> 1. Go to the staff management section, click the button "Add employee". 2. The user enters the necessary information (including some required information) and presses "Save" 3. The system checks the information, if the information is valid, the next step will be taken <p>(Other event stream: Invalid information)</p> <ol style="list-style-type: none"> 4. The system saves the data and reports it successfully.

	5. After successfully adding an employee, the system will send an email to verify the account.
Other Event	1. Invalid information: The system displays a message asking for information re-input. 2. Cannot update the database: Error while updating => Request user to re-enter information; if still error, contact development team.
Special Required	Not available
System state before use case execution	Actor: Admin, Customer. Require: The user has logged into the system and has the right to use this function.

The table 9 shows the use case specification for editing staff functionality.

Table 9. Edit Staff.

Name	Update Staff
Summary	Update employee information. The system only allows updating almost all of the information.
Event	1. Go to the employee management section, click on the employee button 2. The system will display the interface for the list of employees 3. The user selects the employee you want to edit and clicks the button "Edit" 4. User enter the necessary information (including some required information) and click "Save".

	<p>5. The system checks the information, if the information is valid, the next step will be taken.</p> <p>(Other event stream: Invalid information)</p> <p>6. The system saves the data and reports it successfully.</p> <p>(Other event: Cannot update to the database)</p>
Other Event	<p>1. Invalid information: The system displays a message asking for information re-input.</p> <p>2. Cannot update the database: Error while updating => Request user to re-enter information, if still error, contact development team.</p>
Special Required	Not available
System state before use case execution	<p>Actor: Admin, Customer.</p> <p>Require: The user has logged into the system and has the right to use this function.</p>

7 DESIGN THE MOCKUP

7.1 List of Pages

The table 10 shows the pages that were implemented for the E-commerce web shop.

Table 10. Page List.

STT	Page
1	Login
2	Admin Page
3	Register Page
4	Product List Page
5	Add/ Update Product Page
6	Categories Page Admin
7	Add/ Update Categories Page
9	List of staffs
10	Add/ Update Staff Page
11	Message Room
12	Categories Page
13	BestDeal Page
14	Store Page
15	Change Password Page
16	Cart Page
17	Checkout Page

18	Compare Page
19	Wish Page
20	Message Chat Form

7.2 Detailed Description of Pages

7.2.1 Login

The Login page which is shown in Figure 26 allows users to log in to the websites using email and password when user want to order products through websites store. After successfully logging in, the user can buy products.

Input: Email, password.

Process: Enter your username and password and check if the username and password are valid.

Output: If the username and password are correct, the user can use the system. If wrong, the system will request to re-enter.

Login to your account

E&E provide how all this mistaken idea of denouncing pleasure and sing pain born an will give you a complete account of the system, and expound

Remember me
 [Forgot Your Password ?](#)

Figure 26. Login Page.

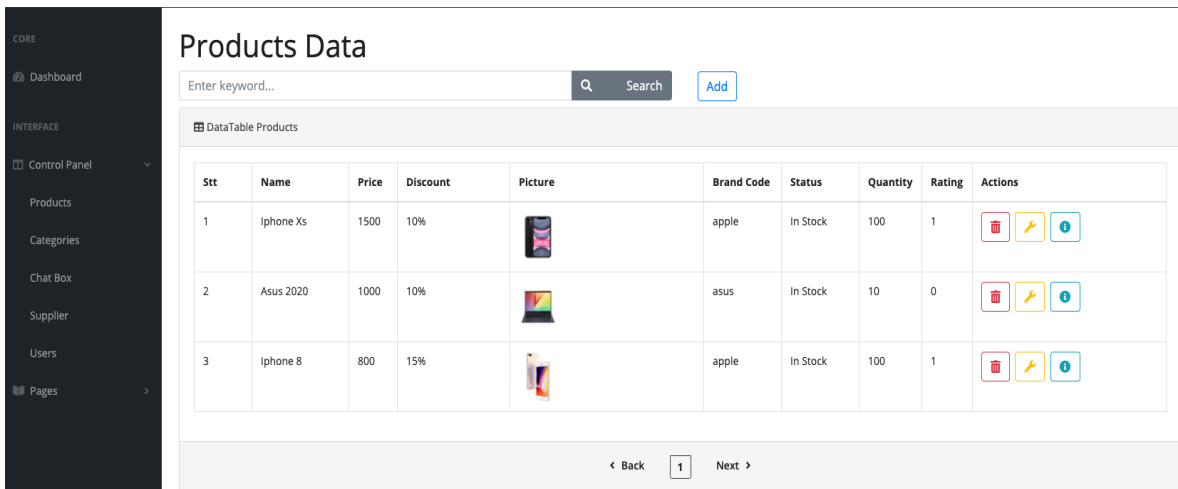
7.2.2 Admin Page

The admin page which is shown in Figure 27 allows admin to manage products data for some method add, update and remove product through websites admin. After successfully logging in, admin has full control data of website.

Input: Email, password.

Process: Enter your username and password and check if the username and password are valid.

Output: If the username and password are correct, the admin can use the system. If wrong, the system will request to re-enter.



The screenshot displays the 'Products Data' section of an admin interface. On the left is a dark sidebar with navigation options: CORE (Dashboard), INTERFACE (Control Panel, Products, Categories, Chat Box, Supplier, Users, Pages), and a search bar. The main content area features a search bar with 'Enter keyword...' and an 'Add' button. Below is a 'DataTable Products' table with columns: Stt, Name, Price, Discount, Picture, Brand Code, Status, Quantity, Rating, and Actions. The table contains three rows of product data. At the bottom, there are navigation controls: '< Back', a page number '1' in a box, and 'Next >'.













Stt	Name	Price	Discount	Picture	Brand Code	Status	Quantity	Rating	Actions
1	Iphone Xs	1500	10%		apple	In Stock	100	1	  
2	Asus 2020	1000	10%		asus	In Stock	10	0	  
3	Iphone 8	800	15%		apple	In Stock	100	1	  

Figure 27. Admin Page.

7.2.3 Register Page

The register page which is shown in Figure 28 allows user can register their account. After successfully register, user can receive an email to active account.

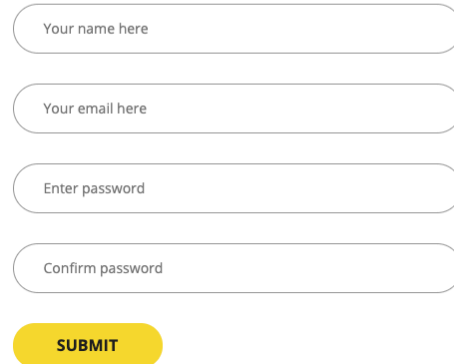
Input: Name, Email, Password, Confirm Password.

Process: Enter information necessary and check if the information are valid.

Output: If the user information are correct, the user will be received an email.

We will need for your registration

E&E provide how all this mistaken idea of denouncing pleasure and sing pain born an will give you a complete account of the system, and expound



A registration form consisting of four rounded rectangular input fields stacked vertically, followed by a yellow rounded rectangular button labeled "SUBMIT". The input fields contain the placeholder text: "Your name here", "Your email here", "Enter password", and "Confirm password".

