RMIT UNIVERSITY

# EEET2580

# Enterprise Application Development

# Group Assignment Report

---

*Lecturer: Dr. Nguyen Ngoc Thanh*

*Submission Date: 25-05-2021*


Group 8:

| | |
|---|---|
| Phạm Huỳnh Ngọc Huệ | s3702554 |
| Phạm Thành Đạt | s3678437 |
| Park Anh Kiệt | s3681475 |
| Hoàng Trần Tuấn Ngọc | s3681441 |

## 1. INTRODUCTION

With the outbreak of the Covid-19 pandemic, consumers tend to buy more online than before because they want to avoid crowded places and protect their health. According to Unicef statistics, during the world's lockdown, businesses and consumers are increasingly interested in e-commerce. As a result, businesses have started a digital transformation process and offered more goods and services online, increasing the e-commerce market share in global retail commerce from 14% in 2019 to about 17% in 2020 [1]. Seizing this opportunity, our team decided to build Destination, an e-commerce website, to serve customers with online shopping needs. Our sales platform can be used on most devices, such as computers, smartphones, tablets, etc. Currently, Destination focuses on selling high-tech products such as smartphones, tablets, and headphones. With a reasonable and intelligent website building, we can turn Destination into an e-commerce site selling most of the products on the market. In the future, our team plans to develop Destination into an e-commerce application like Tiki and Lazada to attract more customers.

## 2. BUSINESS REQUIREMENT

Our website provides an online shopping system specializing in selling electronic products, such as headphones, tablets, and smartphones. Each user will create their account to manage their orders. For security measurements, we will ask users to confirm their registered accounts via personal email. After successful registration and login, users can shop for electronics at reasonable prices. During the purchase process, the customer can add or remove their products. The shopping cart will be updated with quantity and total amount automatically. Moreover, users can manage and update their account information easily and quickly in our application. With the essential, user-friendly, and valuable features of our website, customers will be able to use and experience the shopping space at home without going out.

## 3.USE CASE DIAGRAM


*Figure.1 Online shopping page case diagram*

## 4.TECHNOLOGIES

### 1/ Front end

- Bootstrap: A CSS framework responsible for a responsive design for mobile-first websites.
- HTML: HyperText Markup Language is the basic scripting language used to create websites and display them on the world wide web.
- CSS: Cascading Style Sheets used to decorate elements created in HTML files. It provides control of multiple website layouts all at once
- Javascript: A scripting language that controls the page's complex features such as displaying timely content updates, interactive maps, etc.

**2/ Back end**

- Java EE: Java EE stands for Java Enterprise Edition, and it is mainly used for developing web applications. Moreover, the Java EE also contains a set of specifications from Java SE (Standard Edition) and provides a platform for developers to code and develop several enterprise applications such as web service, distributed computing, reading, and writing from a database [2]. The Java EE is ultimately utilized in developing websites due to its functionality, security, and portability.

**3/ Database**

- PostgreSQL: This is a powerful, free open source object-relational database [3] that has earned many achievements such as reliability, feature robustness, and performance. In this project, we will use this database for storing and extracting data from shopping web applications.

**4/ Framework**

A framework is also known as a concrete platform where pre-written code with generic functionalities can be selected and overridden by other developers or users. Generally, software frameworks will take the form of libraries, which define the application program interface (API) in detail that can be reused anywhere during the development [4]. In this project, our group will use the Spring Boot framework to develop the website back end.

- Spring Boot: This is an open-source Java-based helpful framework for building stand-alone, production-grade applications with low effort [5]. It helps us to configure, code, and efficiently run the web application with fast response. To be more specific, Spring Boot concentrates on providing flexibility by allowing developers to inject different dependencies, which can shorten the code length and support an easier way to operate Spring applications [6].

**5/ Tools**

- Visual Studio Code: This tool is a source code editor which all developers widely use. It comes with built-in support for other languages such as Pythons, C++, etc.… For frontend developers, it supports productivities with syntax highlighting, snippets, bracket-matching, live server, and more.[7]
    - Syntax highlighting: highlight the source code with colors so the developer can distinguish their codes.
    - Snippets: make entering repeated code lines more productive and straightforward.
    - Bracket-matching: this allows developers to distinguish between brackets and easy to identify missing brackets.
    - Live server: enable developers to check their project behaviors on browsers.
- Intellij IDEA: Intellij provides a deep analysis of your code. It's capable of seeking relationships between symbols across all projects and languages. IntelliJ IDEA gives a unified interface for major version control systems, including Git, SVN, Mercurial,

CVS, Perforce, and TFS. The IDE lets you scan the changes, manage branches, merge conflicts, and much more [*8*]
    - Build-in decompiler for Java classes to peek inside a library you don't have without third-party plugins.
    - Database tools: gives significant advantages when editing SQL, live database connection, and schemas management.
    - Terminal: with a built-in terminal, developers can terminate code without leaving the IDEA application.
- Postman: Postman is a great tool to analyze your APIs. It provides a clean interface to make HTML. With Postman, developers can RUN, PATCH, DELETE, and many other requests to develop API. Aside from that, when using Postman for API testing, developers can save former test data to global variables. [*9*]
- PgAdmin4: is widely used by developers for managing databases. It helps developers to create, maintain and use databases by offering a clean and neat UI (User Interface). It offers a rich set of features for database management, such as *[10]*
    - Built-in SQL editors: Import SQL scripts
    - Auto-generate SQL scripts
    - Clean GUI (Graphical user interface) allows users to accustomed to it
    - Accessible through the website or downloadable app.
- Github: Github is a code hosting platform. It provides code control and collaboration between developers.GIT is a version control system. It keeps the code revision history straightforward and stores the modifications in a repository. Hub is the nucleus around which everything involving Git revolves.[*11].*

## 5. ARCHITECTURE

### 5.1.1 Back End

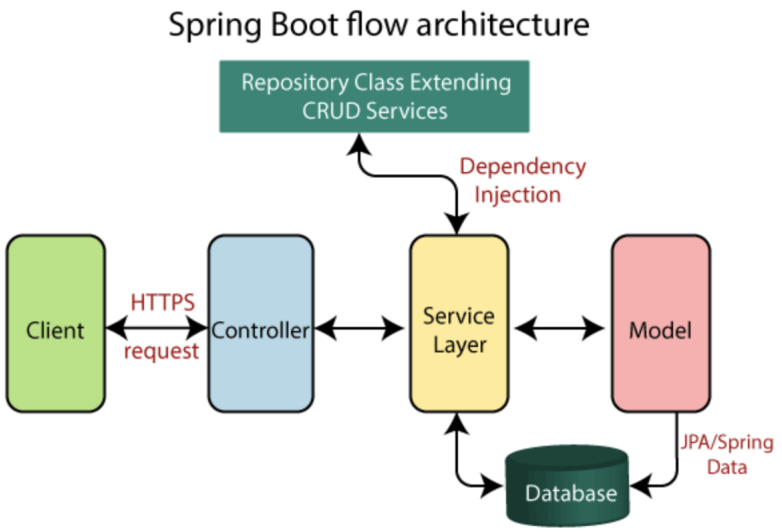


*Figure.2 Spring Boot flow architecture*

Spring Boot is a project developed by JAV (java programming language) and is a module of Spring Framework. Spring Boot is commonly used by developers to create standalone

applications based on Spring and does not require XML configuration. Today, Spring Boot is considered a standard for software design configuration, which simplifies programming the application and focuses on developing the business system's application. Our back-end system is built based on Spring-boot architecture

Config: In application.properties, we have defined spring.jpa.properties.hibernate.hbm2ddl.auto=create-drop. So after we shut down, our database will automatically drop all previous tables. Therefore, we have created a config file to set up and store data in the database during the program launch.

Entity: is where the classes represent domains. It can be understood as business objects such as user, product, cart, etc. At the same time, each class also represents a table in the database. Field fields in a class will correspond to columns in the data table

DTO (Data transfer object): is a place to encapsulate data to transfer between users-servers or between services. DTO reduces the amount of unnecessary information that needs to be transferred, while also increasing security.

Repository: is where the Interface classes receive an entity class corresponding to the java class representing a table in the database and the data type of that table's ID field. It is responsible for communicating with the database, storing, processing queries, and returning data types requested by the service layer (CRUD).

Controller: is the place where the classes are responsible for receiving requests and communicating with the appropriate service classes to handle the request.

Service: is a place to store classes that handle business logic

Security: to be the centralized place to handle information related to web security.

## 5.2. Front End

### *5.2.1. Main Layout*

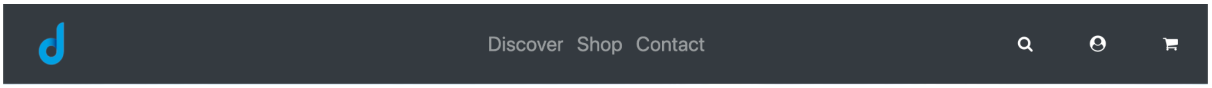


*Figure.3 Navigation Bar*

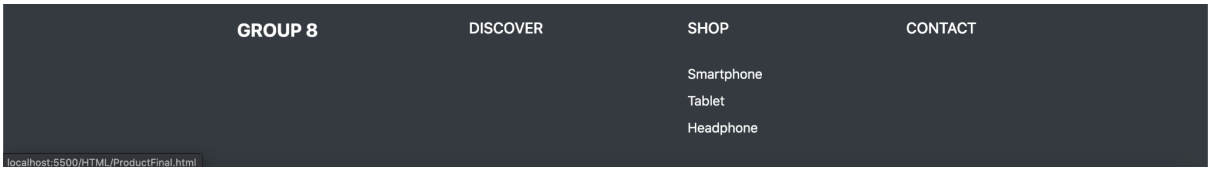


*Figure.4 Footer*

The navigation bar and the footer are made using bootstrap 4 formats. The navigation bar will be displayed on top of the page all time as the user switches pages. The footer will of course stick at the bottom of the page.

### *5.2.2. Main Page*

The main page will have a carousel slide showing some of our featured products. Below the slide are some of the highlighted products of each category that will be exhibited for the user to check and compare the price.

