Group 07

Honie Cosmetic
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
12/12/2022	1.0	The first update.	Team Leader

Table of Contents

Software Architecture Document

1. Introduction

- This document provides an overview of Honie Cosmetic Web software architecture based on the understanding of technical requirements elicited from use cases presented by group 07.
- The document provides a description of the goals of the architecture, the use cases supported by the system and architectural styles and components that have been selected to best achieve the use cases.

1.1 Purpose

- This Software Architecture Document (SAD) provides an architectural overview of Honie Cosmetic in order to capture service level requirements of the platform. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

- This SAD presents the structure and behavior of the entire software. It only covers the first level of decomposition of the platform into its major components.

1.3 Abbreviations, Acronyms and Definitions

Abbreviation	Description
HTTP	Hypertext Transfer Protocol
SAD	Software Architecture Document

Term	Definitions
Cloud	Ubiquitous network access
Operator	User that interacts with the system when it is running

1.4 References

Software Architecture Document	CREATE: Software Architecture Document – ITEA4

2. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture:

- ❖ The system must be able to manage the number of users accessing the website.
- ❖ All registered users and products posted on the website must be in the system's data.
- ❖ System must guarantee that data is completely protected from unexpected access. The system must have a user verification mechanism corresponding to the accounts registered in the system.
- ❖ Allow users to manage the process of ordering, payment, checkout and order in the most effective and simple way.

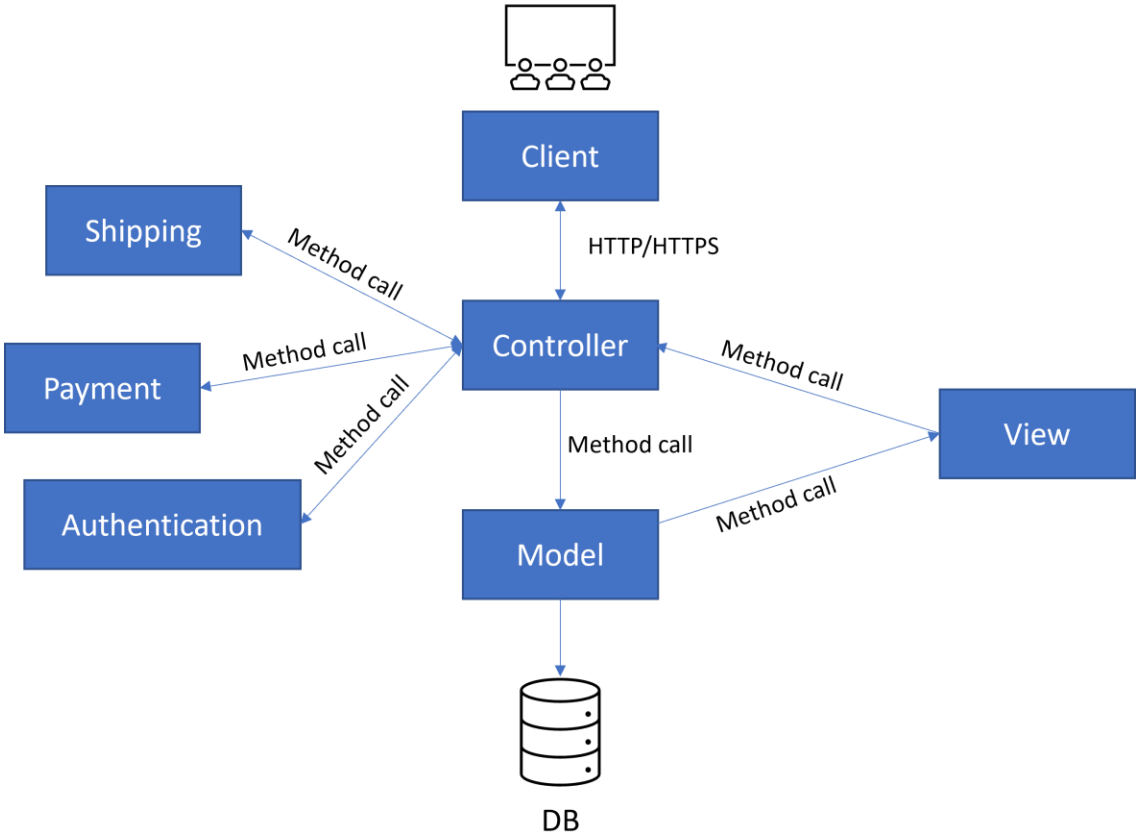
- ❖ The system must have a reasonable architecture to ensure full completion of the functions in the Use-case Specification document. In which the online payment function is a difficult function.