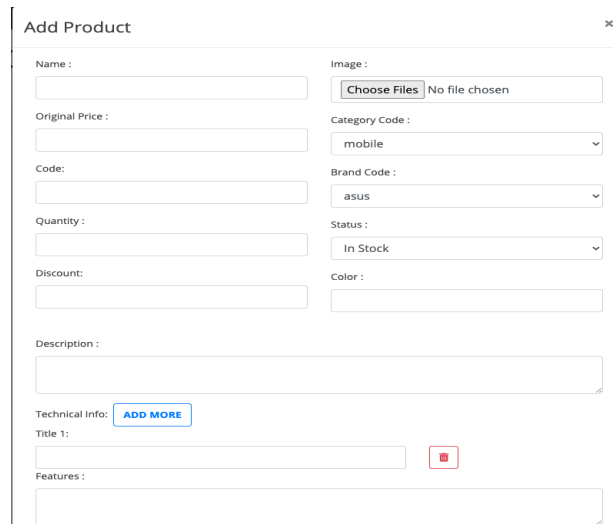
Figure 28. Register Page.

7.2.4 Add/ Update Product Page

The Add/Update page, which is shown in Figure 29 allows admin to add or update product through websites admin.

Input: product information.

Output: If the product information is correct, the admin will be received notification.



An "Add Product" form with a title bar and a close button. The form contains several input fields and dropdown menus arranged in two columns. The left column includes: "Name:" (text input), "Original Price:" (text input), "Code:" (text input), "Quantity:" (text input), "Discount:" (text input), "Description:" (text area), "Technical Info:" (button labeled "ADD MORE"), "Title 1:" (text input), and "Features:" (text area). The right column includes: "Image:" (file upload button labeled "Choose Files" with "No file chosen" text), "Category Code:" (dropdown menu with "mobile" selected), "Brand Code:" (dropdown menu with "asus" selected), "Status:" (dropdown menu with "In Stock" selected), and "Color:" (text input). A small red square icon is visible next to the "Title 1:" input field.

Figure 29. Add/Update Page.

7.2.5 Categories Admin Page

The category page which is shown in Figure 30 allows admin to manage categories data for some methods such as add, update and remove categories through websites admin. After successfully logging in, admin has full control data of website.

Input: Email, password.

Process: Enter your username and password and check if the username and password are valid.

Output: If the username and password are correct, the admin can use the system. If wrong, the system will request to re-enter.

Categories

Dashboard / Tables

DataTable Categories













Stt	Code	Name	Actions
1	mobile	Phone & Tablet	 
2	laptop	Computer & Laptop	 
3	audio	TV & Audio System	 
4	home	Home Appliances	 
5	kitchen	Kitchen Appliances	 
6	accessories	Accessories	 

Figure 30. Categories Admin Page.

7.2.6 Staff Page

The staff page which is shown in Figure 32 allows admin to assign the role for user. To start assign role for user. The admin need to login the admin page and click the button update which shown in Figure 31.



Input: Check box role.

Output: The system shows a notification.

Users

Dashboard / Tables

DataTable Categories

Stt	Name	Roles	Actions
1	TUAN QUANG NGUYEN	ADMIN ACCESS_SWAGGER STAFF	 

< Back 1 Next >

Figure 31. Staff Page.

Update User Role

Name :

Type :

- ADMIN
- ACCESS_SWAGGER
- STAFF

Figure 32. Update Role Page.

7.2.7 Room Page

The room page which is shown in Figure 33 allows staff who can reply the message when client send messages.

Process: The system will receive any message from user when client has question by using socket manager.

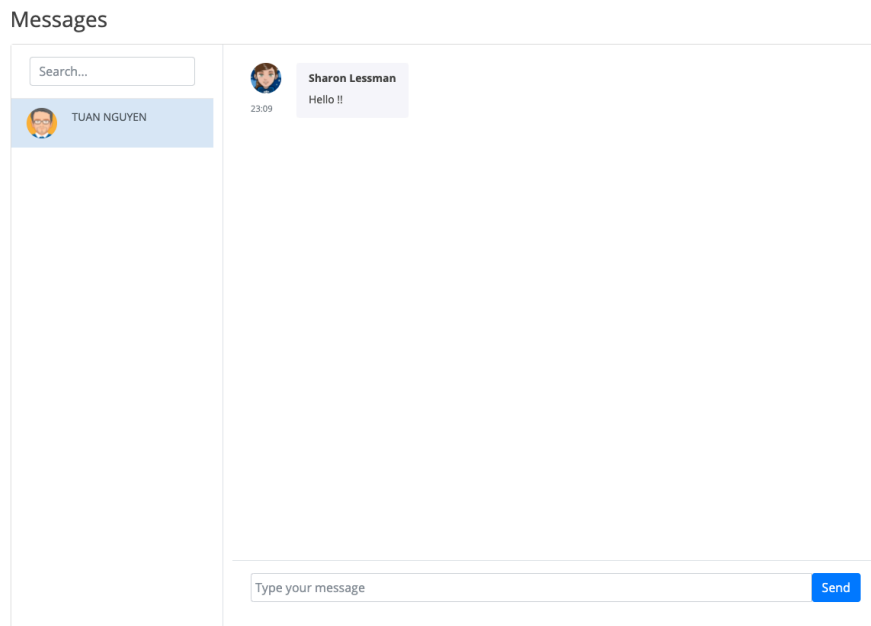


Figure 33. Room Page.

7.2.8 Categories List Page

The category list page which is shown in Figure 34 allows admin to manage categories data for some methods such as add, update and remove categories through websites admin. After successfully logging in, admin has full control data of website.

Input: Email, password.

Process: Enter your username and password and check if the username and password are valid.

Output: If the username and password are correct, the admin can use the system. If wrong, the system will request to re-enter.

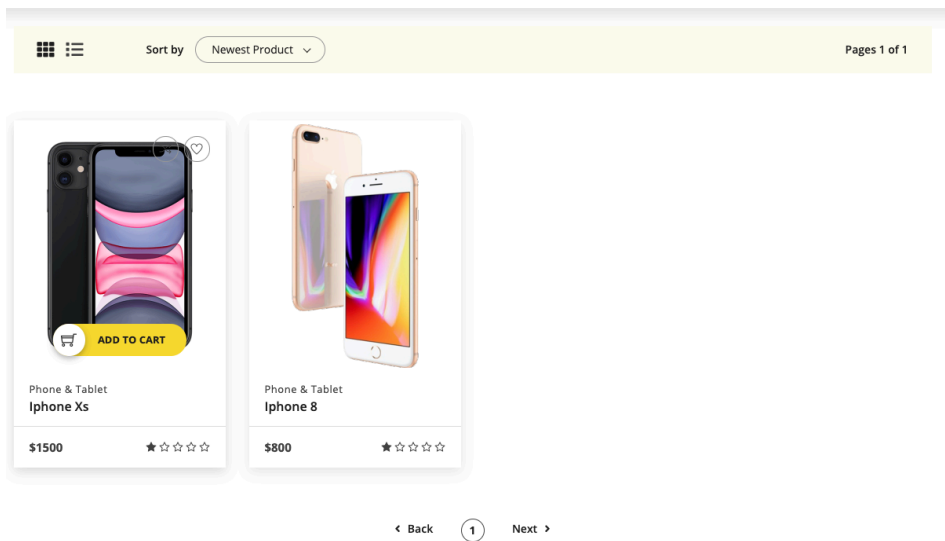


Figure 34. Categories Page.