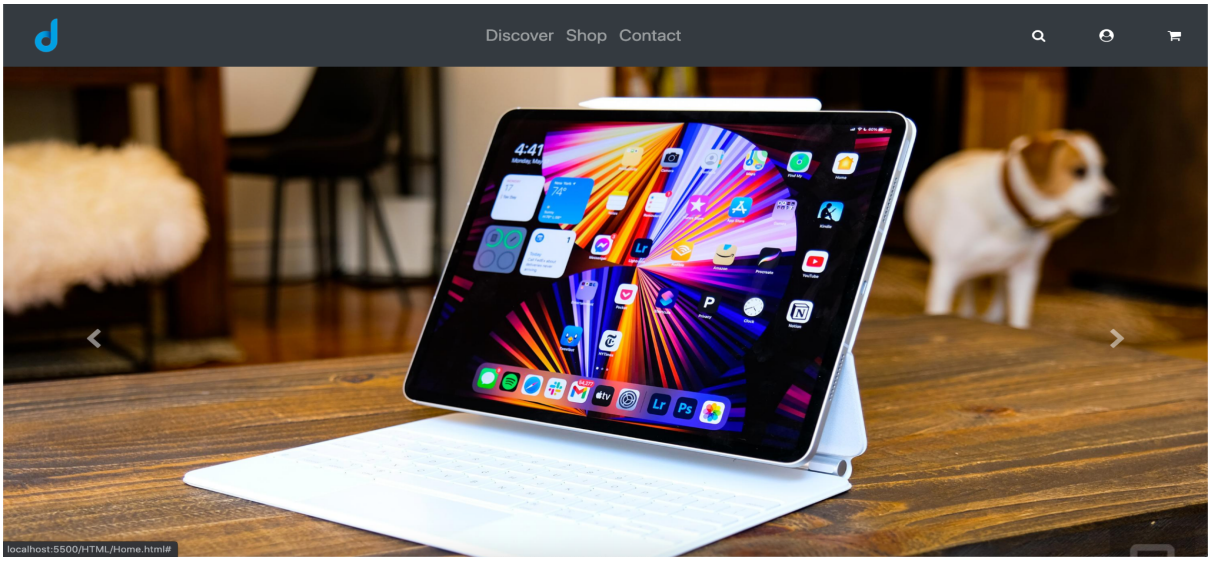


*Figure.5 Carousel slider*

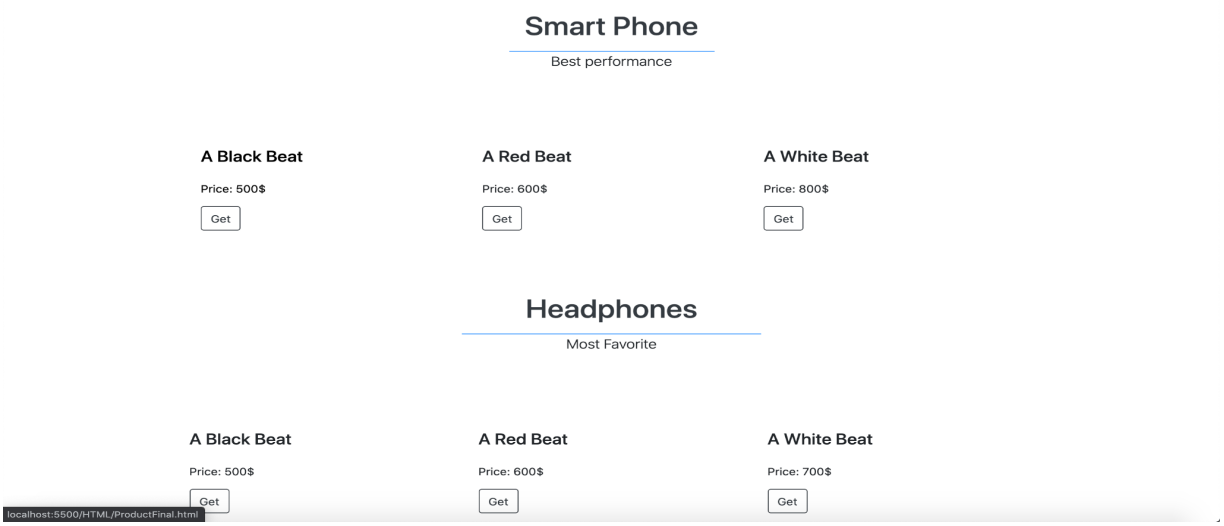


*Figure.6 Homepage*

### *5.2.3. Product Page*

The product information page will appear displaying all product specifications such as name, brand, price, type, warranty period, and detailed description.

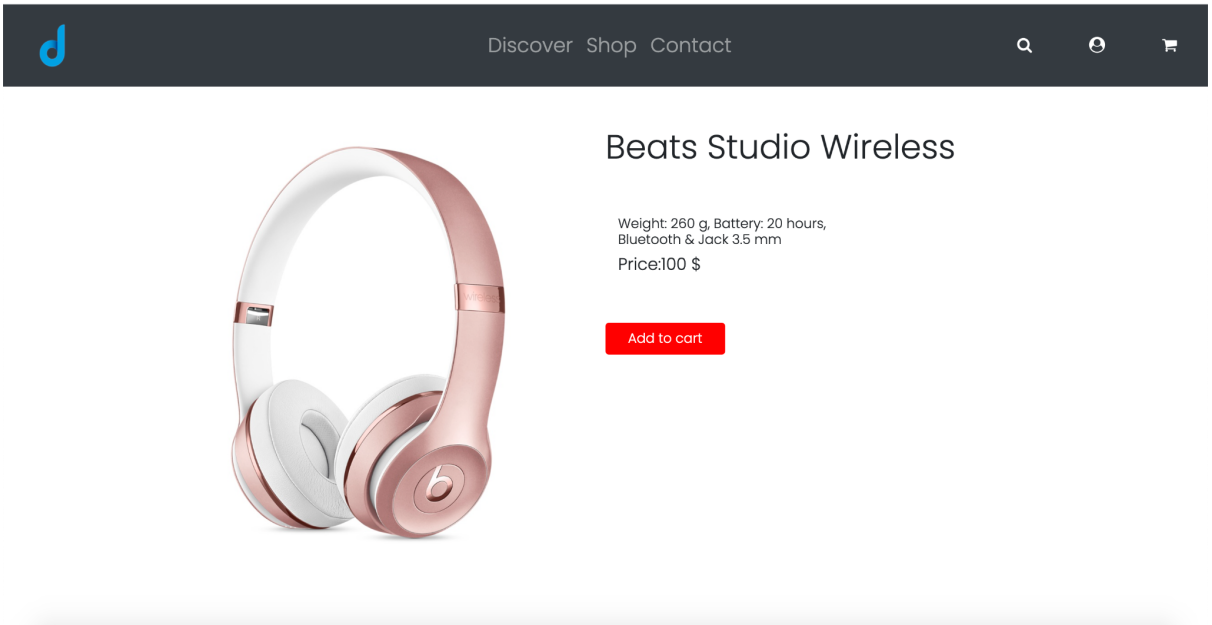


*Figure.7 Product page*

Below the currently viewed product will be the most viewed category and related products.

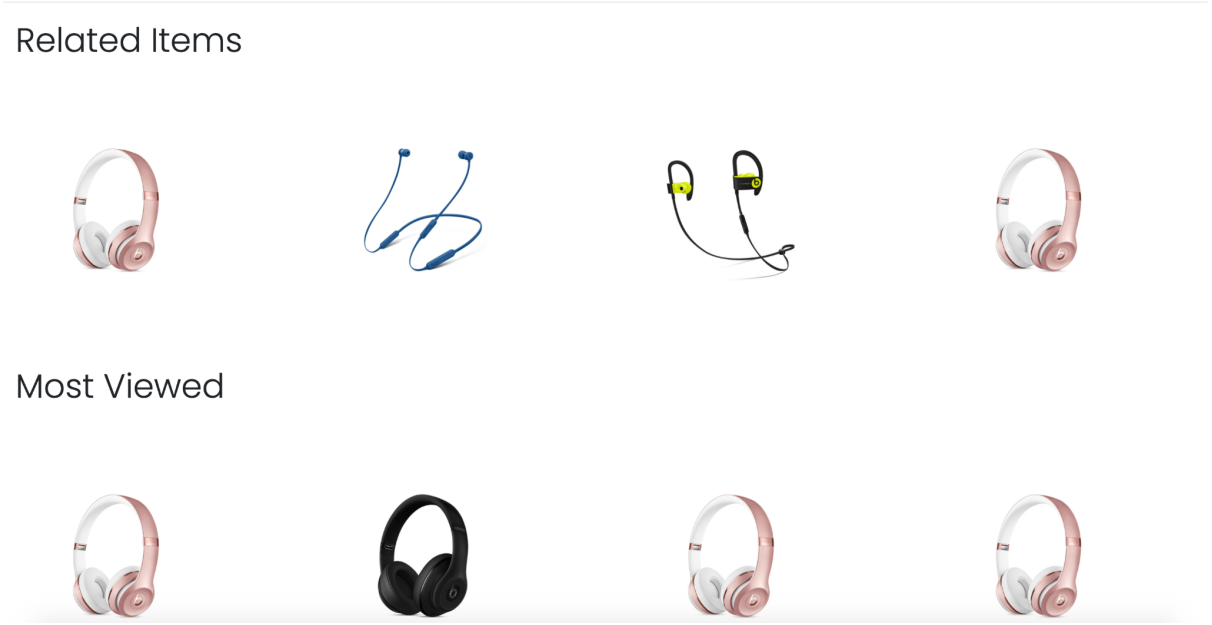


*Figure.8 Related and Most viewed items*

These items will be displayed as a carousel slide so the users won't have to look for other items as they will be later shown on screen.

### *5.2.4. Checkout Cart*

When users put an item in the cart and want to purchase it, the page will redirect users to the checkout page. Items that the user would like to purchase will be displayed here. The users can change the quantity as much as they want.

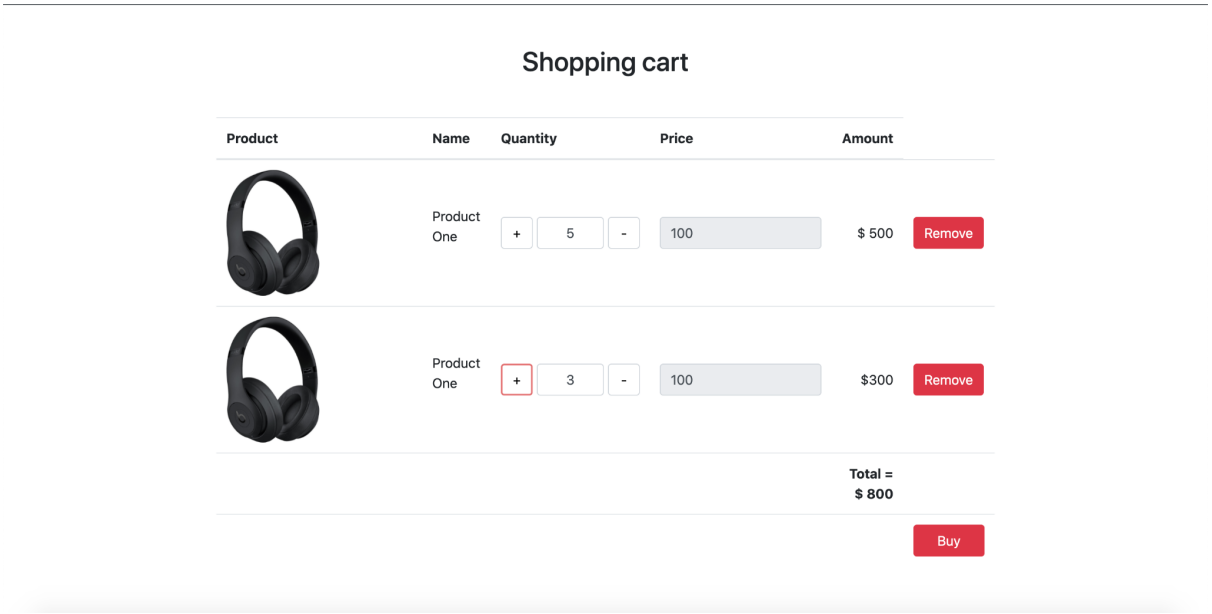


*Figure.9: Checkout cart*

### *5.2.5. User profile*

The user can proceed to their information page and change all of their details later on.

