Supplement and Fitness Shop

Software Architecture Document

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/11/2024 | <1.0> | <details> | Hoàng Văn Khải |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 5

4. Logical View 6

4.1 Component: Web Application 6

4.2 Component: API Gateway 6

4.3 Component: Services 7

4.4 Component: Database 7

5. Deployment 7

6. Implementation View 7

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

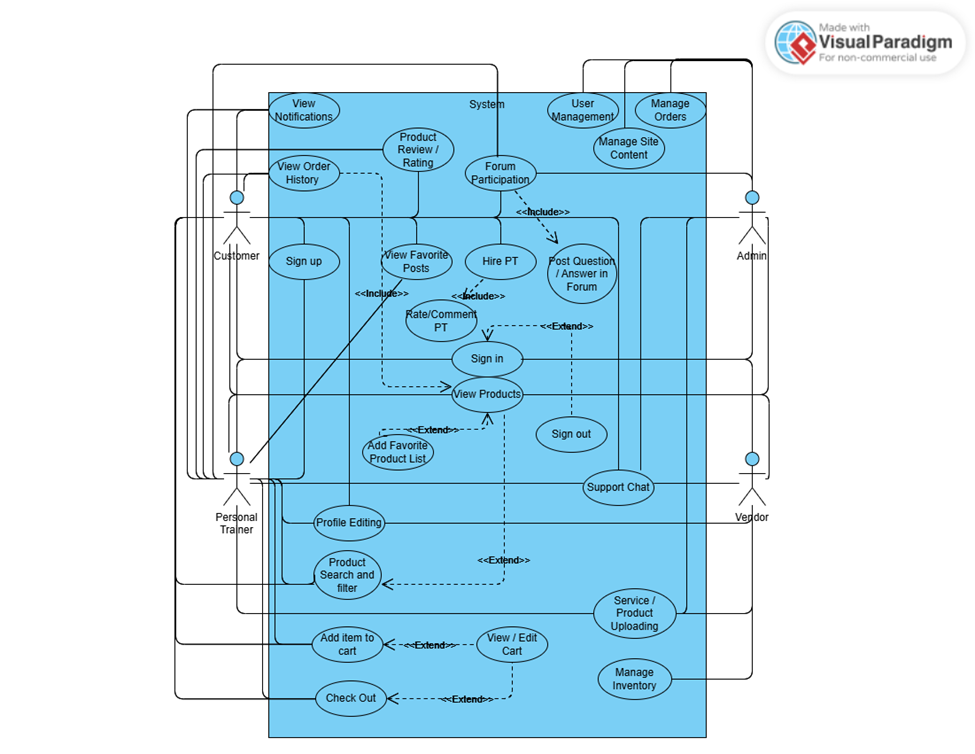
Applicable references are:

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.

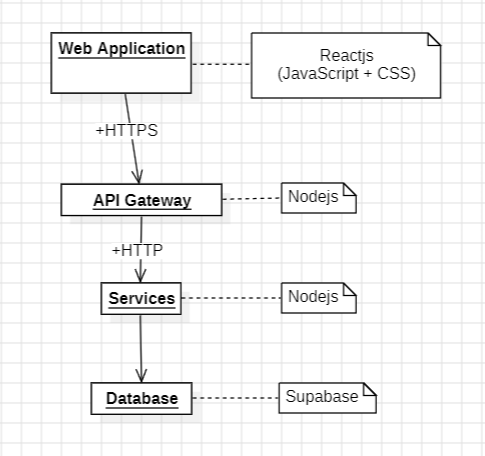
# Architectural Goals and Constraints

* Responding speed: The website should respond to any of the user’s commands within 1 sec
* System requirement: The website can run on any browser, any OS, or any device.
* Availability: The website should be able to be visited 24/7
* Downtime: Should there be any problem and the website has to be shut down for maintenance, downtime should not exceed 12 hours.
* User-friendly: The website’s interface should be friendly to users, and eye-catching. All the features should be easy to understand and quickly get used to. The content must be legal and appropriate for any person.
* Security: The website must have the ability to protect every customer's information.
* Maintenance: The website should be easy to maintain, and repair.
* Database: The website should have a large database to store the website’s data and users’ data.
* Stability: The website should be able to handle 10 users browsing at a time.

# Use-Case Model



# Logical View



* Web Application: The front-end of website. It show view part of website and takes functions of services from API Gateway to complete requirements of stakeholders.
* API Gateway: The way to connect back-end and front-end. It takes requests from front-end, send it to back-end and also takes responses from back-end, send it to front-end.
* Services: The back-end of website. It is designed to complete requirements of stakeholders. It takes requests from API Gateway, takes data from database and send response API Gateway.
* Database: Store data of website.

## Component: Web Application

* **-components**: Array containing reusable UI components (login form, dashboard,…).
* **-styles**: Object holding CSS styles for rendering the UI.
* **+render**(): Handles rendering the UI elements dynamically.
* **+handleRequest**(): Sends requests to the API Gateway.
* **+displayData**(): Displays data received from the API Gateway.

## Component: API Gateway

* **-routes**: Defines the available API routes (/login, /fetch-data).
* **-middlewares**: Middleware functions for request validation, logging,…
* **+routeRequest(req):** Routes incoming requests to the appropriate service.
* **+authenticate(req):** Ensures the request is authorized.
* **+handleResponse(res):** Formats and sends the response to the Web Application.

## Component: Services

* **-logicModules:** Contains specific modules implementing business logic (UserService, ProductService,…).
* **+processRequest(req):** Processes the incoming request and applies business logic.
* **+interactWithDB(query):** Queries the database via defined interfaces.

## Component: Database

A computer screen shot of a computer

Description automatically generated

* **-connection:** Represents the active database connection.
* **-schema:** Defines the structure of tables and relationships.
* **+operation():** To edit database.
* **+executeQuery(query):** Executes SQL or NoSQL queries.
* **+connect():** Establishes a connection to the database.
* **+closeConnection():** Closes the database connection.

# Deployment

# Implementation View