7.2.9 Best Deal Page

The Best Deal page which is shown in Figure 35 shows to customers that all products are on sale or have the best prices. Products on the highest discount will be time reduce.

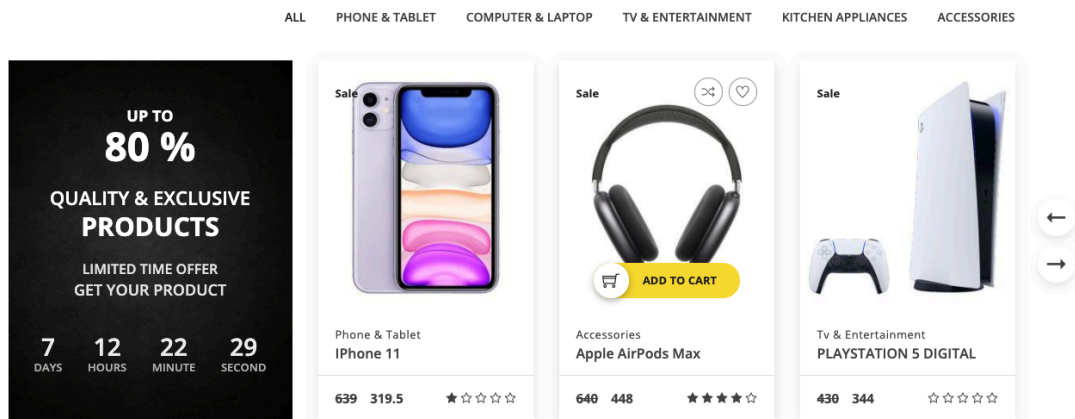


Figure 35. Best Deal Page.

7.2.10 Change Password Page

The Chang password page which is shown in Figure 36 allows user to manage their information data for change password method through web page. The system requires client who had login already.

Input: old password, new password, confirm password.

Process: The system will check if the username and password are valid.

Output: If the username and password are correct, the information will be saved. If wrong, the system will request to re-enter.

CHANGE PASSWORD

E&E provide how all this mistaken idea of denouncing pleasure and sing pain born an will give you a complete account of the system, and expound

Type your old password

Type your new password

Confirm password

[Forgot Your password ?](#)

SUBMIT

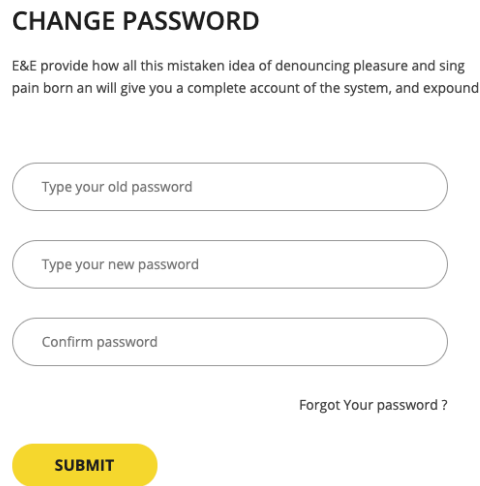
The image shows a web form for changing a password. It has a title 'CHANGE PASSWORD' and a paragraph of text. Below the text are three input fields: 'Type your old password', 'Type your new password', and 'Confirm password'. There is a link 'Forgot Your password ?' and a yellow 'SUBMIT' button.

Figure 36. Change Password Page.

7.2.11 Checkout Page

The Checkout page which is shown in Figure 37 allows user to enter their information data for order product through web page. Client can review all products which was ordered already.

Input: Information fields.

Process: The system will save client information in database.

Output: The system will be redirect to payment page.

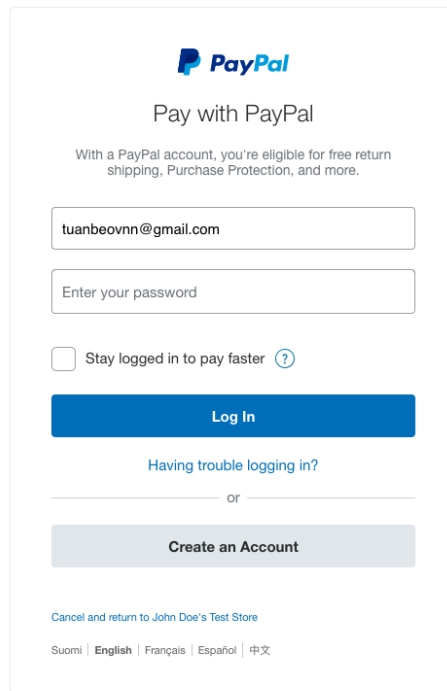
<p>First Name*</p> <input type="text" value="TUAN"/>	<p>Last Name*</p> <input type="text" value="NGUYEN"/>	<table border="1"> <thead> <tr> <th>Product</th> <th>Qty</th> <th>Total</th> </tr> </thead> <tbody> <tr> <td>IMAC 24 "(2021) M1 512 GB</td> <td>x1</td> <td>\$1,755</td> </tr> <tr> <td>IPhone SE</td> <td>x1</td> <td>\$494.1</td> </tr> <tr> <td>Sub Total</td> <td></td> <td>\$2,249.1</td> </tr> <tr> <td>Tax</td> <td></td> <td>10 %</td> </tr> <tr> <td>Grand Total</td> <td></td> <td>\$2024.19</td> </tr> </tbody> </table>	Product	Qty	Total	IMAC 24 "(2021) M1 512 GB	x1	\$1,755	IPhone SE	x1	\$494.1	Sub Total		\$2,249.1	Tax		10 %	Grand Total		\$2024.19
Product	Qty		Total																	
IMAC 24 "(2021) M1 512 GB	x1		\$1,755																	
IPhone SE	x1		\$494.1																	
Sub Total			\$2,249.1																	
Tax			10 %																	
Grand Total		\$2024.19																		
<p>Email Address*</p> <input type="text" value="tuanquangnguyen1710@gmail.com"/>	<p>Phone No*</p> <input type="text" value="465464028"/>																			
<p>Address*</p> <input type="text" value="Palosaarentie 62, E25"/>																				
<p>Country*</p> <input type="text" value="China"/>	<p>Town/City*</p> <input type="text" value="VAASA"/>																			
<p>State*</p> <input type="text" value="FINLAND"/>	<p>Zip Code*</p> <input type="text" value="65200"/>																			
<p>Payment Method</p> <p><input checked="" type="checkbox"/> Paypal</p> <p><input checked="" type="checkbox"/> I've Read And Accept The Terms & Conditions</p>																				

Please Singin Before Purchase

PLACE ORDER

Figure 37. Check out Page.