The major design and implementation constraints for the application are:

- ❖ Simplicity
- ❖ Flexibility
- ❖ Security
- ❖ Maintainability: Thriving the website maintenance and identifying and resolving system failures in the future to maintain the website.
- ❖ Scalability (Khả năng mở rộng): Can grow and increase the features and functionality of the website without affecting the performance of the website (add more memory, servers, or disc space for making more transactions on the website).
- ❖ Correctness: Extent to which program satisfies specifications and fulfills user's mission objectives.
- ❖ Usability:
  - Easy to use, efficient, and accessible.
  - Users should be helped appropriately to fill in the mandatory fields, in case of invalid input.
  - Keep track of documentation, activities, and responses.

3. Use-Case Model

4. Logical View



a) Client-Server Pattern

- Clients initiate interactions with servers, which provide a set of services. The clients invoke services as needed from those servers, and then wait for the results of those requests. The client is responsible for displaying and performing small updates on the data, while the server handles data management. The client-server pattern supports modifiability and reuse, as it factors out common services, allowing them to be modified in a single location. This pattern also supports scalability and availability by centralizing the control of these resources and servers.

b) Model-View-Controller Pattern

- The MVC pattern separates user interface functionality from application functionality. With MVC, application functionality is divided into three types of components:
  - Models, which contain the application data.
  - Views, which display the underlying data and interact with the user.
  - Controllers, which mediate between the model and the view and manage state changes.
- The MVC pattern supports usability, as it allows the user interface to be designed and implemented separately from the rest of the application.

❖ ***Client:***

- Client is a component that invokes services of a server component. Clients have ports that describe the services they require. In the context of the Coop Evaluation System, the client is a web browser being used to access the system. The client makes HTTP

requests using RESTful web services.

- Client will provide the user with a graphical interface for interacting with the system.
- It will be composed of HTML, CSS, JavaScript, and other related files that will run in the user's web browser of choice. In addition, my project uses some frameworks like Bootstrap to layout and ReactJS. The reason why we use above technologies is useful, larger development community, …

❖ ***Controller:***

- The controller contains all the business-related logic and handles incoming requests. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view. The server acts as a RESTful API server, providing interfaces to receive requests from the client and call to corresponding services.

❖ ***Model:***

- Contains all the business logic for the application, including logic for how to "build" itself, whether via database queries or some other method.
- The model contains services to prepare data and send it back to the Controller.

❖ ***Database:***

- Store, access and update data and many other applications. The database helps to manage the security and recovery services of the data management system, helping to enforce constraints within this underlying system.

❖ ***External API:***

- An open or external API is an API designed to be accessed by a larger population as well as web developers, admin, owner. This implies that the external API can easily be used by developers inside the organization (who published the API) and any other external developers who wish to subscribe to the interface.

c) Concerning the technologies chosen:

❖ ***Client:***

- React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components". HTML, CSS and JavaScript are the based components to build websites and we are using Bootstrap to layout our website.
- The reason for choosing ReactJS:
    - React is currently one of the most popular JavaScript libraries, with a solid foundation along with a strong developer community.
    - React uses all Components.
    - React can speed up apps, page load times. React applications are also easy to maintain and debug.

