Supplement and Fitness Shop

Software Architecture Document

Version <1.1>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/11/2024 | <1.0> | Overview of Software Architecture | Hoàng Văn Khải  Nguyễn Đình Kiên |
| 5/12/2024 | <1.1> | Add Deployment and Implementation View  Rewrite Logical View | Hoàng Văn Khải |
|  |  |  |  |
|  |  |  |  |

**TABLE OF CONTENTS**

[1. Introduction 4](#_Toc184438771)

[1.1 Purpose 4](#_Toc184438772)

[1.2 Scope 4](#_Toc184438773)

[1.3 Definitions, Acronyms and Abbreviations 4](#_Toc184438774)

[1.4 References 4](#_Toc184438775)

[2. Architectural Goals and Constraints 4](#_Toc184438776)

[2.1 Goals 4](#_Toc184438777)

[2.2 Constraints 4](#_Toc184438778)

[3. Use-Case Model 5](#_Toc184438779)

[4. Logical View 6](#_Toc184438780)

[4.1 Views Part 7](#_Toc184438781)

[4.1.1 Component: NavBar 7](#_Toc184438782)

[4.1.2 Component: Footer 7](#_Toc184438783)

[4.1.3 Component: Home 7](#_Toc184438784)

[4.1.4 Component: About 7](#_Toc184438785)

[4.1.5 Component: Profile 7](#_Toc184438786)

[4.1.6 Component: Products 8](#_Toc184438787)

[4.1.7 Component: Shopping Cart 8](#_Toc184438788)

[4.1.8 Component: PTs 9](#_Toc184438789)

[4.1.9 Component: Forum 9](#_Toc184438790)

[4.1.10 Component: SignUp 9](#_Toc184438791)

[4.1.11 Component: SignIn 9](#_Toc184438792)

[4.2 LogicsPart 10](#_Toc184438793)

[4.2.1 Component: ProfileLogic 10](#_Toc184438794)

[4.2.2 Component: ProductLogic 10](#_Toc184438795)

[4.2.3 Component: CartLogic 10](#_Toc184438796)

[4.2.4 Component: PaymentLogic 10](#_Toc184438797)

[4.2.5 Component: PTLogic 10](#_Toc184438798)

[4.2.6 Component: ForumLogic 10](#_Toc184438799)

[4.2.7 Component: SignUpLogic 10](#_Toc184438800)

[4.2.8 Component: SignInLogic 10](#_Toc184438801)

[4.3 Database 10](#_Toc184438802)

[5. Deployment 11](#_Toc184438803)

[5.1 Deployment Diagram(s) 11](#_Toc184438804)

[5.2 Nodes in diagram 11](#_Toc184438805)

[5.3 Connection 12](#_Toc184438806)

[6. Implementation View 12](#_Toc184438807)

Software Architecture Document

# Introduction

## Purpose

The purpose of this document is to define the software architecture for the Supplement and Fitness Shop system. It provides a high-level overview of the system’s architecture, including its main components, their interactions, and the underlying principles and guidelines. This document serves as a reference for stakeholders, including developers, testers, project managers, and maintainers, ensuring alignment and consistency throughout the software development lifecycle.

## Scope

This Vision Document applies to the Supplement and Fitness Shop web, which will be developed by the Nao Co Bap team. The team will develop this web-based system to facilitate the sale and promotion of supplements and fitness-related products, integrating with an existing product inventory database.

## Definitions, Acronyms and Abbreviations

* **API** (Application Programming Interface): is a set of rules and protocols that allows one software application to communicate with another.
* **Stakeholder:** Individuals or groups interested in the project, including developers, users, and managers.

## References

Applicable references are:

1. PA3 instuction file
2. Viblo, API Gateway là gì? Tại sao một hệ thống microservices lại cần API Gateway?, link: <https://viblo.asia/p/api-gateway-la-gi-tai-sao-mot-he-thong-microservices-lai-can-api-gateway-Do754pDX5M6>, published: 15/1/2019.
3. Example: Software Architecture Document, link: <https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm>, last accessed: 20/11/2024.