After placing an order, login page to PayPal payment is opened as shown in Figure 38. PayPal allows user to enter their information data for payment through PayPal page. Client can review the total price to pay. After login to PayPal system will be redirect to payment page and client will receive an email.



Pay with PayPal

With a PayPal account, you're eligible for free return shipping, Purchase Protection, and more.

tuanbeovnn@gmail.com

Enter your password

Stay logged in to pay faster ?

Log In

Having trouble logging in?

or

Create an Account

Cancel and return to John Doe's Test Store

Suomi | English | Français | Español | 中文

Figure 38. Login PayPal

7.2.12 Compare Page

The compare page which is shown in Figure 39 allows user allows users to search and compare the selling price of the products which are interested before making a purchase.

Input: Click to add compare page.



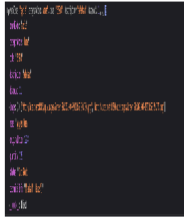



Product	 Phone & Tablet IPhone15	 Phone & Tablet Iphone 8	 Phone & Tablet Iphone 11
Description	Good		dasdasdasdsaddas
Price	1600	800	774
Stock	InStock	In Stock	In Stock
Add to cart			

Figure 39. Compare Page.

7.2.13 Wish Page

The Wish page which is shown in Figure 39 allows where users can review all the products which were interested in before adding them to the shopping cart.

Input: Click to add wish page.



IMAGE	PRODUCT	PRICE	QUANTITY	TOTAL	REMOVE
	lphone 8	800\$	- 1 +	ADDED	
	lphone 11	774\$	- 1 +	ADD TO CART	

Figure 40. Wish Page.

7.2.14 Message Form

The Message Form which is shown in Figure 41 allows user to chat with sales staff to ask for product information or other information.

Input: Enter name and email.

Let's chat? - Online

Please fill out the form below to start chatting with the next available agent.

Your Name

Email Address

Start Chat

X

Figure 41. Message Form.

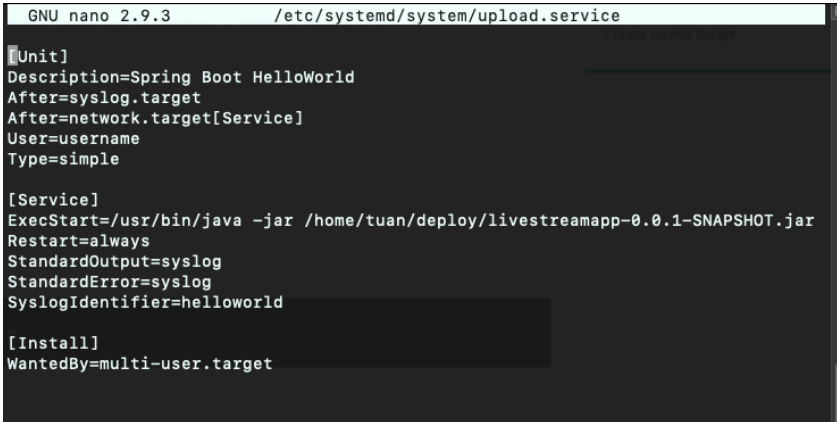
8 IMPLEMENTATION

8.1 Deployment

8.1.1 Back-end and Front-end Side.

a. Init Script

The purpose of script is to read the .jar file following path name “ExecStart” when Spring Boot application start on reboot and run file service by command : “systemctl start”. Figure 42 shows the Init Script.



```
GNU nano 2.9.3 /etc/systemd/system/upload.service
[Unit]
Description=Spring Boot HelloWorld
After=syslog.target
After=network.target[Service]
User=username
Type=simple

[Service]
ExecStart=/usr/bin/java -jar /home/tuan/deploy/livestreamapp-0.0.1-SNAPSHOT.jar
Restart=always
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=helloworld

[Install]
WantedBy=multi-user.target
```

Figure 42. Init Script.

b. Swagger UI

Swagger UI is a tool that allows anyone - from developers to end users - to visualize and interact with project API resources. This tool automatically generates API documents from the Swagger config file, with a visual view and easier client-side deployment. Figure 43 shows all APIs that were implemented for e-commerce web shop (2021 SmartBear).

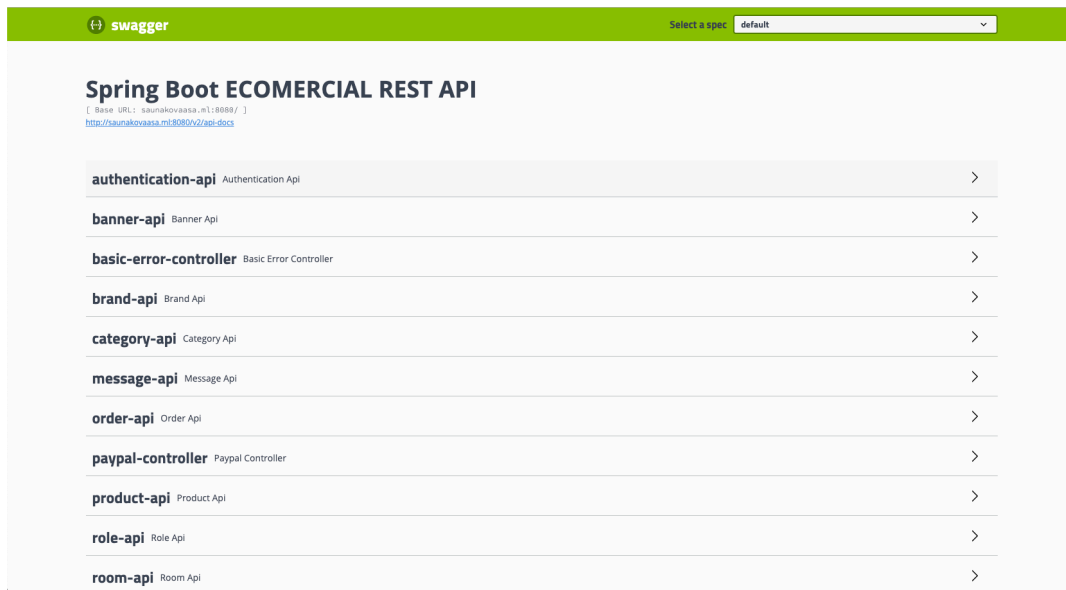


Figure 43. Swagger UI

8.2 Implementation

a) User Register

The class save method to create a new user is shown in Figure 44. When user enter all the necessary information which were required and click on submit, an email will send to user's mail. The user information will be saved to database. Class method is shown in Figure 45 which used to validation information before register such as name, password and email.

```

/**
 * Author @TuanNguyen
 * @param dto
 * @return
 * @throws MessagingException
 */
@Override
public UserDto save(UserDto dto) throws MessagingException {
    UserEntity userEntity = Converter.toModel(dto, UserEntity.class);
    UserEntity userEntity1 = userRepository.findOneByEmail(dto.getEmail());
    if (userEntity1 != null) {
        throw new EmailExistsException();
    }
    List<RoleEntity> roleEntities = new ArrayList<>();
    for (String roles : dto.getRoles()){
        RoleEntity roleEntity = roleRepository.findOneByName(roles);
        if (roleEntity == null){
            throw new ClientException("Your role could not find");
        }
        roleEntities.add(roleEntity);
    }
    userEntity.setRoles(roleEntities);
    userEntity.setPassword(passwordEncoder.encode(userEntity.getPassword()));
    userEntity = userRepository.save(userEntity);
    UserDto userDT0 = Converter.toModel(userEntity, UserDto.class);
    return userDT0;
}