❖ ***Server:***

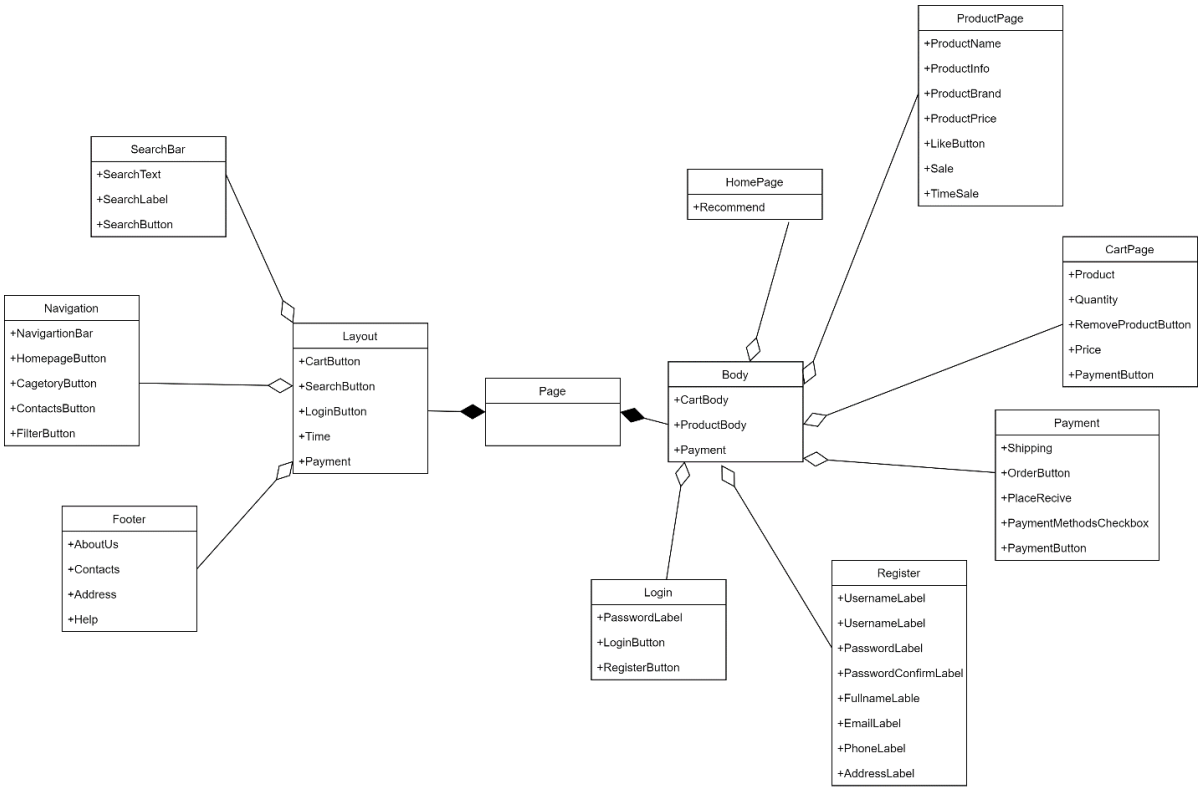
- Server is a component that receives requests and handles requests sent by the client, developed on NodeJS environment. Node.js is a cross-platform open-source runtime environment and library used to run web applications outside of the client's browser. In Addition, we use the framework ExpressJS. Express is a Nodejs web application framework that provides broad features for building web and mobile applications.
- The reason for choosing ExpressJS:

- ExpressJS supports HTTP and middleware methods, creating an extremely powerful and easy-to-use API.
- ExpressJS helps organize web applications into a more organized MVC architecture

