**Healthy Life website manual**

**1. Introduction**

**Healthy Life** is an all-in-one platform designed to promote healthy living through innovative tools and features. The website provides users with an online shop for healthy products, offering a seamless shopping experience with secure payments powered by Stripe. The product catalog is designed to help users easily find and purchase items that support their wellness journey.

In addition to the shop, the platform offers a real-time consultation registration system using WebSockets, enabling users to book appointments with health experts effortlessly. A built-in daily calorie calculator allows users to determine their caloric needs based on their personal details, ensuring they can make informed dietary choices.

The website leverages AI technology to fetch detailed nutritional information about foods and calculate nutrient values, presenting the results in a clear and organized table format. By using Redis for caching frequently accessed objects, such as product details and calorie calculations, the platform ensures high performance and reduced reliance on MySQL database queries.

Users can create personal profiles to manage their activities, view past orders, and save preferences for a tailored experience. The website also features an advanced admin panel for managing products, users, nutritional data, and orders, providing administrators with powerful tools to maintain and optimize the platform. Additionally, an employee portal streamlines the management of order processing, allowing employees to view and update their assigned tasks efficiently.

To enhance security and user experience, the website implements an advanced filtering system in the backend. This includes JWT-based authentication, IP filtering, and rate-limiting mechanisms to protect against unauthorized access and excessive requests. For even faster and more efficient filtering, RabbitMQ is integrated to handle request counting and processing, enabling rapid response times and improved system scalability. Additionally, Redis is employed for caching IP data, ensuring that frequently queried IP addresses are quickly validated without repeated database lookups. This combination of RabbitMQ and Redis optimizes the filtering system, delivering both speed and reliability while maintaining robust security measures.

Combined with a modern and intuitive user interface, **Healthy Life** offers a holistic solution for healthy living, blending e-commerce, expert consultations, and advanced health tools in one cohesive platform.

**2. System architecture**

**FRONT**

The website's frontend architecture is designed to provide an intuitive and visually engaging user experience. It consists of multiple sections, each fulfilling specific functionalities and offering clear navigation paths. The structure emphasizes accessibility and responsiveness, ensuring compatibility across different devices.

**Global Navigation Bar:**

* Positioned at the top of every page.
* Contains links to the primary sections:
  + Home
  + Shop (with dropdown options like "Healthy Products" and "Diet Plans")
  + Calculator
  + About Us
  + Languages
* Includes:
  + A shopping cart icon for quick cart access.
  + A hamburger menu icon to access user account options (e.g., Profile, Admin Panel).

**Pages/Sections:**

* Home Page
  + Hero section
  + Welcoming banner
* About page
  + Info
* Admin Page
  + User control
  + Product control
  + Food item control
  + Order control
* Calculator page
  + Calories calculator
  + Food calorie database entry
  + Meal calorie calculator
* Cart page
  + Cart modification
  + Stripe payment
* Calendar page
  + Registration for service
* Employee page
  + Service order control
* Shop page
  + Filter products
  + Add products to cart

**BACK**

The backend of the website is developed using a robust and scalable technology stack, ensuring optimal performance and reliability. The core backend is built with **Java Spring Boot**, which serves as the framework for managing RESTful APIs and handling business logic. **MySQL** is utilized as the primary database to store structured data, while **Redis** is employed for caching frequently accessed data to improve query performance and reduce database load. Additionally, **RabbitMQ** is integrated for efficient asynchronous communication and message queuing, ensuring smooth handling of tasks and inter-service communication. The backend endpoints are thoroughly documented using **SwaggerEditor**, providing developers with a clear and interactive API specification. Both the Swagger API specification file and the MySQL database schema are uploaded to **GitHub** for easy access and collaboration. The schematic logic of the entire process, from frontend interaction to backend processing, is detailed in diagrams 1–15, showcasing how user requests flow through the system and how data is retrieved, processed, cached, and returned. This architecture ensures a seamless, high-performing, and user-friendly experience.   
 MySQL:

https://github.com/SergejusBal/Final\_health\_shop/blob/master/ExampleOfDataStructure.sql

Swagger:

https://github.com/SergejusBal/Final\_health\_shop/blob/master/HealthyShopAPI.yaml