```

Figure 44. Method for user registration.

```

handleRegister = (avatar) => {
    const { email, password, name, confirmPassword } = this.state;
    if (!email || !name || !password || !confirmPassword) {
        let message = 'Email is required';
        if (!name) message = 'Name is required';
        if (!password) message = 'Password is required';
        if (!confirmPassword) message = 'ConfirmPassword is required';
        this.setState({
            error: true,
            message: message,
        })
    } else if (!(/^([a-zA-Z0-9.!#$%&'*/=?^_`{}~]-@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/).test(email)) {
        this.setState({
            error: true,
            message: 'Email is not correct format',
        })
    } else if (password.length < 4) {
        this.setState({
            error: true,
            message: 'Password Length must more than 4 digits',
        })
    } else if (password !== confirmPassword) {
        this.setState({
            error: true,
            message: 'Password and ConfirmPassword are different',
        })
    } else {
        const body = { email, password, name, avatar };
        this.props.registerUser(body, (data) => {
            if (data && data.success) {
                this.setState(
                    {
                        success: true
                    }
                )
            } else {
                this.setState({
                    error: true,
                    message: data && data.message
                })
            }
        });
    }
};
}

```

Figure 45. Method for handling register.

a. Load User

The method to load userName and passWord when user would like to login the web-page is shown in Figure 46. This method has a function to find email in database and check the role of user when login. This is very important method help to get all user data from database.

```
/**
 * @TuanNguyen
 * @param email
 * @return
 * @throws UsernameNotFoundException
 */
@Override
public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

    UserEntity userEntity = userRepository.findOneByEmail(email);
    if(userEntity == null){
        throw new UsernameNotFoundException("User not found");
    }
    MyUserDTO user = Converter.toModel(userEntity, MyUserDTO.class);
    List<GrantedAuthority> authorities = new ArrayList<>();
    user.getRoles().clear();
    for(RoleEntity roleEntity : userEntity.getRoles()){
        user.getRoles().add(roleEntity.getName());
        authorities.add(new SimpleGrantedAuthority(roleEntity.getName()));
    }
    user.setAuthorities(authorities);
    return user;
}
```

Figure 46. Method for LoadUserByUserName.

b. Login

The method to login the web-page using email and password is shown in Figure 49. This is the method to login by using email and password. By default, Spring adds an additional filter in the Spring security filter chain which is capable of persisting the Security Context. The filter uses the repository in order to load and store the security context before and after the execution of the rest of defined filters in the chain, but it uses a custom wrapper over the response which is passed to the chain.

```

/**
 * @TuanNguyen
 * @param loginInfo
 * @return
 */
@Override
public TokenDto login(LoginInput loginInfo) {
    Authentication authentication = null;
    Authentication auth = new UsernamePasswordAuthenticationToken(loginInfo.getEmail(), loginInfo.getPassword());
    SecurityContextHolder.getContext().setAuthentication(auth);
    try {
        authentication = authenticationManager.authenticate(auth);
    } catch (Exception e) {
        throw new EmailOrPasswordNotCorrectException();
    }
    SecurityContextHolder.getContext().setAuthentication(authentication);
    MyUserDTO myUserDTO = (MyUserDTO) authentication.getPrincipal();
    myUserDTO.setRoles();
    final String token = tokenProvider.generateToken(myUserDTO);
    if (myUserDTO.getVerifyAccount() == 0) {
        throw new ClientException("Your Account has not been verified");
    }
    UUID refreshToken = UUID.randomUUID();
    TokenDto tokenDto = new TokenDto();
    tokenDto.setAccessToken(token);
    tokenDto.setRefreshToken(refreshToken.toString());
    tokenDto.setEnvoked(false);

    tokenService.saveToken(tokenDto);
    return tokenDto;
}

```

Figure 47. Method for Log In.

The Figure 47 shows the method using for using Facebook account to login. This method requires the input token from Facebook account, the method will find the information getting from token such as: id, email, name, gender. Moreover, this method will check email with database and if the info does not match. The system will create a new user with email and generate a token for user to login using Facebook.

```

/**
 * @TuanNguyen
 * @param tokenInput
 * @return
 */
@RequestMapping(value = "/facebook/login", method = RequestMethod.POST)
@ApiResponses(value = {@ApiResponse(code = 200, message = "true : success, false : failed")})
public ResponseEntity loginFacebook(@RequestBody TokenInput tokenInput) {

    Facebook facebook = new FacebookTemplate(tokenInput.getAccessToken());
    String[] fields = {"id", "email", "name", "gender"};
    User userProfile = facebook.fetchObject("me", User.class, fields);
    UserEntity userEntity = userRepository.findOneByEmail(userProfile.getEmail());
    if (userEntity == null) {
        UserEntity userEntity1 = Converter.toModel(userProfile, UserEntity.class);
        String username = userProfile.getEmail().substring(0, userProfile.getEmail().indexOf("@") + 1)
            + UUID.randomUUID().toString().substring(0, 3);
        userEntity1.setUsername(username);
        byte[] imageBytes = facebook.userOperations().getUserProfileImage();
        String avatar = uploadService.saveImage(imageBytes, imageName: UUID.randomUUID().toString() + ".jpg");
        String avatarUrl = amazonClient.uploadFileFb(UUID.randomUUID().toString() + ".jpg", imageBytes);
        userEntity1.setAvatar(avatar);
        userRepository.save(userEntity1);
    }
    Authentication auth = new UsernamePasswordAuthenticationToken(principal: "", credentials: "");
    SecurityContextHolder.getContext().setAuthentication(auth);
    MyUserDTO myUserDTO = (MyUserDTO) jwtUserService.loadUserByUsername(userProfile.getEmail());
    auth = new UsernamePasswordAuthenticationToken(myUserDTO, credentials: "");
    SecurityContextHolder.getContext().setAuthentication(auth);
    final String token = tokenProvider.generateToken(myUserDTO);
    UUID refreshToken = Generators.randomBasedGenerator(new Random()).generate();
    TokenDto tokenDto = new TokenDto();
    tokenDto.setRefreshToken(refreshToken.toString());
    tokenDto.setAccessToken(token);
    tokenService.saveToken(tokenDto);
    return ResponseEntity.status(200).body(tokenDto);
}