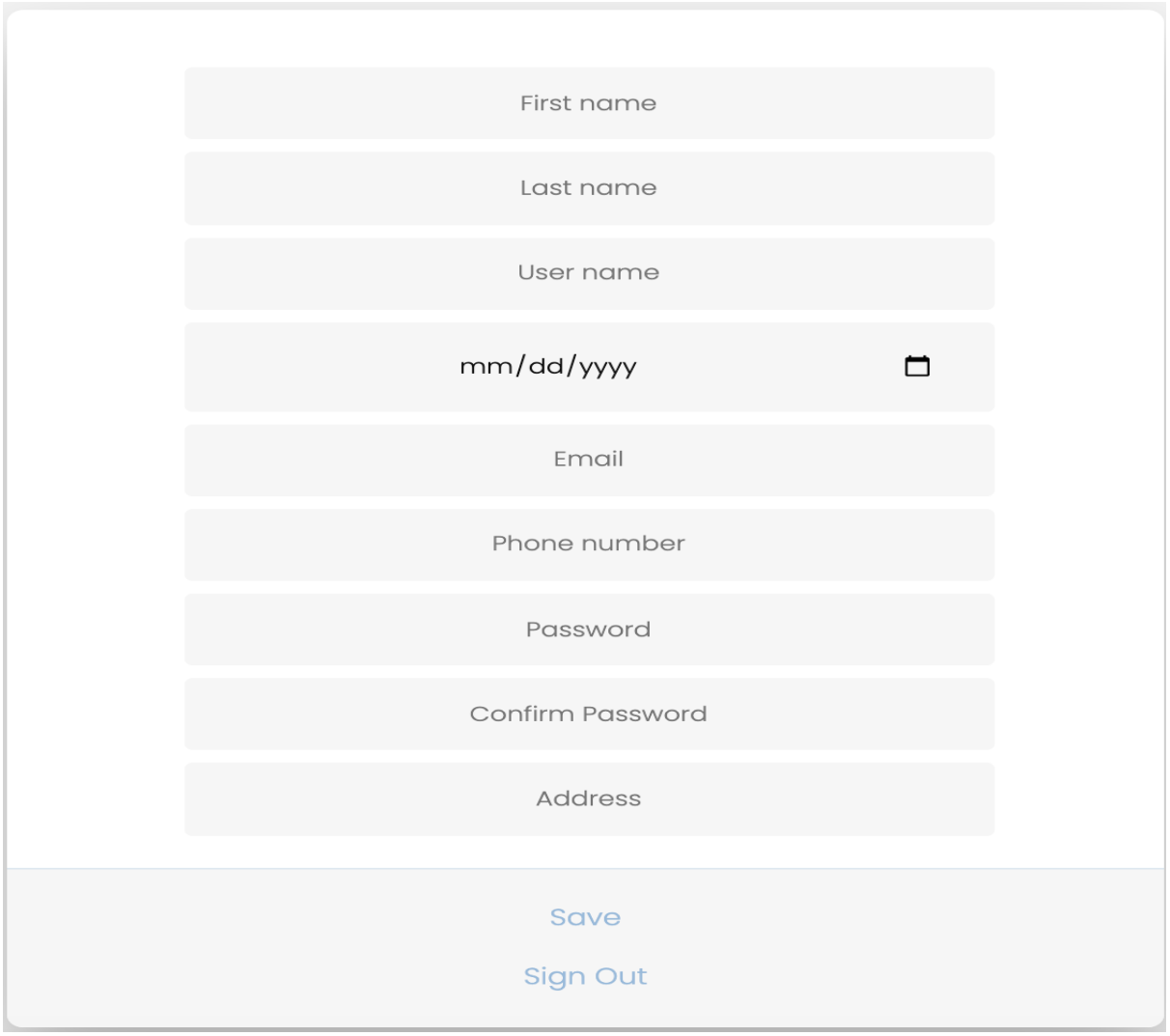


*Figure.10: Userprofile page*

They can either save and go back to shopping or sign out. If they sign out, they will have to log back in in order to use the shopping cart.

## 6. CLASS DIAGRAM

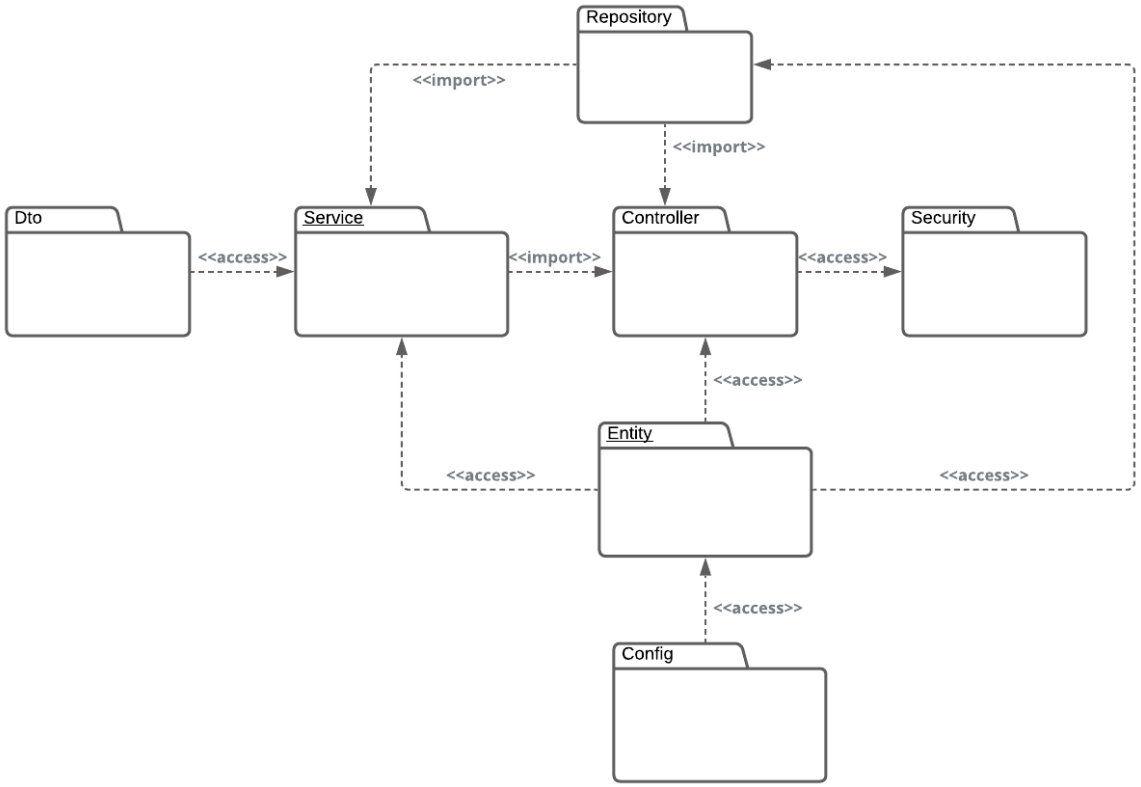


*Figure.11: Package diagram*

As we can see in figure 11, we will have 7 main packages. Each package has its role and support to process, exchange and store information into the database. First, the config folder stores the product information in the database and defines BCryptPasswordEncoder to secure and confirm the account via email for people. Next, the Entity folder will contain the necessary objects for our sales website. In the Service folder, we will have each service class for each object, and this is where we will create the function that handles business logic for the entire system. Then, these functions will be exported to the package controller to process and exchange information between the front-end and the database. Each object will have its controller. Package repository makes it possible for us to use the Query annotation to interact with the database in filtering and updating the information directly. The DTO directory is where data is packaged for transmission between the user-server or between services in the registration and login sections. Finally, package security helps us secure the password and transmission between the back end and the front end.