❖ ***Database:***

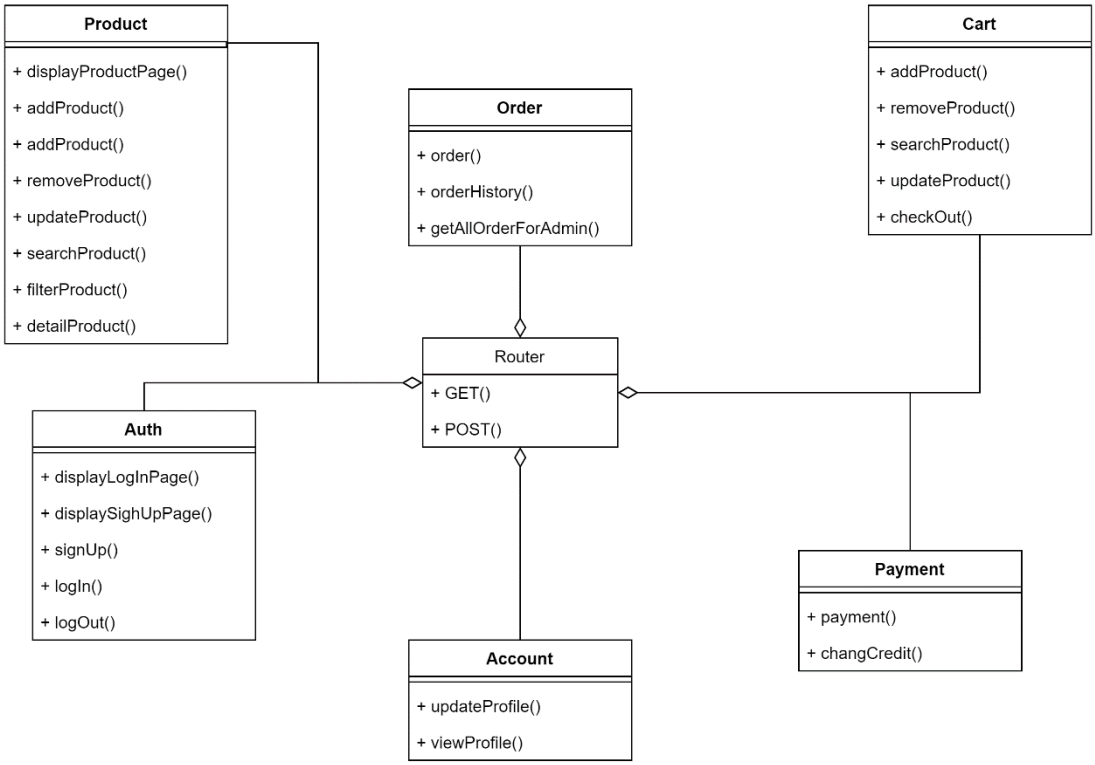
- MySQL is one of the RDBMS software. RDBMS and MySQL are often thought of as one because of MySQL's immense popularity. The biggest web apps like Facebook, Twitter, YouTube, Google, and Yahoo! All use MySQL for data storage purposes. Even though it was initially used for very limited use, it is now compatible with many important computing platforms such as Linux, macOS, Microsoft Windows, and Ubuntu.
  - High performance: Many server clusters use MySQL. No matter if you store large data of e-commerce sites or heavy business related to information technology, MySQL can handle it smoothly, high speed.
  - Flexible and easy to use: You can modify the source code to meet your needs at no additional cost
  - Security: Data safety is always the most important issue when choosing RDBMS software. With its decentralized access and account management system, MySQL sets very high security standards. Encryption of credentials and authentication from the host are both available.