# Architectural Goals and Constraints

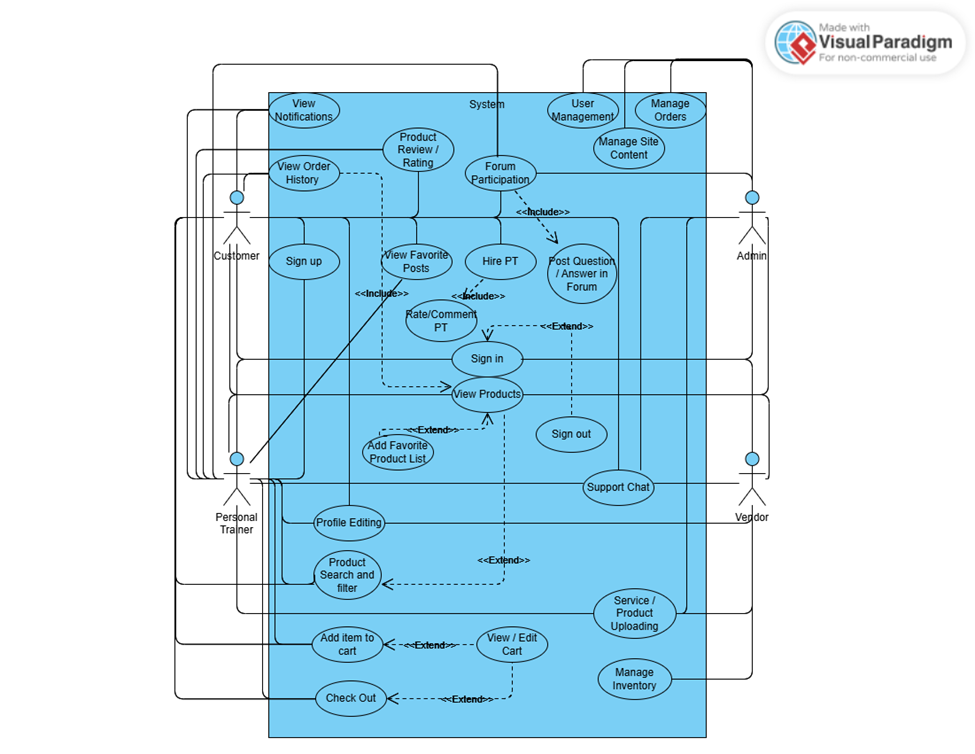
## Goals

* **Responsive Design:** The website must respond to any user action within 1 second to ensure a smooth and fast user experience.
* **Platform Independence:** The website should run seamlessly on all major browsers, operating systems, and devices (e.g., desktops, laptops, tablets, and smartphones).
* **High Availability:** The website must be accessible 24/7 without interruptions to provide continuous service to users.
* **User-Friendly Interface:** The design of the website must be intuitive and visually appealing to attract and retain users. Features must be straightforward and easy for all users, regardless of technical knowledge.
* **Robust Security:** Ensure strong protection of users' personal data and sensitive information to prevent breaches or unauthorized access.
* **Scalability:** The system should support increasing amounts of data and more concurrent users as the website grows.
* **Ease of Maintenance:** The system architecture must be simple to maintain, update, and repair without significant downtime or complexity.