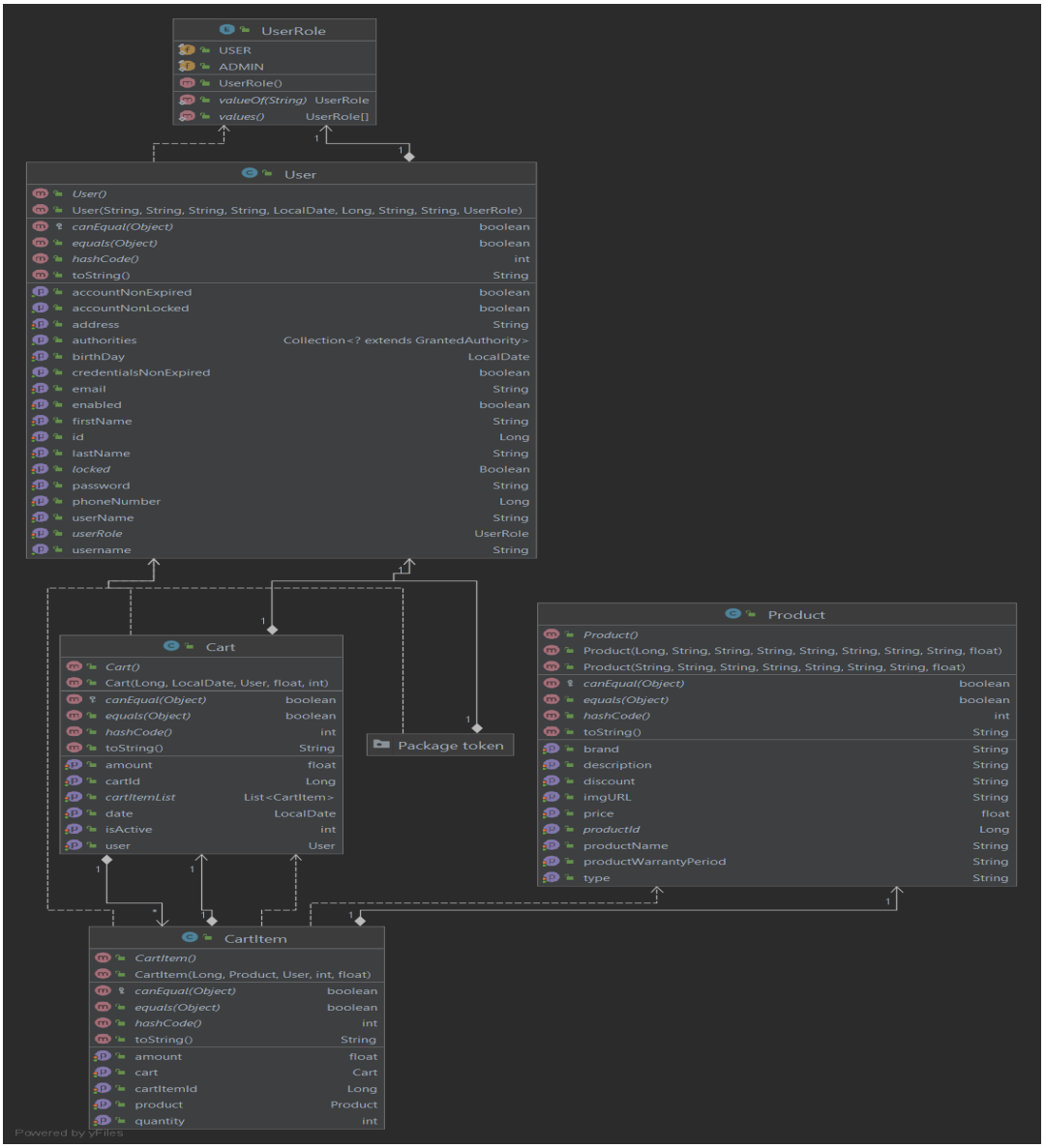


*Figure.12: Object diagram*

Our system has 6 main objects: User, UserRole, ConfirmationToken, Cart, CartItem, and Product. Each object will be asserted with explicit attributes and some getter and setter, constructor, and toString functions. For example, in figure 12, we can realize that one user can create multiple shopping carts. Likewise, a shopping cart can contain multiple cart items, and a product can be placed in multiple carts. To connect relationships between objects, we will use annotation relationships and foreign keys.

## 7. IMPLEMENTATION RESULT

### 7.1. Front-end

When the user decides to checkout, the page will redirect them to the checkout page where they can proceed to buy the item or remove the item from the cart.

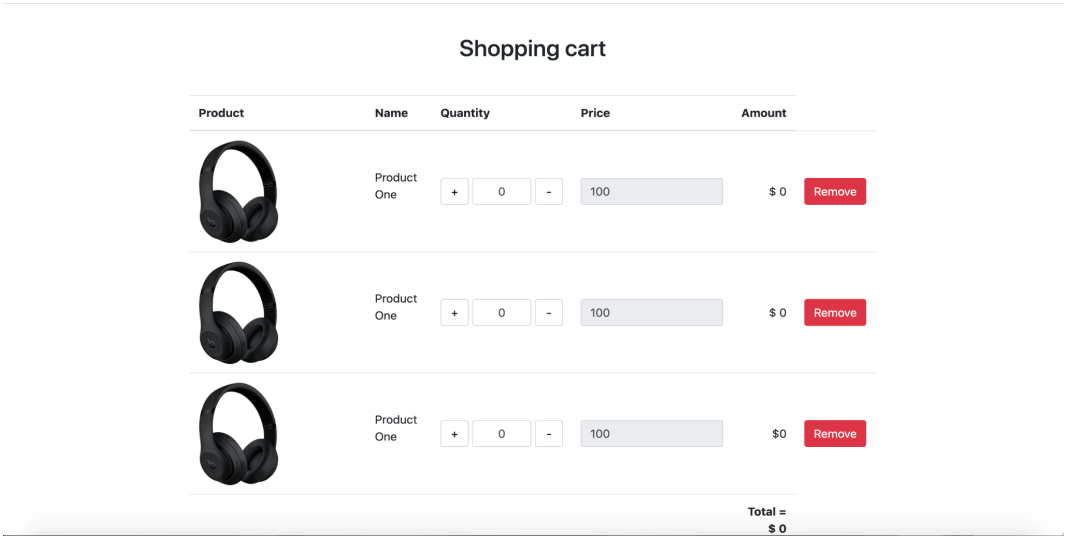


*Figure.13: Checkout cart with a non-updated price*

If the user wants to change the quantity of a product they want to buy, they can increase or decrease the quantity and Javascript will adjust the price based on the quantity they entered.

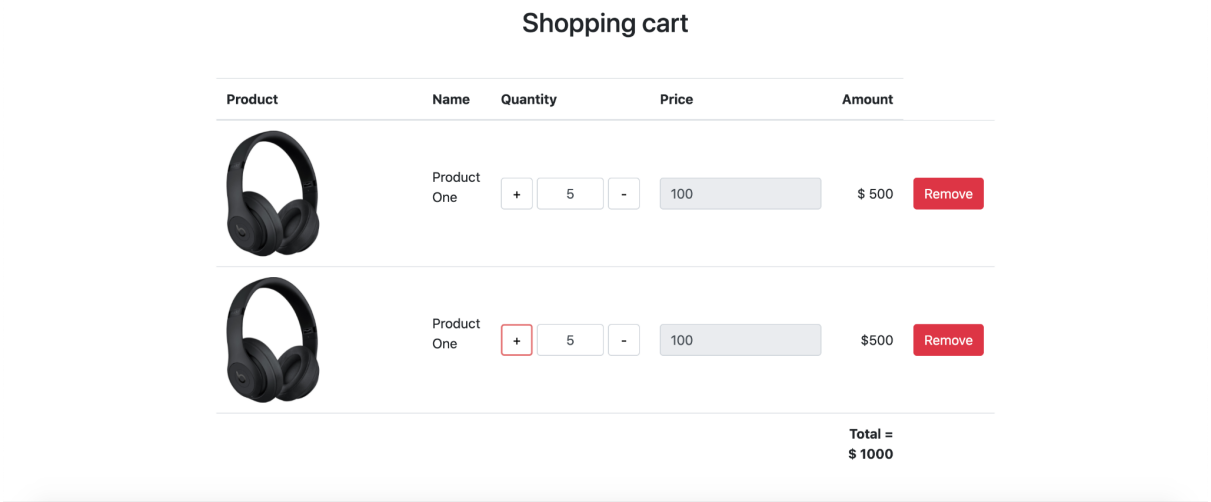


*Figure.14: Checkout cart with updated price*

The moment the user removes one item from the cart, the price will be automatically updated.

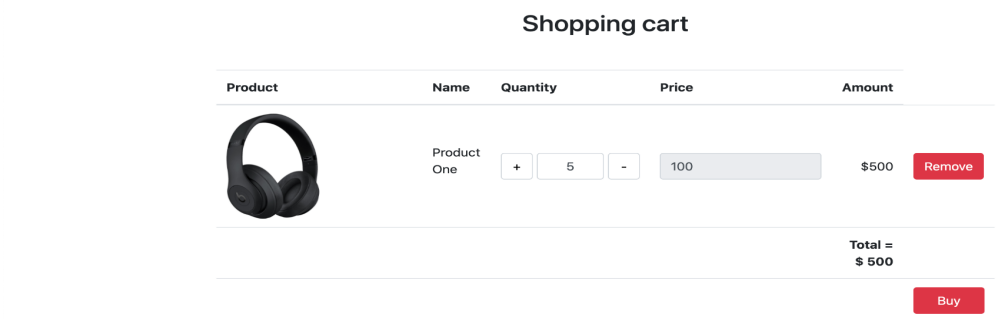


*Figure.15: Checkout cart with fewer items*

When the user wants to check the cart status before signing in, a pop-up said that they need to log in first and be redirected to the login page.

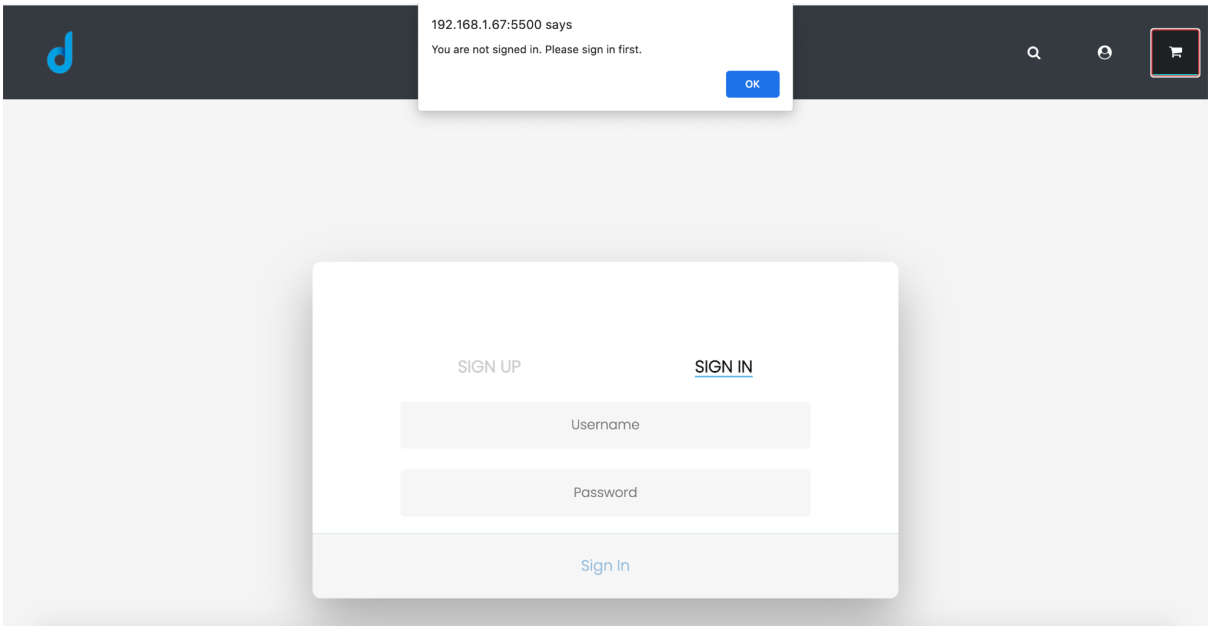


*Figure.16: Request sign-in pop-up*

If the user is already logged in, it will alert the user that they already are.