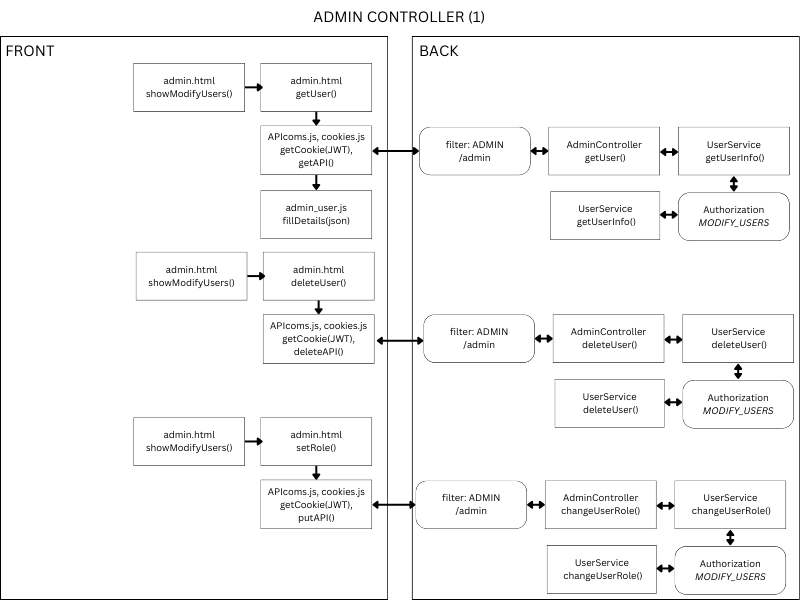
Figure 1. Admin controller schematics.

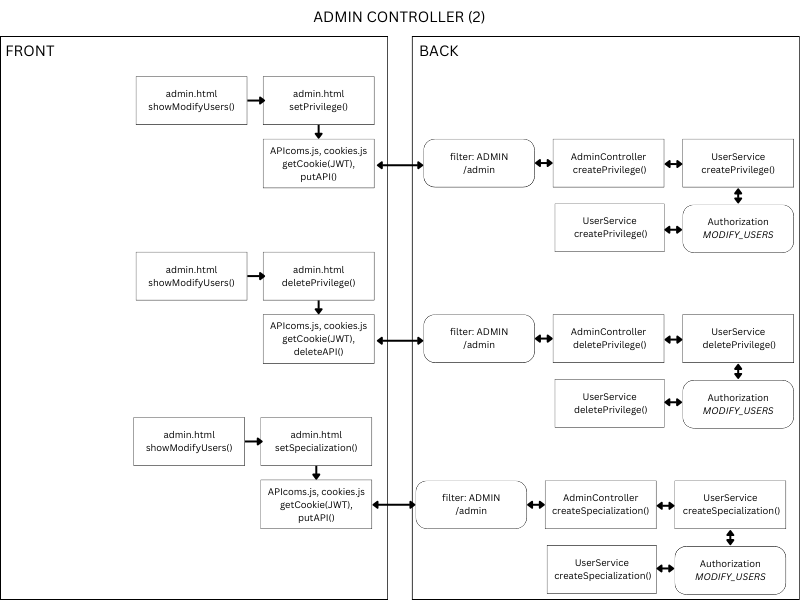
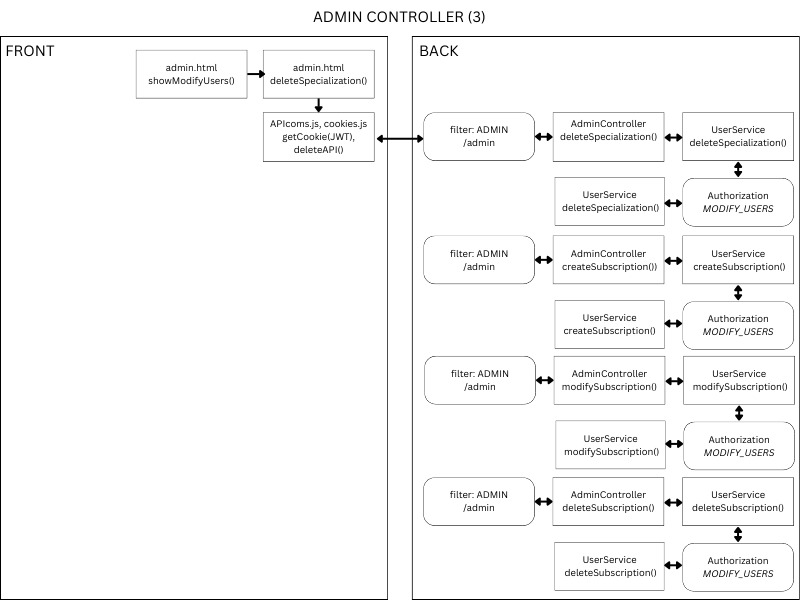
Figure 2 ,3. Admin controllers schematics.