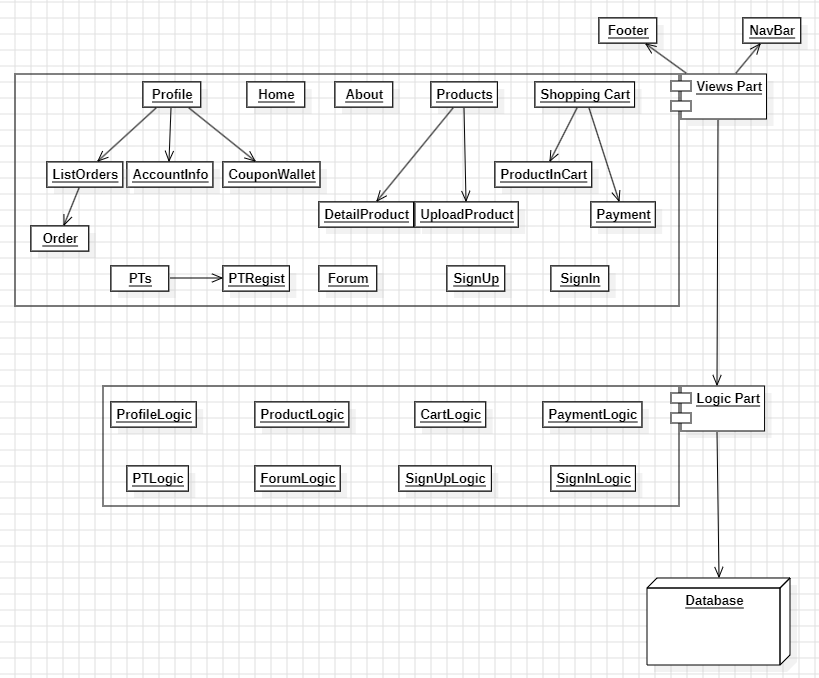
## Constraints

* **Downtime Limitation:** In the event of maintenance, the downtime must not exceed 12 hours.
* **Performance Limits:** The system should reliably support at least 10 concurrent users without performance degradation.
* **Database Requirements:** The database must be capable of storing extensive website data and user information efficiently.
* **Legal Compliance:** All content and functionalities on the website must comply with applicable legal standards and regulations.
* **Security Standards:** Implement strong encryption for data storage and communication, following industry best practices (e.g., HTTPS, password hashing, and secure access controls).
* **Budget and Resources:** The architectural choices should be feasible within the given budget, development time, and available resources.
* **Design Constraints:** The system must ensure compatibility with modern web standards (HTML5, CSS3, etc.). UI/UX designs should adhere to accessibility guidelines (e.g., WCAG 2.1).
* **Hardware and Network Constraints:** The solution should perform efficiently on typical user devices and network conditions without requiring high-end hardware or extremely fast internet connections.

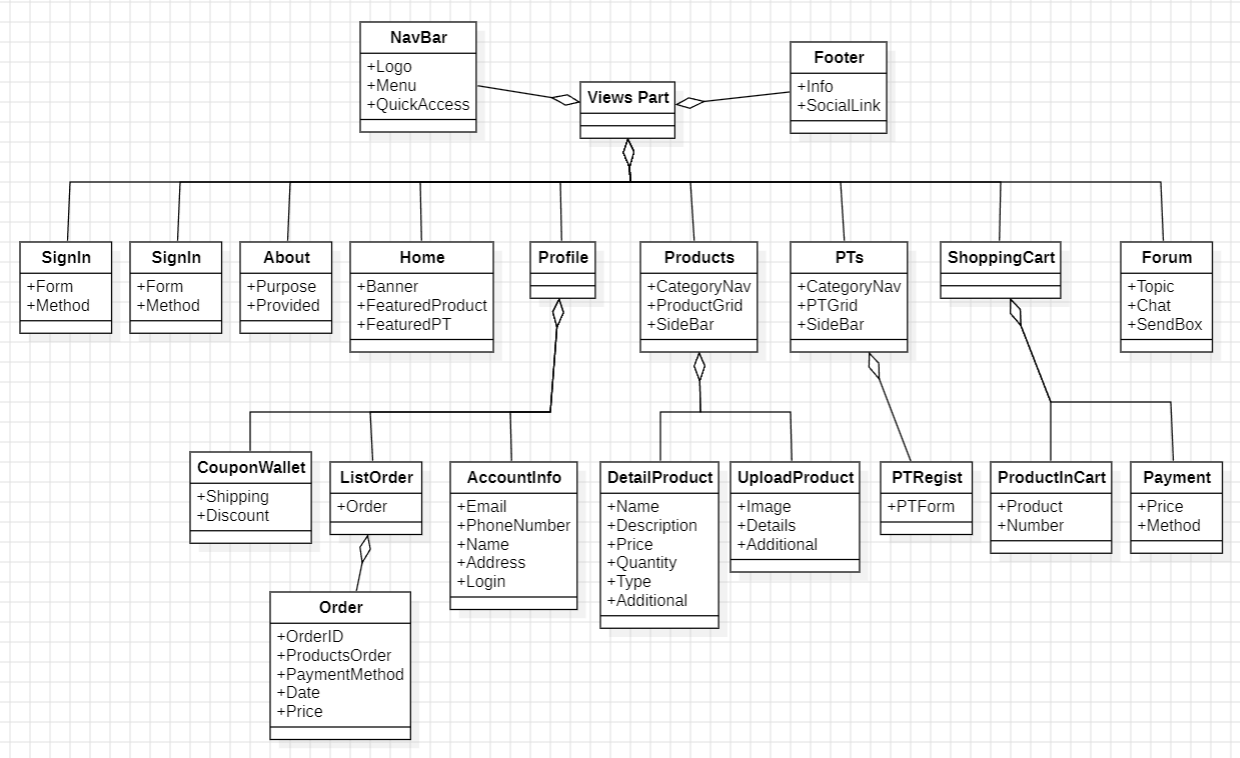
# Use-Case Model



# Logical View



## Views Part



### Component: NavBar

A menu with black text

Description automatically generated

Navigation Bar is always on top of the Views Part.