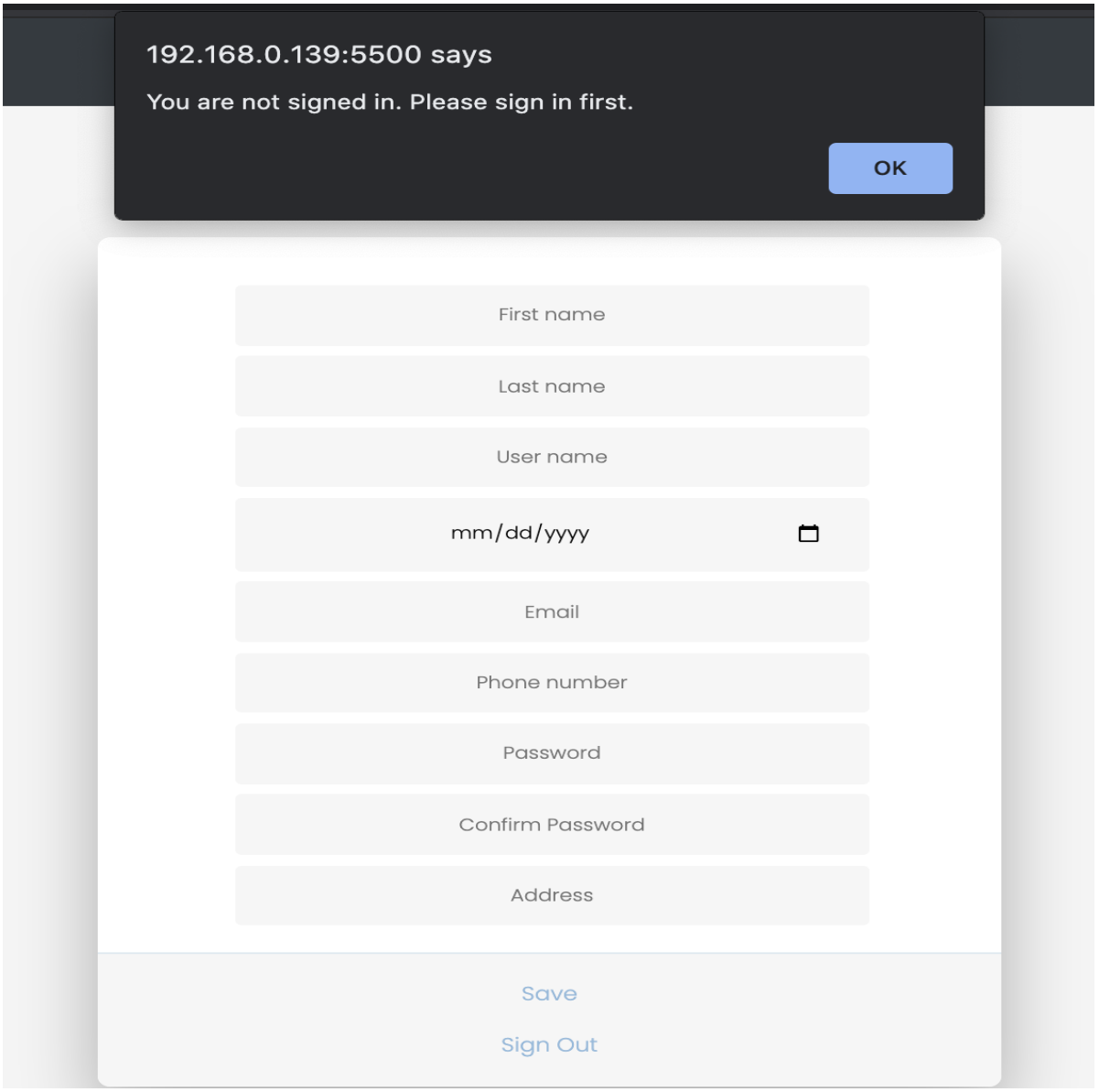


*Figure.17: Request sign-in pop-up [2]*

The user will have to sign in first to access their information page. If they try to access it without signing in or register, they will be redirected to the registration page.

For registration, the user needs to provide detailed information including address and phone number to contact the customer when the order is ready or pickup. As well as an email sent to the user for future items update.

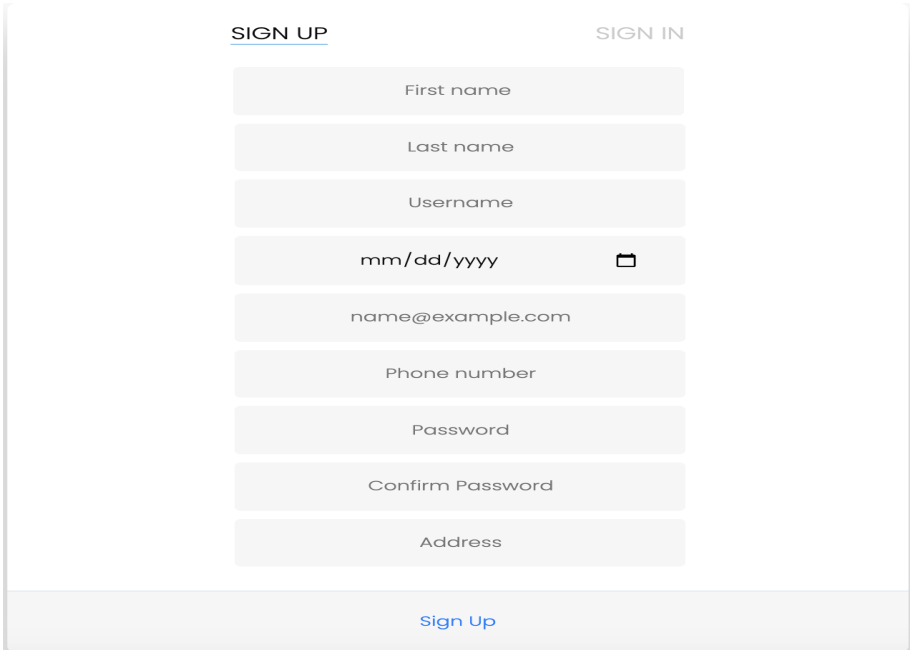


*Figure.18: Registration*

After filling up all the necessary information revolving around the user, they will receive a pop-up stating that they have finished the registration and will receive an email.

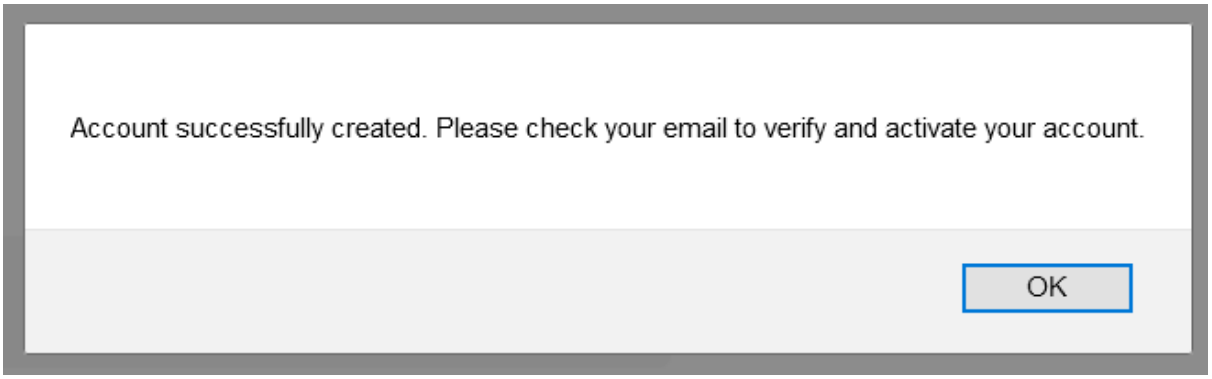


*Figure.19: After registration notification*

When the user finishes registering, they will receive an email to confirm their registration. After that, they can come back to the page and begin shopping.

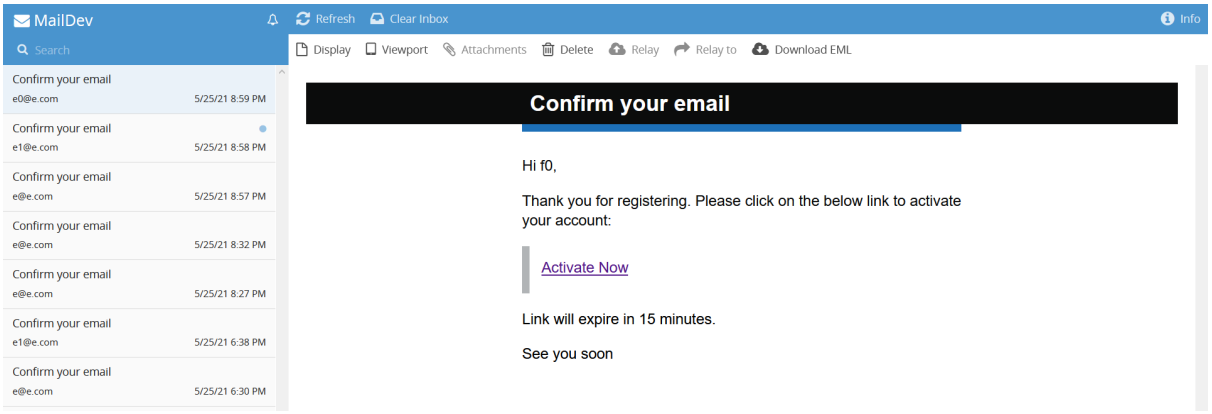


*Figure 20: Email confirmation*

### 7.2. Back-end

Product

To manage and interact with the product information with the front end, we make 6 main functions. First, we create a “product search” function by name, price, and manufacturer. At the same time, we can sort products by price from low to high or vice versa. Second, we create two main functions that load product information based on product name or id to export product information to the front-end. Furthermore, we can also remove and add products.

Cart

We create 4 essential functions in creating and managing customer carts. First, we can create a new cart and show it to the customer. Moreover, we can also show the list of shopping carts that customers have purchased. Finally, we can update the cart total when the customer drops or adds a product to the cart in the front-end.

Cart Item

For convenient shopping cart management, we get 5 essential functions. First, we have a “add a new cart item” function to add the product to the cart. Then, for the customer to see and check the current product in the cart, we create a function showCartItem. At the same time, we also make 2 functions of adding and subtracting the number of products in the cart to update the number of products in the cart and the total amount according to the quantity of each product automatically. Finally, we create the function to remove the product from the cart if the customer does not want to buy that product anymore.

User profile

The management of customer information is critical, so we create 4 main functions for users. The first function is to load the user's information by user name. When users log in or register successfully, they can go to the user profile section to check their information. The following 2 functions are to add a new user to the database and remove the user from the database. To allow the user to update information, we create function update information for the user. When the user successfully logs in and register, we will show their information on the user

profile section. They will change the information they want and click save. The back-end system will automatically save the changed information to the database quickly.