4.1 Component Client



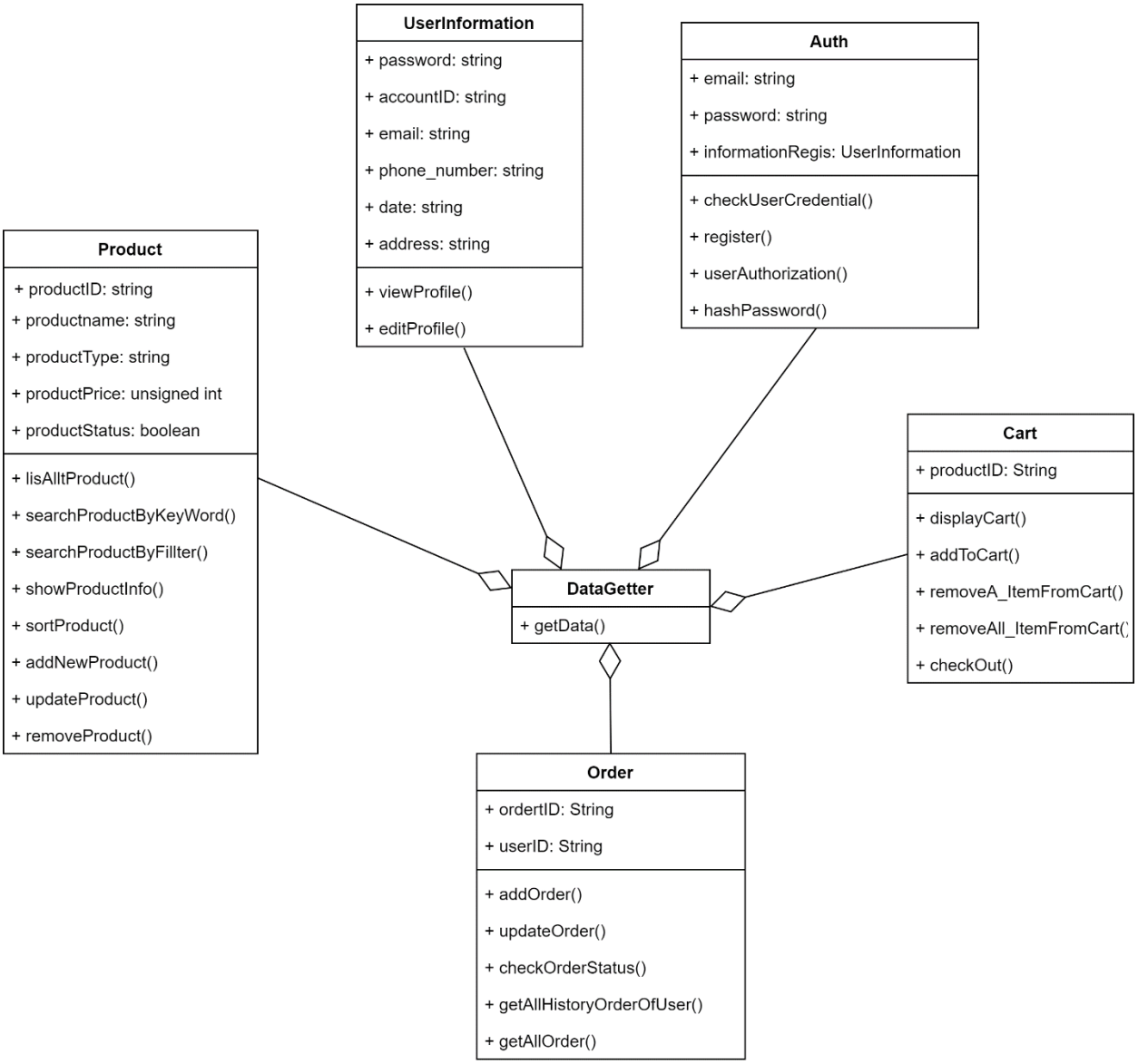
- ❖ Client class represents types of objects residing in the UI component.
- ❖ A UI includes the content of layout and body. The layout can contain ingredients such as search bar, navigation bar, footer, … The body display the content of interfaces for different purpose like product page is a collection of pages related to the product (display detail of a product, display list of product and so on).

4.2 Component: Controller



- ❖ Controller class includes controller components: Product, Auth, Order, Account, Payment and Cart. After receiving a request from client, the route will navigate to corresponding controller. After that, call the services in the model and receive data by model and view receive.
  - Product Controller: call to the services corresponding product such as display product page, detail product, add product, update product, …
  - Auth Controller: call to the services corresponding authentication and authorization such as login, registration, …
  - Order Controller: call to the services like order process, display order history or get all orders for admin to manage.
  - Account Controller: call the corresponding services account.
  - Payment Controller: payment, change credit, ...
  - Cart Controller: add item to cart, remove item from cart, update and check out.

4.3 Component: Model



- Model class have the services corresponding components. When the model was called by the controller, its process, query and perform operations with the database to get data corresponding to each request. Like the controller, the model also have a different component: Product, User Information, Auth, Cart and Order. Each component performs various services.

5. Deployment

★ *This section will be updated in the next sprint.*

6. Implementation View

★ *This section will be updated in the next sprint.*