* Logo: Logo of website
* Menu: A way to access other pages
* Quick Access: Quick Access to search, cart and account

### Component: Footer

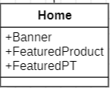
A white and black text

Description automatically generated with medium confidence

Footer is always at the bottom of the Views Part.

* Info: Information of our team such as Address, Phone Number,…
* SocialLink: Link to access our social media such as Facebook, Youtube,…

### Component: Home



Home page is the first page users will see after access website.

* Banner: Show promotion contents, important information,…
* FeaturedProduct: Show our remarked, outstanding products.
* FeaturedPT: Show our remarked, outstanding Personal Trainers.

### Component: About

A white rectangular object with black text

Description automatically generated

* Purpose: Show our purpose when create website.
* Provided: All things we provided throw website.

### Component: Profile

#### Component: ListOrder

A screenshot of a computer

Description automatically generated

This page will show users’orders.

* Order: each order in list.

#### Component: Order

A close-up of a list

Description automatically generated

This section show each order of user.

* OrderID: ID of order.
* ProductsOrder: All products in order.
* PaymentMethod: Payment method of order.
* Date: Created Date of order.
* Price: Total Price of order.

#### Component: AccountInfo

A screen shot of a phone number

Description automatically generated

This page show all account information

* Email: Email of user.
* PhoneNumber: Phone Number of user.
* Name: Name of user.
* Address: Address of user.
* Login: Login method (if use social account such as Facebook, Google).

#### Component: CouponWallet

A screenshot of a coupon

Description automatically generated

This page show user’s coupon.

* Shipping: Coupon to reduce shipping fee.
* Discount: Coupon to discount price.

### Component: Products

A close-up of a product list

Description automatically generated

This page show our products.

* CategoryNav: Navigation bar for our all category of products.
* ProductGrid: Products shown in Grid form.
* SideBar: Side bar for search and filter.

#### Component: UploadProduct

A screenshot of a computer

Description automatically generated

This page is for manager to upload product

* Image: Image of product.
* Details: Details of product (name, description, price, quantity, type).
* Additional: Additional information of product (brand, flavor, weight/size, ingredients, directions).

#### Component: DetailProduct

A close-up of a product

Description automatically generated

Show detail of each product.

* Name: Name of product.
* Description: Description for product.
* Price: Price of product.
* Quantity: Stock quantity of product.
* Type: Type of product.
* Additional: Additional information of product (if product has)

### Component: Shopping Cart

This page show all products that users add to cart and way to pay.

#### Component: ProductInCart

A close-up of a product

Description automatically generated

* Product: Product that customer add to cart.
* Number: Number of product.

#### Component: Payment

A close-up of a price tag

Description automatically generated

This page show payment method.

* Price: Total Price to pay.
* Method: Ways to pay.

### Component: PTs

A close-up of a chart

Description automatically generated

This page show our Personal Trainers.

* CategoryNav: Navigation bar for our all category of Personal Trainers.
* PTGrid: Personal Trainers shown in Grid form.
* SideBar: Side bar for search and filter.

#### Component: PTRegist

A close-up of a graph

Description automatically generated

This page show a form for registering to become our Personal Trainer.

* PTForm: Form for registering.

### Component: Forum

A screenshot of a computer

Description automatically generated

This page show a forum for discussing.

* Topic: Topic to discuss.
* Chat: Chat box to show discussing for each topic.
* SendBox: Box for chatting with everyone on forum.

### Component: SignUp

A sign with black text

Description automatically generated

This page show form to sign up.

* Form: Sign up form.
* Method: Sign up with Social method such as Facebook, Google (if not sign up by form).