Login

To protect and manage customer information, we will require customers to log in before using our website. Each user will have a unique username, email and password. Our system will check the login information and user information in the database to match or not. If the result is correct, the system will accept the user successfully registered.

Registration

Our system will ask users to provide their complete information during the registration process. To secure and confirm customer information via email, we have created a token used in email account verification and used MailDev to check whether the email account confirmation process is successful or not. After the person confirms the account via email simulated by MailDev, they can easily log in to the website. Each user will have their user name and email, so if the new user enters the same user name and email as the user in the database, the system will report an error.

All back end functionality is tested via postman and on the front end.

## 8. QUALITY ASSURANCE

### 8.1. Testing

| Test case ID | Test case name | Description | Input | Expected result | Actual result (Pass/ Fail) |
|---|---|---|---|---|---|
| 1 | The user logs in with the correct credentials | User system login | Username = Anhkiet Password = anhkiet123 | Log in successfully | Pass |
| 2 | The user logs in with the incorrect username | User system login. Username does not include in database system | Username = AnhkIeT Password = anhkiet123 | Log in unsuccessfully | Pass |
| 3 | The user logs in with the incorrect password | User system login. The password is not included in the database | Username = AnhKiet Password = ANHKIET!@# | Log in unsuccessfully | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | system. | | | |
| 4 | The user logs in with incorrect username and password | User system login. Username and password does not include in database system | Username = ANHKIET Password = ANHKIET!@#!@ | Login unsuccessfully | Pass |
| 5 | The user logs in with empty password | User system login. | Username = AnhKiet Password = | Login unsuccessful | Pass |
| 6 | The user registers with empty password | The user did not enter password when they register | The password is empty | Register unsuccessfully | Pass |
| 7 | The user registers with password confirmation not matching password | The user enters wrong password confirmation | The password is different with the password confirm | Register unsuccessfully | Pass |
| 8 | The user registers successfully | User successfully registered | The user enter information on the register form fully | Register successfully | Pass |
| 9 | The user changes profile data with empty password | The user did not enter password | The password is empty | Register unsuccessfully | Pass |
| 10 | The user changes profile data with password confirmation not matching | The user did not enter password confirm | The password is not match | Register unsuccessfully | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | password | | | | |
| 11 | The user registers with a non-unique username | Username matches with another one | The user enter same the username with another one | Register unsuccessfully | Pass |
| 12 | The user registers with non-unique email | Email matches with another one | The user enter same email with another one | Register unsuccessfully | Pass |
| 13 | The user registers with non-unique username and email | Email and username match with another one | The user enter same email and the username with another one | Register unsuccessful | Pass |
| 14 | The user changes profile data with non-unique username | Email and username match with another one on user profile | The user enter email and username matches with another one on userprofile | User profile update unsuccessfully | Pass |
| 15 | The user changes profile data with non-unique email | Email matches with another one on userprofile | The user enters email matches with another one on userprofile | User profile update unsuccessfully | Pass |
| 16 | The user changes profile data with non-unique username and email | Email and username match with another one on user profile | The users enter email and username matches with another one on user profile | User profile update unsuccessfully | Pass |
| 17 | The user tries to access the cart without logging in | The user did not log in | The user did not log in | They could not purchase any item | Pass |

| | | | | | |
|---|---|---|---|---|---|
| 18 | The user tries to add the product to the cart without logging in | The user did not login | The user did not log in | A windows alert should appear, letting the user know that they are not logged in and need to be logged in first. | Pass |
| 19 | The user tries to add one product twice into cart | A mechanism prevents the user from adding the same product to the same cart twice to avoid duplicates in the database | The user is already logged in.<br><br>Clicking the “add to cart” button more than once for a single product | A windows alert should appear, letting the user know that the product is already in their cart and that they can change the quantity in the cart page. | Pass |
| 20 | The user logs out | When the user is done with their session, they can log out using a button in the user profile page. | Clicking the logout button | A window alert appears telling the user that they have been logged out.<br><br>All session storage data purged.<br><br>User redirected to homepage. | Pass |

**8.2. Bugs and limitations**

Due to time constraints, exhaustive testing is impossible and 100% implementation for the intended features was not reached. This resulted in the following imperfections:

- Storing cart data for non-logged-in users was not implemented in time. This is the reason why the cart is currently only accessible by logged-in users.

- A search bar has been implemented in the front-end, and a search function has been implemented in the back-end, but these two parts have not been connected in time to produce a functioning search function. Instead, the back-end search function is currently used for loading products of a particular type e.g., headphones, smartphones, tablets to display on their own pages.

- The cart page still lacks a function to calculate the total price of the cart and to finish a cart (deactivate cart). Additionally, real-time price adjustments based on quantity of a particular product in cart only works for the first product shown in the list.

- Normally, trying to press the “add to cart” button of a product already in the cart will result in a message saying that the product is already in the cart and the product’s quantity should be changed in the cart page. This function was implemented on two pages, but only worked on one page with seemingly no clear reason as the code for this function was identical on both pages.

- The carousel framework is implemented but currently has no content to display. This resulted in the “Related Products” and “Bestseller” sections of the website showing empty carousels with placeholders.

## 9. CONCLUSION

In conclusion, this report will help readers better understand the relationship between the pages in this project. First, the front-end team explains the pages we've designed. Besides, Team Back-end also clearly explains the database system and handles business logic applied to this online purchasing website. In this project, we used the most popular languages to create a destination, an e-commerce site with full features for users. For the back-end, we use Springboost and PG admin 4, and Postman to make those features perfect. However, the website still has some limitations that we have not been able to overcome. We will add some features and fix those limitations we mentioned if given more time. Through this project, we have gained experience and skills in building a complete website from Front-end to Back-end.

## 10. REFERENCES

[1] "How COVID-19 triggered the digital and e-commerce turning point | UNCTAD", *Unctad.org*, 2021. [Online]. Available: https://unctad.org/news/how-covid-19-triggered-digital-and-e-commerce-turning-point. [Accessed: 25- May- 2021].

[2]"Java EE | Java Enterprise Edition - Javatpoint", *www.javatpoint.com*, 2021. [Online]. Available: https://www.javatpoint.com/java-ee. [Accessed: 25- May- 2021].

[3]"PostgreSQL", *PostgreSQL*, 2021. [Online]. Available: https://www.postgresql.org/. [Accessed: 25- May- 2021].

[4]"What is Software Framework? - Definition from Techopedia", *Techopedia.com*, 2021. [Online]. Available: https://www.techopedia.com/definition/14384/software-framework. [Accessed: 25- May- 2021].

[5]"Learn Spring Boot | Baeldung", *Baeldung*, 2021. [Online]. Available: https://www.baeldung.com/spring-boot. [Accessed: 25- May- 2021].

[6]"What Is Spring Boot?", *Stackify*, 2021. [Online]. Available: https://stackify.com/what-is-spring-boot/. [Accessed: 25- May- 2021].

[7]V. Code, "Why Visual Studio Code?", *Code.visualstudio.com*, 2021. [Online]. Available: https://code.visualstudio.com/docs/editor/whyvscode. [Accessed: 25- May- 2021].

[8]"Features - IntelliJ IDEA", *JetBrains*, 2021. [Online]. Available: https://www.jetbrains.com/idea/features/#:~:text=IntelliJ%20IDEA%20analyzes%20your%20code,and%2C%20of%20course%2C%20refactorings. [Accessed: 25- May- 2021].

[9]"Postman API Testing | Features | Benefits | How to Use?", *Teknotrait Solutions*, 2021. [Online]. Available: https://teknotrait.com/postman-api-testing-features-benefits/. [Accessed: 25- May- 2021].

[10]"Getting Started with PGAdmin on a Distributed SQL Database - The Distributed SQL Blog", *The Distributed SQL Blog*, 2021. [Online]. Available: https://blog.yugabyte.com/getting-started-with-pgadmin-on-a-distributed-sql-database/. [Accessed: 25- May- 2021].

[11]"What Is GitHub, and What Is It Used For?", *How-To Geek*, 2021. [Online]. Available: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/. [Accessed: 25- May- 2021].