```

Figure 48. Method for Facebook Login.

c. Add Product

The method to add product when admin would like to add more products to the web-page is shown in Figure 49. This method requires the admin have to login to the management page and press the add product button, when filled in, the information will be saved to the database. In case, there are some errors the information won't save to database.

```
/**
 * Author @TuanNguyen
 * @param productInput
 * @return
 */
@PreAuthorize("hasRole('ROLE_STAFF') or hasRole('ROLE_ADMIN')")
@Override
public ProductOutput save(ProductInput productInput) {
    MyUserDTO myUserDTO = (MyUserDTO) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    ProductEntity productEntity = new ProductEntity();
    productEntity = Converter.toModel(productInput, ProductEntity.class);
    CategoryEntity categoryEntity = iCategoryRepository.findOneByCode(productInput.getCategoryCode());
    productEntity.setCategory(categoryEntity);
    BrandEntity brandEntity = brandRepository.findOneByCode(productInput.getBrandCode());
    productEntity.setBrand(brandEntity);
    productEntity.setPrice((productInput.getOriginalPrice() * (100 - productInput.getDiscount()) / 100));
    String image = String.join(";", productInput.getImage());
    productEntity.setImage(image);
    String color = String.join(";", productInput.getColor());
    productEntity.setColor(color);
    UserEntity userEntity = userRepository.findById(myUserDTO.getId()).get();
    productEntity.setUser(userEntity);
    productEntity = productRepository.save(productEntity);
    return Converter.toModel(productEntity, ProductOutput.class);
}
```

Figure 49. Method for adding product.

```

handleAddProduct = (imageUp) => {
  const { brandCode, categoryCode, code, description, discount,
  image, name, originalPrice, status, technicalInfo, quantity, color } = this.state;
  const tempColor = color.split(';');
  const tech = {};
  technicalInfo.map((item) => {
    tech[item.key] = item.value
  })
  const techString = JSON.stringify(tech); // string to backend
  if (!name || !description || !categoryCode || !brandCode || !quantity || !originalPrice) {
    let message = 'Name is required';
    if (!description) message = 'Description is required';
    if (!categoryCode) message = 'Category Code is required';
    if (!brandCode) message = 'Brand Code is required';
    if (!quantity) message = 'Quantity is required';
    if (!originalPrice) message = 'Original Price is required';
    this.setState({
      error: true,
      message: message,
    })
  } else {
    const body = { brandCode, categoryCode, code, description, discount: Number(discount),
    image: imageUp, name, originalPrice: Number(originalPrice), status, technicalInfo: techString,
    quantity: Number(quantity), color: tempColor };
    this.props.addProduct(body, (data) => {
      if (data && data.success) {
        this.setState({
          success: true
        })
      } else {
        this.setState({
          error: true,
          message: data && data.message
        })
      }
    })
  }
}

```

Figure 50. Handling add product.

d. Sending Email

The method to send email when user would like to register an account on the web-page is shown in Figure 51.

```

@Service
public class EmailService {
  private JavaMailSender javaMailSender;

  public EmailService(JavaMailSender javaMailSender) { this.javaMailSender = javaMailSender; }

  public void sendMail(String toEmail, String subject, String message) throws MessagingException {

    MimeMessage mimeMessage = javaMailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, encoding: "utf-8");
    String htmlMsg = message;
    helper.setText(htmlMsg, html: true);
    helper.setTo(toEmail);
    helper.setSubject(subject);
    javaMailSender.send(mimeMessage);
  }
}

```

Figure 51. Method for sending email.

e. Chatting Support

The method to chat supporting when user would like to ask some information about the product on the web-page is shown in Figure 52. Chatting support is a important method

to configure Spring to enable WebSocket and STOMP messaging. The `EnableWebSocketMessageBroker` annotation is to enable WebSocket message handling, backed by a message broker.

The `configureMessageBroker` method was implemented from `WebSocketMessageBrokerConfigurer` to configure message. It will start `enableSimpleBroker` a simple memory-based message broker to implement the messages back to client on destinations by prefix with `/topic`. The `/app` prefix for messages that are bound for methods `MessageMapping` annotation. This prefix will be used to define all the message mappings.

The `registerStompEndpoint()` method registers the `/ws` endpoint. The client will connect to `/ws` and use the best available transport websocket.

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Autowired
    private JwtTokenUtil jwtTokenUtil;

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker(...destinationPrefixes: "/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint(...strings: "/ws").setAllowedOrigins("*");
    }

    @Override
    public void configureWebSocketTransport(WebSocketTransportRegistration registration) {
        registration.addDecoratorFactory(CustomWebSocketHandlerDecorator::new);
    }

    @Override
    public void configureClientInboundChannel(ChannelRegistration registration) {
        registration.interceptors((ChannelInterceptor) preSend(message, channel) -> {
            StompHeaderAccessor accessor =
                MessageHeaderAccessor.getAccessor(message, StompHeaderAccessor.class);
            return message;
        });
    }
}
```

Figure 52. Web Socket Configuration.

The method will connect with backend API via web-page is shown in Figure 53. Connecting the socket from backend is done by using the endpoint `/ws`. Moreover, the `socket.instate.onChannel()` method is used to listening the message from this user to that user real time.


```

connectSocket = async () => {
  const socket = new WebSocket(subscribeUrl); //create wrapper
  const client = Stomp.over(socket); //connect using your client
  this.setConnectionStatus(WAITING_CONNECTION);

  client.connect(
    {
    },
    () => {
      if (this.getConnectionStatus == OPEN_CONNECTION) {
        client.disconnect();
        return;
      }
      this.stomp = client;
      this.setConnectionStatus(OPEN_CONNECTION);
      this.channel.forEach((v, k) => {
        this.subscribe(k, v);
      });
    },
    (e) => {
      if (this.connectionStatus == OPEN_CONNECTION) {
        this.setConnectionStatus(CLOSED_CONNECTION);
      }
      setTimeout(() => {
        if (this.connectionStatus == CLOSED_CONNECTION) {
          this.connectSocket();
        }
      }, 3000);
    },
  );
};

```

Figure 53. Connect WebSocket Configuration.

```

handleSubmit = (e) => {
  e.preventDefault();
  const email = this.state.email;
  const name = this.state.name;
  this.props.createRoom({ email, name }, (data) => {
    if (data && data.success) {
      this.setState({
        roomId: data.details.roomId,
      })
      SocketManager.instance.onChannel(data.details.roomId, (e) => { // listening va save storage
        console.log(e.data);
        this.state.messages.push(e.data);
        this.setState({
          messages: [...this.state.messages]
        })
        this.scrollToBottom();
      });
      localStorage.setItem("roomChat", data.details.roomId);
    }
  })
}
}

```

Figure 54. On listening message.

The method is to save message when user send message is shown in Figure 54. This is the method save message which means that the message will be saved on database when the user or staff sent message.

The method is to list all message when admin log in to admin page is shown in Figure 55. This is the method get message which means that the message will be showed on admin page.