### Component: SignIn

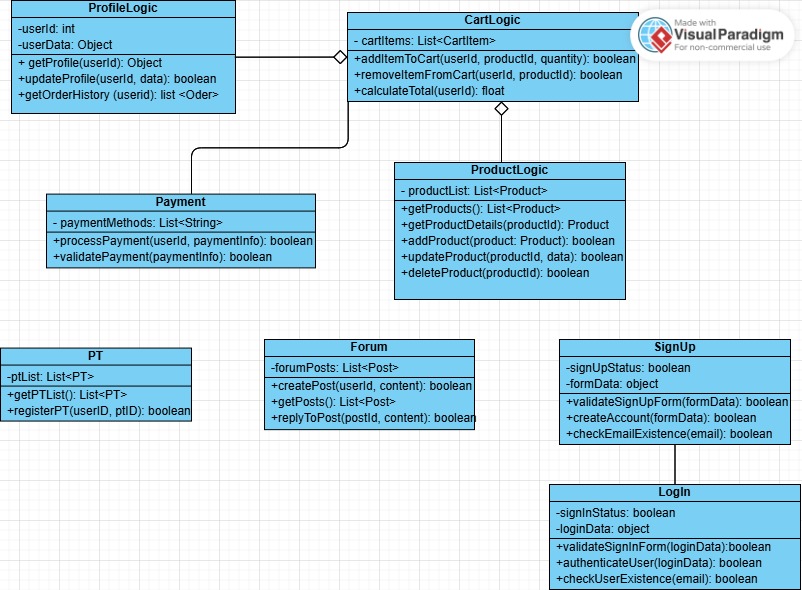
A sign in a graph

Description automatically generated

This page show form to sign in.

* Form: Sign in form.
* Method: Sign in with Social method such as Facebook, Google (if not sign in by form).

## LogicsPart



### Component: ProfileLogic

A blue screen with black text

Description automatically generated

Attributes:

* userId: (int) The unique identifier for the user.
* userData: (Object) An object containing the user's profile information (e.g., name, email, address).

Methods:

* getProfile(userId): Retrieves the profile data of a specific user based on their userId. Returns an object containing the user's profile information.
* updateProfile(userId, data): Updates the user's profile with the provided data. Returns true if the update is successful, otherwise returns false.
* getOrderHistory(userId): Fetches the order history of the user identified by userId. Returns a list of orders.

### Component: ProductLogic

A blue screen with text and black text

Description automatically generated

Attributes:

* productList: (List<Product>) A list containing all the available products on the website.

Methods:

* getProducts(): Retrieves a list of all products from the system.
* getProductDetails(productId): Retrieves detailed information about a specific product identified by productId. Returns a Product object containing details such as name, price, stock, etc.
* addProduct(product): Adds a new product to the system. Returns true if the product is successfully added, otherwise returns false.
* updateProduct(productId, data): Updates the information of a specific product identified by productId using the provided data. Returns true if the update is successful, otherwise returns false.
* deleteProduct(productId): Deletes a specific product from the system identified by productId. Returns true if the product is successfully deleted, otherwise returns false.

### Component: CartLogic

A blue screen with black text

Description automatically generated

Attributes:

* cartItems: (List<CartItem>) A list of items in the user's shopping cart, each containing product details and quantity.

Methods:

* addItemToCart(userId, productId, quantity): Adds a specific product to the user's shopping cart. Returns true if the item is successfully added, otherwise returns false.
* removeItemFromCart(userId, productId): Removes a specific product from the user's shopping cart. Returns true if the item is successfully removed, otherwise returns false.
* calculateTotal(userId): Calculates and returns the total price of the items in the user's shopping cart.

### Component: PaymentLogic

A blue box with black text

Description automatically generated

Attributes:

* paymentMethods: (List<String>) A list of available payment methods (e.g., Credit Card, PayPal, etc.).

Methods:

* processPayment(userId, paymentInfo): Processes the payment for a user based on the provided paymentInfo. Returns true if the payment is successfully processed, otherwise returns false.
* validatePayment(paymentInfo): Validates the provided payment information (e.g., payment details, card number). Returns true if the payment information is valid, otherwise returns false.

### Component: PTLogic

A blue screen with black text

Description automatically generated

Attributes:

* ptList: (List<PT>) A list of personal trainers (PTs) available for registration and viewing on the platform.

Methods:

* getPTList(): Retrieves a list of all personal trainers (PTs) from the system.
* registerPT(userId, ptId): Registers a user as a personal trainer. Returns true if the registration is successful, otherwise returns false.

### Component: ForumLogic

A blue rectangular box with text

Description automatically generated

Attributes:

* forumPosts: (List<Post>) A list of posts in the forum, including content, user, and timestamps.

Methods:

* createPost(userId, content): Allows a user to create a new forum post with the provided content. Returns true if the post is successfully created, otherwise returns false.
* getPosts(): Retrieves a list of all forum posts.
* replyToPost(postId, content): Allows a user to reply to an existing post identified by postId with the provided content. Returns true if the reply is successfully posted, otherwise returns false.

### Component: SignUpLogic

A blue box with black text

Description automatically generated

Attributes:

* signupStatus: The sign-up status of the user.
* formData: The form data submitted during the sign-up (e.g., name, email, password).