## 11. APPENDIXES

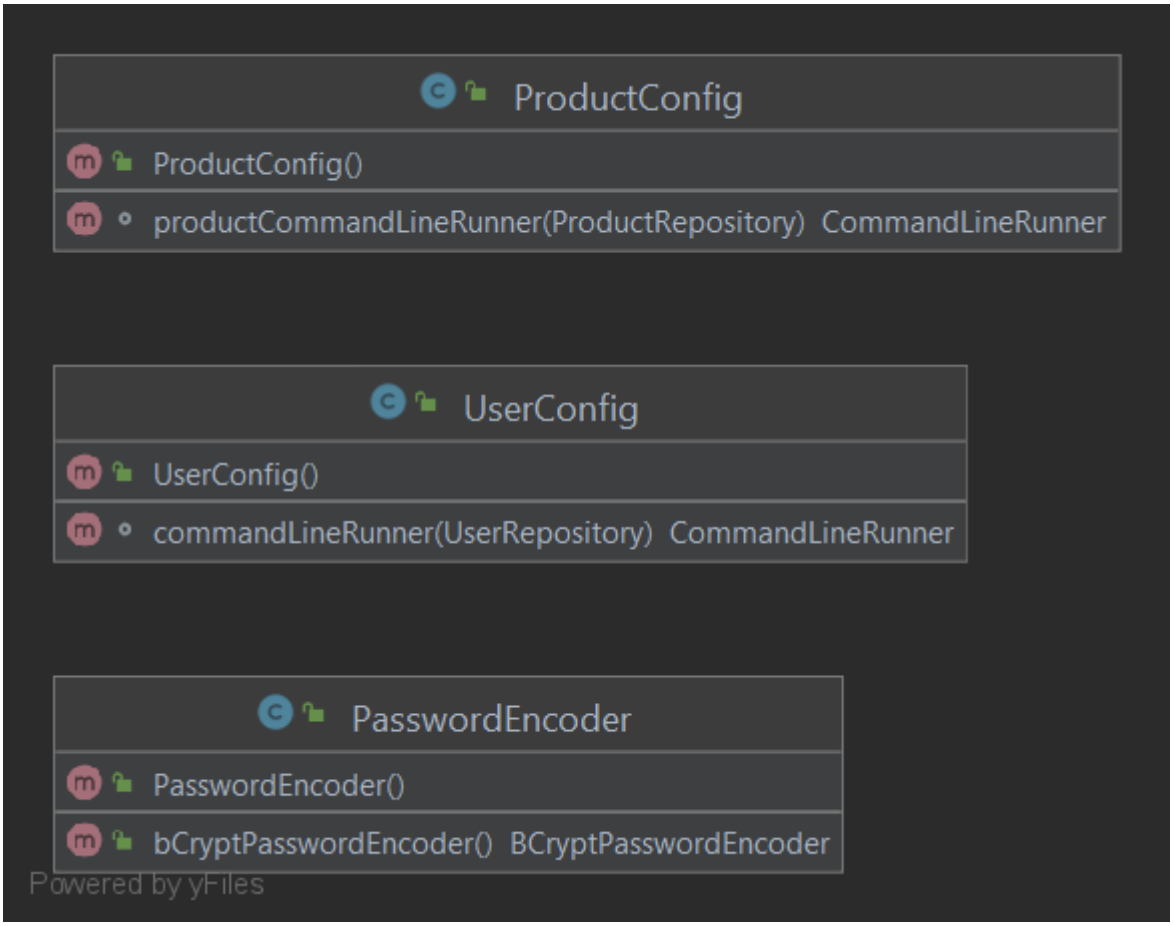


*Figure 21: Appendix 1 Configuration class diagram*

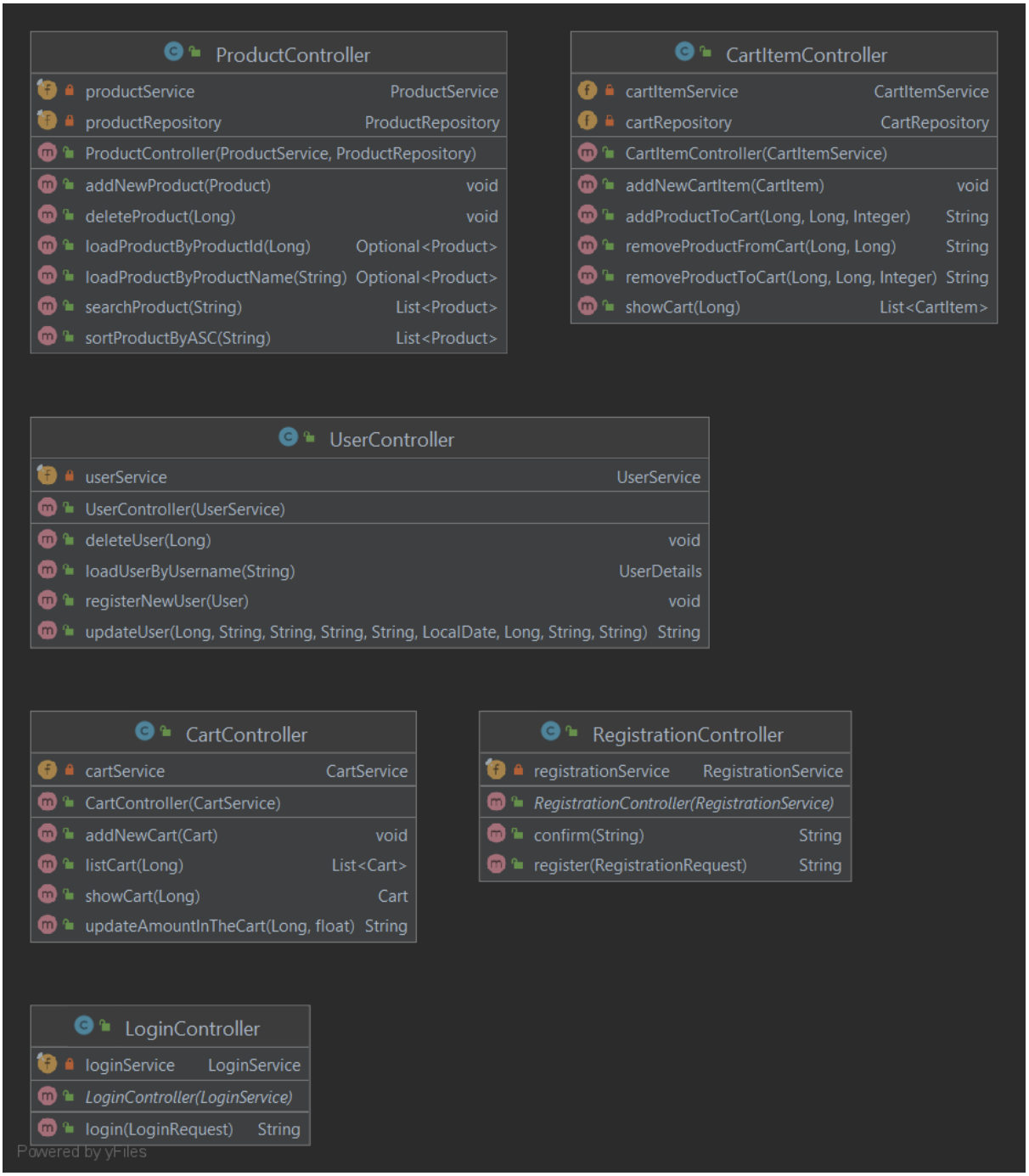


*Figure 22: Appendix 2  Controller class diagram*

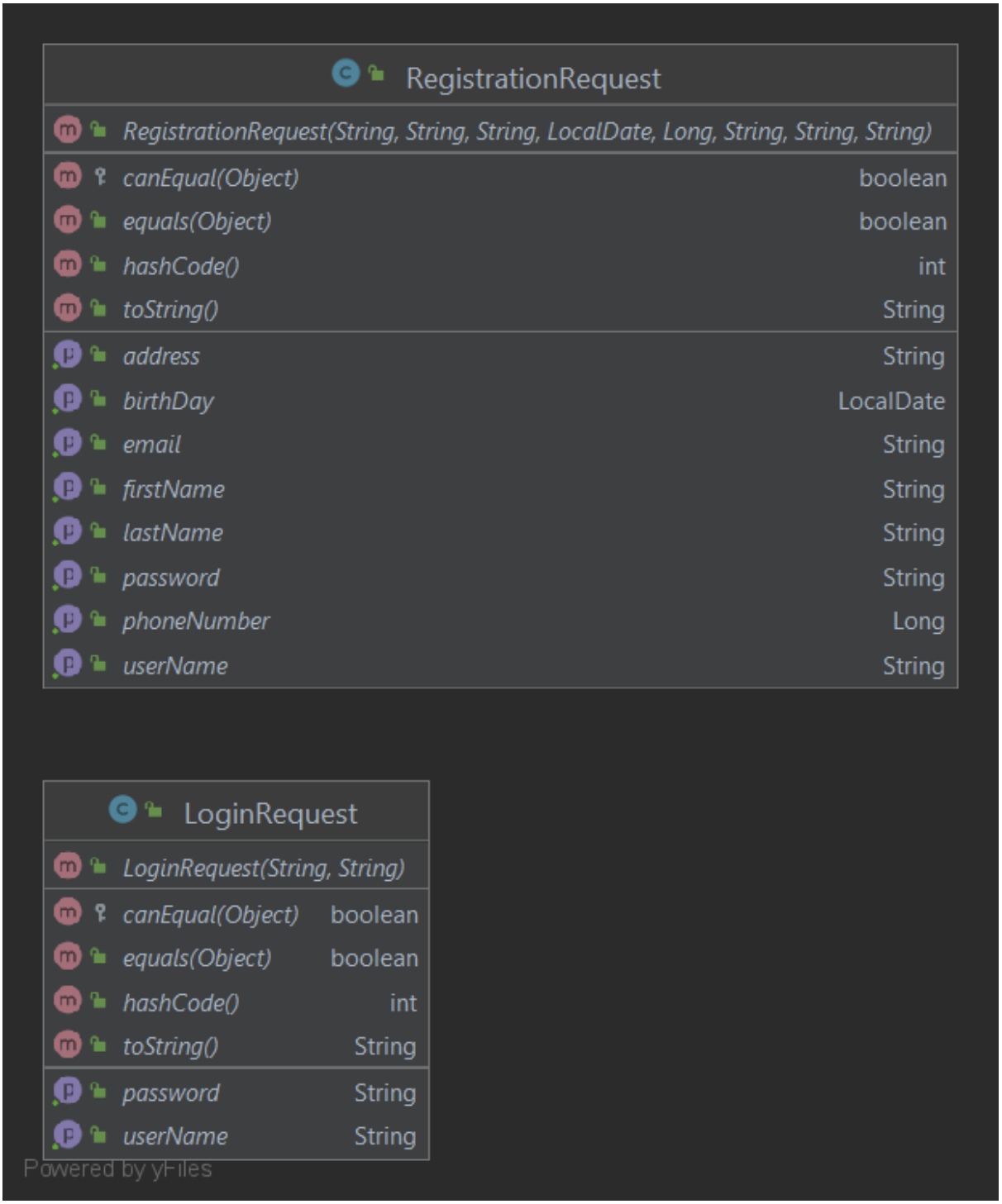


*Figure.23: Appendix 3 Dto class diagram*

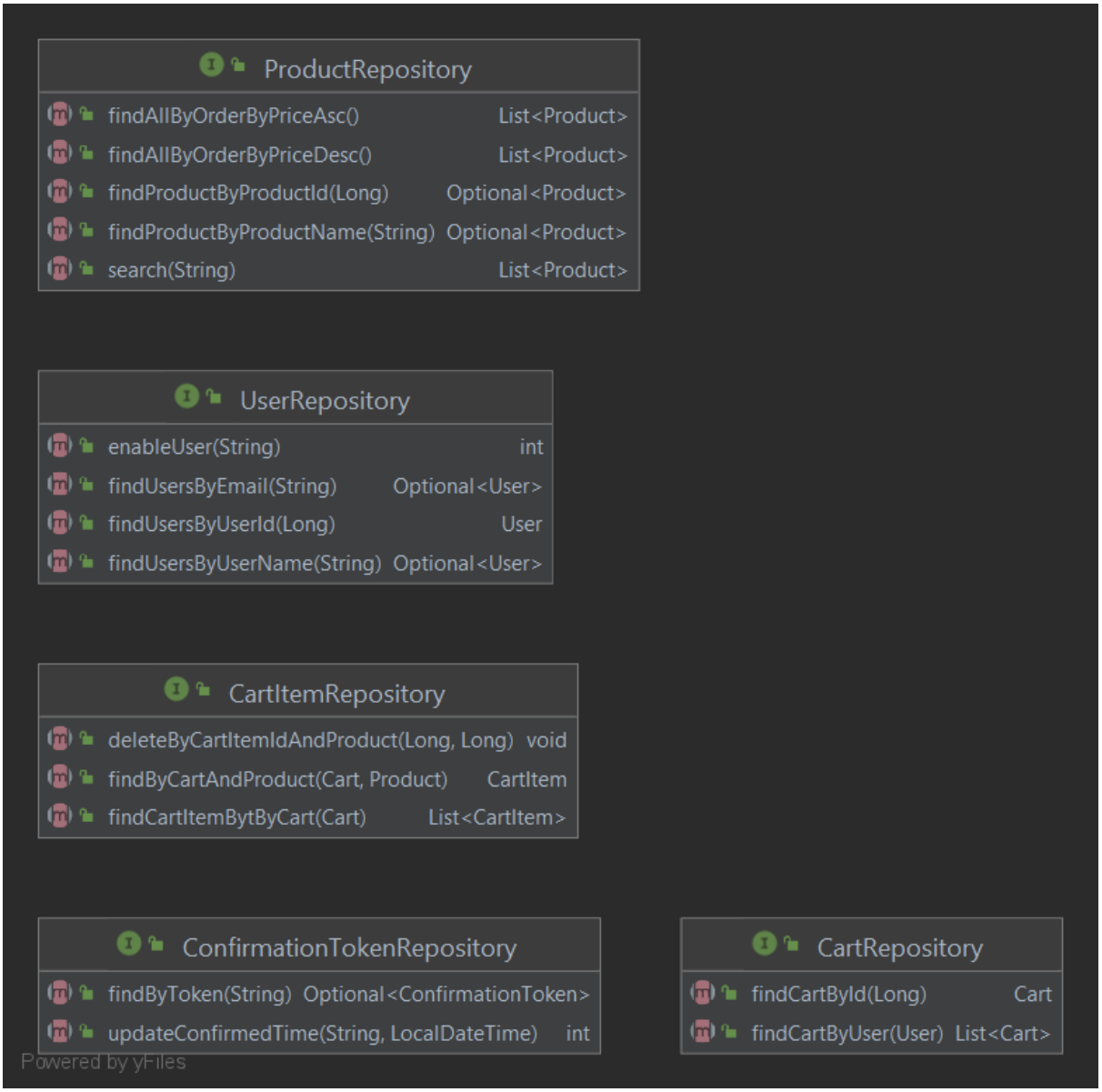