```

/**
 * @TuanNguyen
 * @param messageDto
 * @return
 */
@Override
public MessageDto save(MessageDto messageDto) {
    MessageEntity messageEntity = Converter.toModel(messageDto, MessageEntity.class);
    messageEntity = messageRepository.save(messageEntity);
    return Converter.toModel(messageEntity, MessageDto.class);
}

/**
 * @TuanNguyen
 * @param room
 * @param pageable
 * @return
 */
@Override
public List<MessageDto> findAllByNewest(String room, Pageable pageable) {
    MyUserDTO myUserDTO = (MyUserDTO) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    List<MessageEntity> messageEntity = messageRepository.findAllByNewest(room, pageable);
    List<MessageDto> messageDto = Converter.toList(messageEntity, MessageDto.class);
    if (myUserDTO.getRoles().contains("STAFF")) {
        RoomEntity roomEntity = roomRepository.findAllByRoomId(room);
        if (roomEntity.getStaffId() == 0) {
            roomEntity.setStaffId(myUserDTO.getId());
            roomRepository.save(roomEntity);
        }
    }
    return messageDto;
}

```

Figure 55. Sending message and list message.

f. Order

The method is to order product when user would like to order a product on the web-page is shown in Figure 56. Method saves order information to the database when the user presses the checkout button to pay and will subtract the quantity in stock.

```

/**
 * @Tuan Nguyen
 * @param orderDto
 * @return
 */
@Override
public OrderDto save(OrderDto orderDto) {
    OrderEntity orderEntity = Converter.toModel(orderDto, OrderEntity.class);
    MyUserDTO myUserDTO = (MyUserDTO) SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    OrderEntity orderEntity = Converter.toModel(orderDto, OrderEntity.class);
    OrderEntity finalOrderEntity = orderEntity;
    List<OrderDetailEntity> orderDetailEntity = orderDto.getDetails().stream().map(e -> {
        OrderDetailEntity orderDetailEntity1 = new OrderDetailEntity();
        ProductEntity productEntity = productRepository.findById(e.getProductId()).get();
        if (e.getQuantity() > productEntity.getQuantity()){
            throw new ClientException("We only have " + productEntity.getQuantity());
        }
        productEntity.setQuantity(productEntity.getQuantity() - e.getQuantity());
        orderDetailEntity1.setProduct(productEntity);
        orderDetailEntity1.setQuantity(e.getQuantity());
        orderDetailEntity1.setOrder(finalOrderEntity);
        orderDetailEntity1.setImage(productEntity.getImage());
        orderDetailEntity1.setName(productEntity.getName());
        orderDetailEntity1.setPrice(productEntity.getPrice());
        orderDetailEntity1.setTotal(productEntity.getPrice() * e.getQuantity());
        orderDetailEntity1 = orderDetailsRepository.save(orderDetailEntity1);
        productEntity = productRepository.save(productEntity);
        return orderDetailEntity1;
    }).collect(Collectors.toList());
    orderEntity.setDetails(orderDetailEntity);
    UserEntity userEntity = userRepository.findById(myUserDTO.getId()).get();
    orderEntity.setUser(userEntity);
    orderEntity = orderRepository.save(orderEntity);
    OrderDto orderDto1 = Converter.toModel(orderEntity, OrderDto.class);
    return orderDto1;
}

```

Figure 56. Method for order product.

```

handleSubmit = (e) => {
  e.preventDefault();
  const { cart } = this.props;
  const { currency, intent, method, firstName, lastName, mobile, zipCode, email, state, city, address, country, description } = this.state;
  if (!firstName || !lastName || !mobile || !zipCode || !email || !state || !city || !address || !country) {
    let message = 'firstName is required';
    if (!lastName) message = 'New is required';
    if (!mobile) message = 'Mobile is required';
    if (!zipCode) message = 'ZipCode is required';
    if (!email) message = 'Email is required';
    if (!state) message = 'State is required';
    if (!city) message = 'City is required';
    if (!address) message = 'Address is required';
    if (!country) message = 'Address is required';
    this.setState({
      error: true,
      message: message,
    });
    console.log(message);
  } else {
    const body = {
      firstName,
      lastName,
      mobile,
      zipCode,
      email,
      state,
      city,
      address,
      country,
      details: cart.map(({ image, name, qty, id, price }) => {
        return {
          image: image && image[0],
          name: name,
          productId: id,
          quantity: qty,
          price: price,
          total: qty * price
        }
      })
    };
    this.props.addOrder(body, (data) => {
      if (data) {
        let price = 0;
        cart.map((item) => {
          price += item.price * item.qty;
        })
      }
    });
  }
}

```

Figure 58. Handling submit order.

g. Chat Room

The method is to create a chat room to start chat with staff on the web-page is shown in Figure 59. User needs to fill their information and the system will automatically release unique room for user and staff. Each client will have different room to chat with staff. The staff will receive message from client through admin page.

```

@Override
public RoomDto save(RoomInput roomInput) throws JsonProcessingException {
    RoomEntity roomEntity = Converter.toModel(roomInput, RoomEntity.class);
    ObjectMapper objectMapper = new ObjectMapper();
    String clientInfo = objectMapper.writeValueAsString(roomInput);
    roomEntity.setClientInfo(clientInfo);
    roomEntity.setRoomId(UUID.randomUUID().toString());
    roomEntity.setStaffId(0L);
    roomEntity = roomRepository.save(roomEntity);
    RoomDto roomDto = Converter.toModel(roomEntity, RoomDto.class);
    roomDto.setClientId(-1L);
    return roomDto;
}

```

Figure 59. Method for creating chat room.

h. Upload Files

The method is to upload files when admin would like to upload image of product on the admin-page is shown in Figure 60. The method uploads the image to the server and returns a image URL to the client. In addition, same method is used to save avatar images when users log in with their Facebook account.