Methods:

* validateSignUpForm(formData): Validates the sign-up form data.
* createAccount(formData): Creates a new account for the user.
* checkEmailExistence(email): Checks if the email already exists in the system.

### Component: SignInLogic

A blue box with black text

Description automatically generated

Attributes:

* signInStatus: The sign-in status of the user.
* loginData: The login data (email, password).

Methods:

* validateSignInForm(loginData): Validates the sign-in form data.
* authenticateUser(loginData): Authenticates the user using the login data.
* checkUserExistence(email): Checks if the user exists in the system

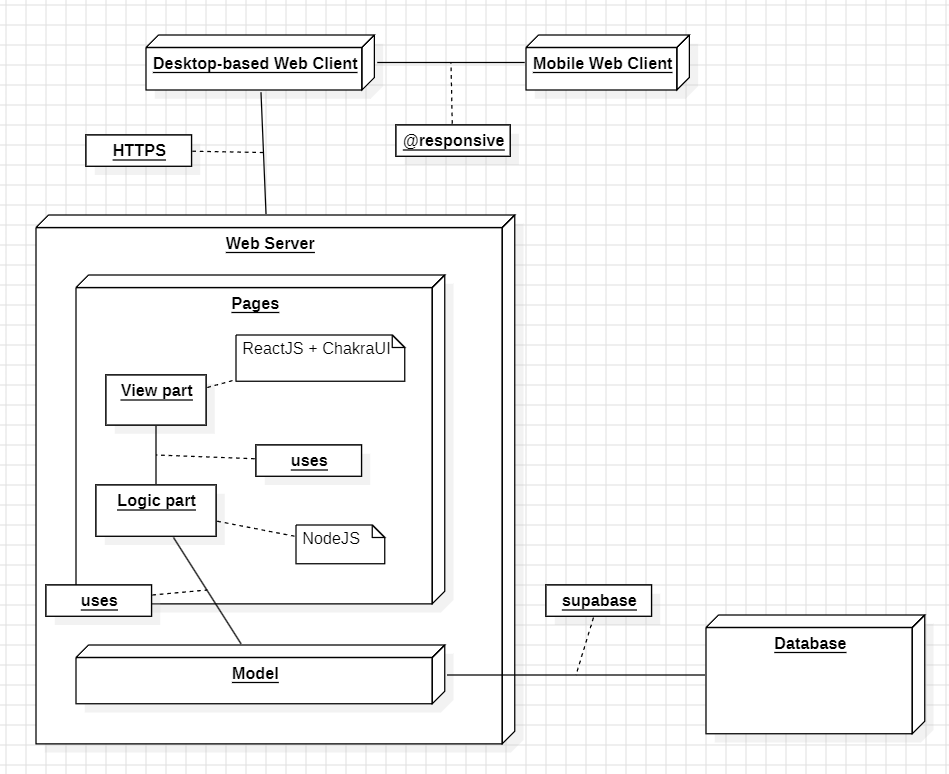
## Database

A computer screen shot of a computer

Description automatically generated

# Deployment

## Deployment Diagram(s)



## Nodes in diagram

* **Database:** Database of software, store all data of software such as Products, User Information, …
* **Web Server:** Contains code of software, it has 2 main parts:
  + Pages: Communicate with Users, it has 2 main components in each page:
    - View part: Front-end of software, display the UI for Users. It use ReactJS and chakraUI Library for implementation.
    - Logic part: Back-end of software, communicate with Users. It use NodeJS for implementation.
  + Model: Communicate with Database, it create query to use on Database and take Database responses.
* **Desktop-based Web Client:** End-Users, connected with Web Server by HTTPS protocol, who can view UI and communicate with software.
* **Mobile Web Client:** From Desktop-based Web Client, uses responsive to make software compatible with mobile devices.

## Connection

* **Web Client and Web Server:** using HTTPS protocol throw Internet connection.
* **In Pages of Web Server:**
  + View part and Logic part: Front-end and Back-end connect by importing files of they are implemented in same files.
  + Logic part and Model: connected by importing files.
* **Model in Web Server and Database:** using supabase.

# Implementation View

src

|\_assets

|\_components

|\_NavBar

|\_Footer

|\_pages

|\_About

|\_DetailProduct

|\_Forum

|\_Home

|\_Payment

|\_Profile

|\_SignIn

|\_SignUp

|\_Products

|\_UploadProduct

However, this folder structure could be changed in the future to match the final system architecture and

design, this structure above here is just a reference to the components illustrated in this document.