*Figure.24: Appendix 4 Repository class diagram*

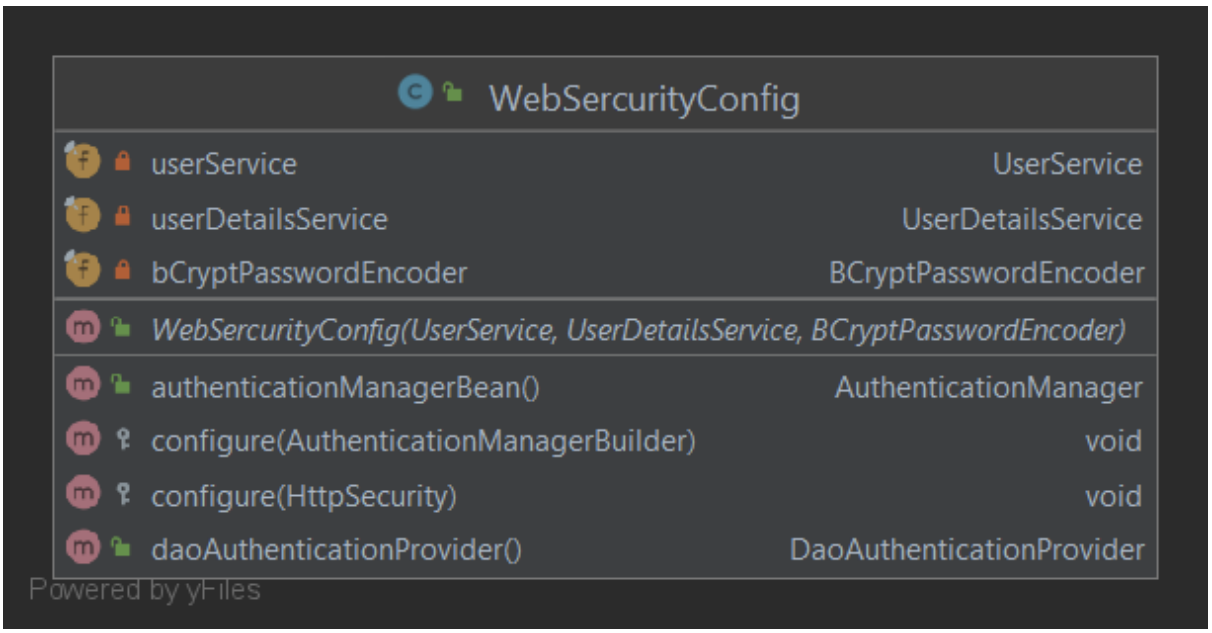


*Figure.25: Appendix 5 Security class diagram*

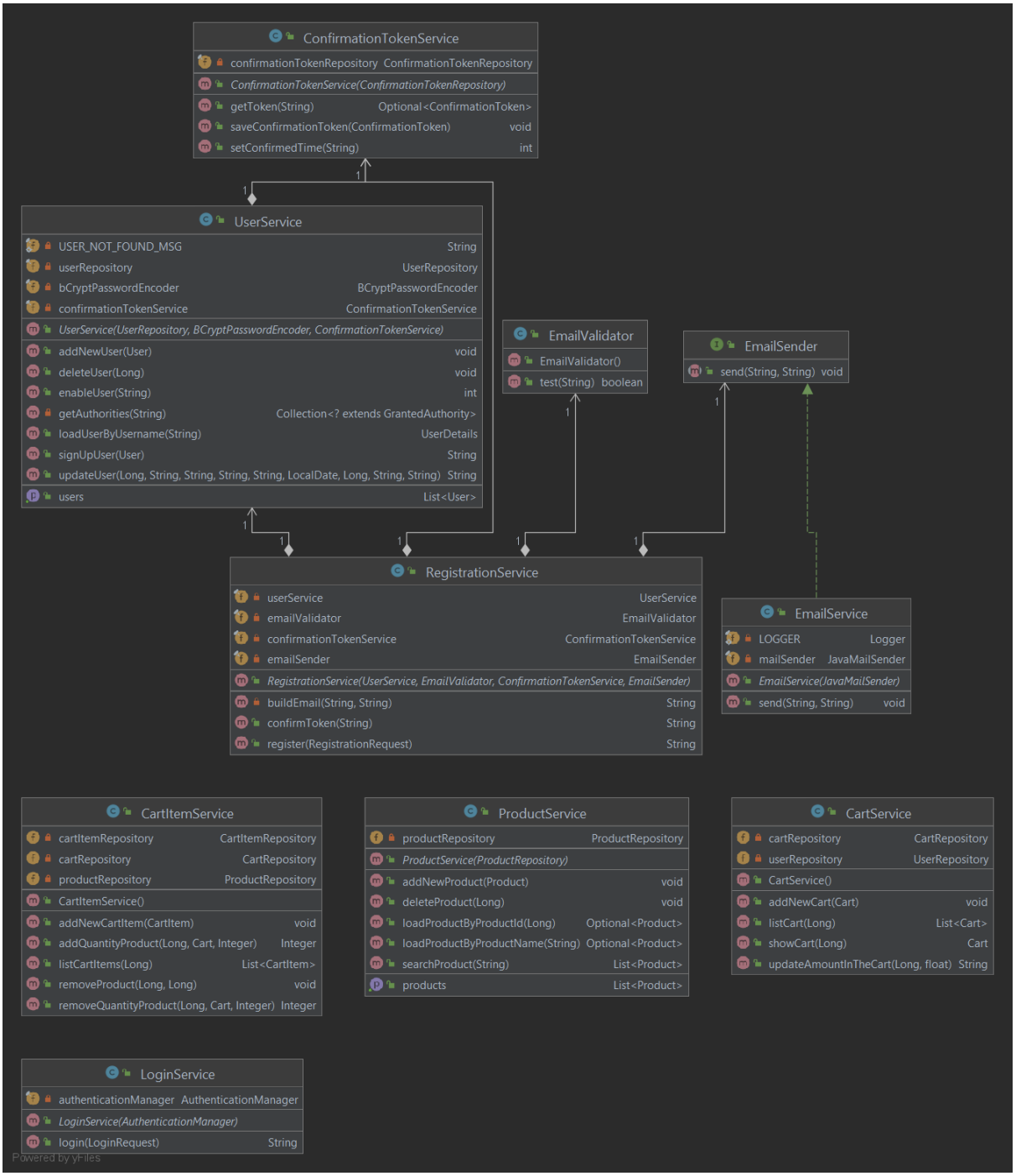


*Figure.26: Appendix 6 Service class diagram*

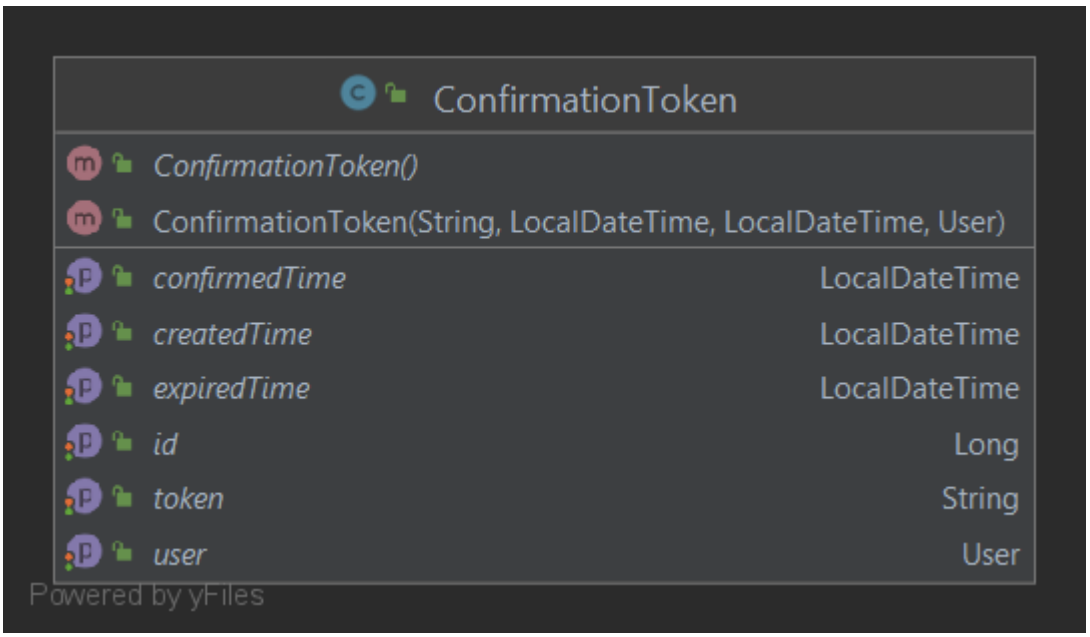


*Figure.27: Appendix 7 Confirmation token class diagram*