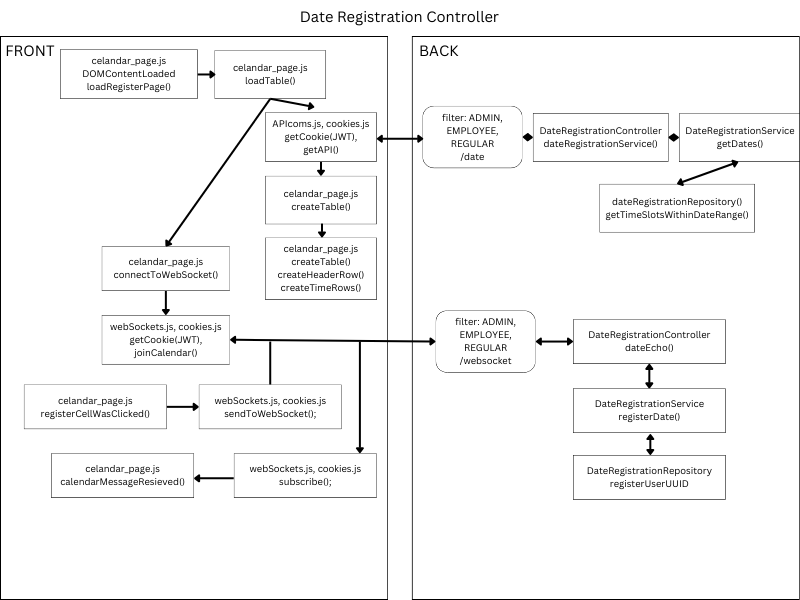
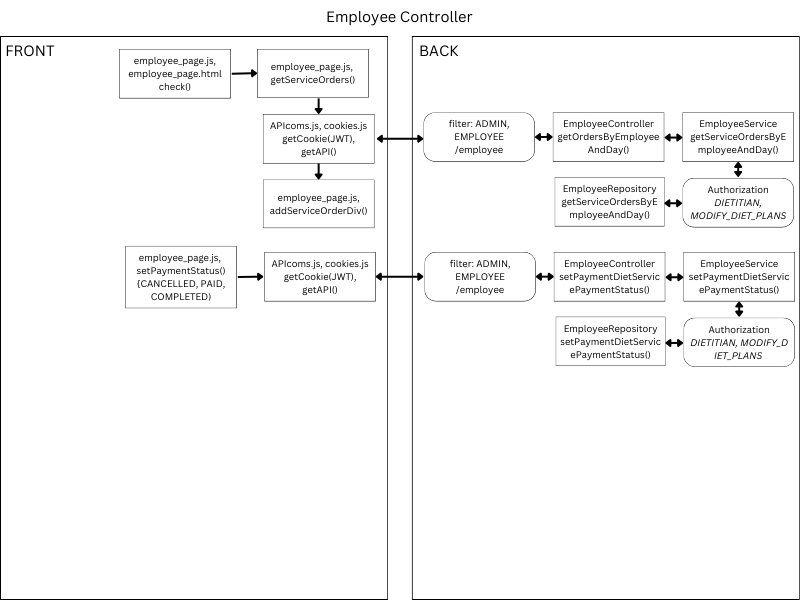
Figure 4, 5. Date registration and employee controllers schematics.

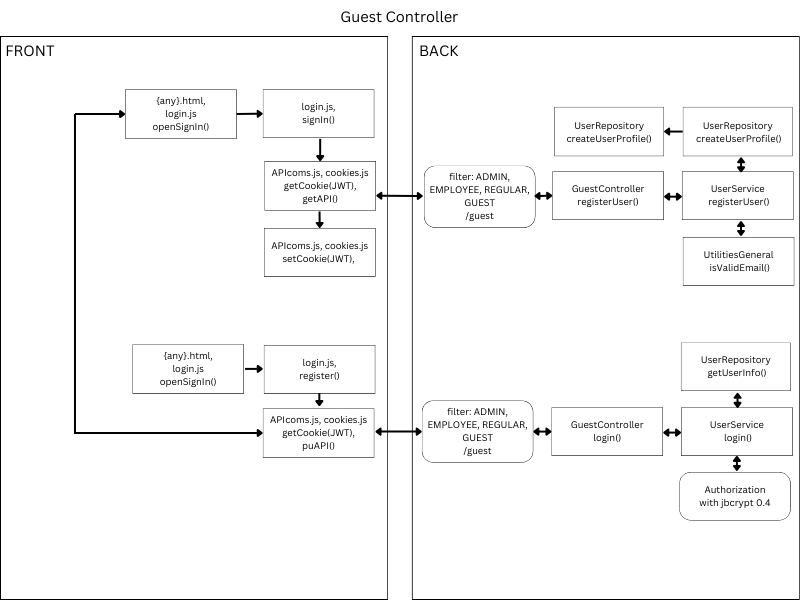
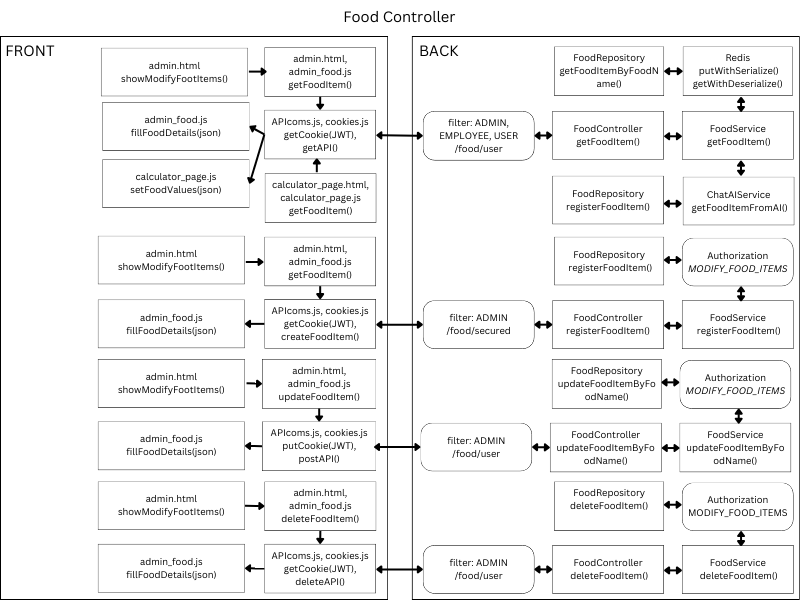
Figure 6, 7. Food item and guest controllers schematics.