```
/**
 * @TuanNguyen
 * @param files
 * @param scaledWidth
 * @param scaledHeight
 * @return
 */
@Override
public List<String> saveImage(MultipartFile[] files, int scaledWidth, int scaledHeight) {
    List<String> images = new ArrayList<>();
    for (MultipartFile file : files) {
        if (!file.isEmpty()) {
            try {
                String uploadsDir = UPLOAD_DIR;
                if (!new File(uploadsDir).exists()) {
                    new File(uploadsDir).mkdir();
                }
                String tagFile = file.getOriginalFilename();
                String splitTag = tagFile.substring(tagFile.lastIndexOf( str "." ));
                int tagFileSub = tagFile.lastIndexOf( str "." );
                String name = tagFile.substring(0, tagFileSub > 10 ? 10 : tagFileSub);
                String fullName = name + LocalDateTime.now() + splitTag;
                String filePath = uploadsDir + fullName;
                File dest = new File(filePath);
                file.transferTo(dest);
                images.add(HOST+"/api/images/"+fullName);
                UploadService.resize(filePath, scaledWidth, scaledHeight);
            } catch (IOException e) {
                e.printStackTrace();
                return null;
            }
        }
    }
    return images;
}
```

Figure 60. Method for upload picture.

9 TESTING

Table 12 shows a important part of the project in verifying the functionalities. This project is in the process of commissioning and with functions and techniques that suitable to business needs. For testing purposes, this project has been deployed and running on Nginx and accessed on a web browser. For simplicity, the test sample in Table 12 has been used for the test.

Table 1. Testing Content.

S/N	Test Description	Steps	Response	Status
1.	View products by category	Click menu bar to choose category	The list of products by category	Pass
2.	Checking for email existing when registration an account	Enter email to register an account. Click button submit.	The status will show the email has been register already.	Pass
3.	Checking for input information change password.	Enter password information included old and new password	The system will show message if the old password which was not match.	Pass
4	Check out with PayPal payment	Click place order on order page	The application will be redirect to login page.	Pass
5	CRUD product	Enter product information and click button	The product will be showed on the top of list product after added.	Pass

6	Remove item from cart	Click remove product icon.	The item of the list cart will be remove and update total price again.	Pass
7	Checking form message	Enter information to start chat with staff	The system will be showed the fields which were required input.	Pass
8	Add product to cart	Slide button to add product to cart.	The number of products will be show on the right top page.	Pass
9	Add product to wish list	Click icon wish list to add product to wish page.	The number of products will be show on the right top page.	Pass
10	Add product to compare	Click icon comparison to add product to compare page.	The number of products will be show on the right top page.	Pass
11	Access admin page.	Enter account information to login admin page.	The page will be showed notification if account do not have right	Pass
12	Send email register	Enter email account to register.	The system will sent an email to user account to verify.	Pass

13	Forgot Password	Click forgot password page and input email to receive password link reset.	The system will sent an email to user receive password reset.	Pass
14	Receiving message from client	Click menu bar chat box and get the list of client message	The system will be updated immediately	Pass
15	Login with Facebook account	Access login page and click Facebook icon to login by Facebook account	The system will check email, if email had been existed. Client can login. In case, the email does not exists, the system will be automatically created an account and save to database.	Pass
16	Update information account	Access user profile to update information	User can receive the notification update successfully.	Pass
17	Sale time end.	The admin can update product the time end for sale.	The system will return the time end of product.	Pass
18	Search product	Enter the name of product or	The system will return product.	Pass

		choose category to search product name		
19	Sort product	Click the sort product by price, newest and name.	The system will arrange the product by user sort.	Pass
20	Assign role	Check box the role which admin want to change.	The system will update role of users on database.	Pass
21	Register with already Email	Enter information into register form	The system will return a message from backend and frontend will get the error 500 and message "The email has been already exist". Frontend will show the message for client	Failed
22	Insert Product with same code	Enter information into add product form	The system will return a message from backend and frontend will get the error 500 and message "The code has been already exist". Frontend will show the message for client.	Failed

10 CONCLUSION

After implementing the topic and systematically presenting the basic contents of Spring Boot, ReactJS, MySQL and a number of other technologies and techniques in building enterprise Java applications on the web, helping to understand overview of Spring Framework as well as the basic principles and working mechanism of this framework. On the other hand, the websites has a three tier web model architecture in Spring and the mechanisms for securing a web application are supported in the Spring Security module. The thesis gave understanding how the communication mechanism between client and server in modern RESTful service-oriented web model works and how to communication between components of a system.

Commercial web design with separate web server and data server model to increase performance accessing the website. The websites is built to communicate with the server through a RESTful service with the help of ReactJS. UI design with Bootstrap with responsive support makes the websites responsive and user experienced. Building a successful e-commerce website with full basic functions so that users can buy product easily. The web application is simple, elegant and functional with important features for an online store.

The theme of E-commerce web shop is quite popular and highly capable in practical application. However, due to the limited time of research and experience, the E-commerce web only develops at the level of completing the requirements of the topic, the processing speed is not yet completed. The further research will focus on understanding the method of managing the system as well as handling large data blocks with high efficiency, expanding the scope of this project.

11 REFERENCES

- 1) F5. 2021. What is NGINX? Accessed 22.04.2021. <https://www.nginx.com/resources/glossary/nginx/>
- 2) SmartBear. 2021. API Development Accessed 22.04.2021. <https://swagger.io/>
- 3) Facebook Inc. 2021. Getting Started. Accessed 22.04.2021. <https://reactjs.org/docs/getting-started.html>
- 4) Yaghini, V. 2021. Organize your application code in three-tier architecture. OpenClassRooms. Accessed 22.04.2021. <https://openclassrooms.com/en/courses/5684146-create-web-applications-efficiently-with-the-spring-boot-mvc-framework/6156961-organize-your-application-code-in-three-tier-architecture>
- 5) The-Future-Of Customer Engagement and Experience. 2021. What is e-commerce Definition, benefits, examples. Accessed 22.04.2021. <https://www.the-future-of-commerce.com/2020/01/19/what-is-e-commerce-definition-examples/>
- 6) Henderson, R.2021. What is PayPal and how does it work ?. Access 22.04.2021. <https://www.pocket-lint.com/apps/news/138438-what-is-paypal-and-how-does-it-work>
- 7) Spring Boot. 2021. Overview. Access 22.03.2021. <https://spring.io/projects/spring-boot>.
- 8) Tutorialspoint. 2021. HTML Tutorial. Accessed 22.04.2021 <https://www.tutorialspoint.com/html/>
- 9) W3Schools. 2021. CSS Tutorial. Accessed 22.04.2021 <https://www.w3schools.com/css/>
- 10) Developer Mozilla. 2021. What is JavaScript ?. Assessed 13.05.2021. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
- 11) Factory. 2021. Benefits of E-commerce for businesses and consumers. Accessed 13.05.2021. <https://factory.hr/blog/benefits-of-E-commerce-for-businesses-and-consumers>
- 12) BloomIdea. 2021. Types of e-commerce. Accessed 13.05.2021. <https://bloom-idea.com/en/blog/types-e-commerce>

- 13) Maxoffsky. 2021. Building RESTful API in Laravel. Assessed 13.05.2021.
<https://maxoffsky.com/code-blog/building-restful-api-in-laravel-start-here/>
- 14) Tram, H.2021. What is the RESTful API ? Assessed 13.05.2021.
<https://itzone.com.vn/en/article/what-is-the-restful-api/>
- 15) Herawan Dwika, P.2021. What is MySQL: MySQL Explained For Beginners.
Assessed 13.05.2021. <https://www.hostinger.com/tutorials/what-is-mysql>
- 16) Elated. 2021. MySQL for Absolute Beginners. Assessed 13.05.2021.
<https://www.elated.com/mysql-for-absolute-beginners/>
- 17) BlueClawdb. 2021. MySQL Advantages and Disadvantages. Assessed
13.05.2021. <https://blueclawdb.com/mysql/advantages-disadvantages-mysql/>
- 18) Reduxjs. 2021. Redux Fundamentals., Part 6 Async Logic and data Fetching.
Assessed 13.05.2021. <https://redux.js.org/tutorials/fundamentals/part-6-async-logic>