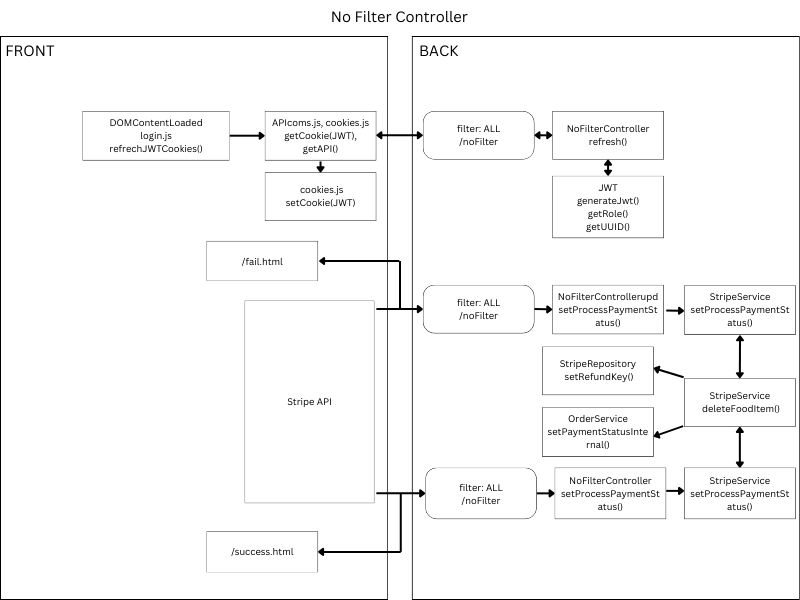
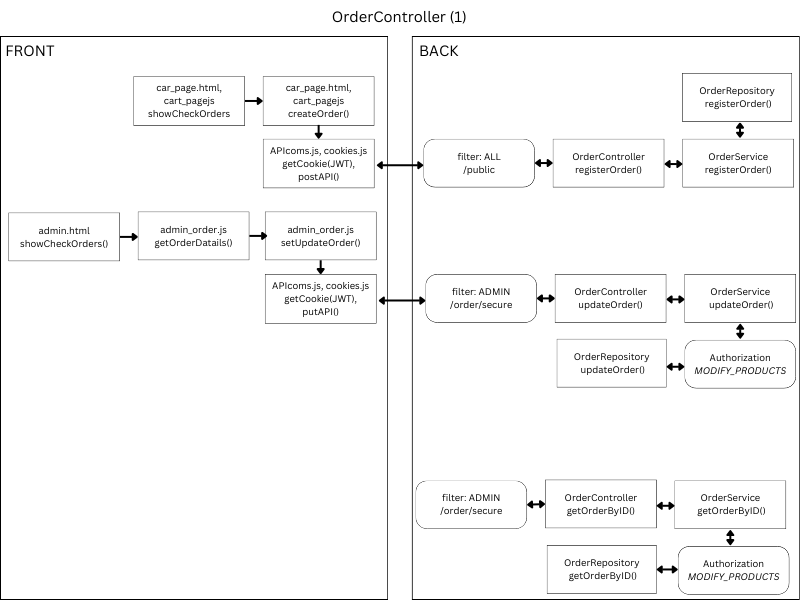
Figure 8, 9. No filter and order controllers schematics.

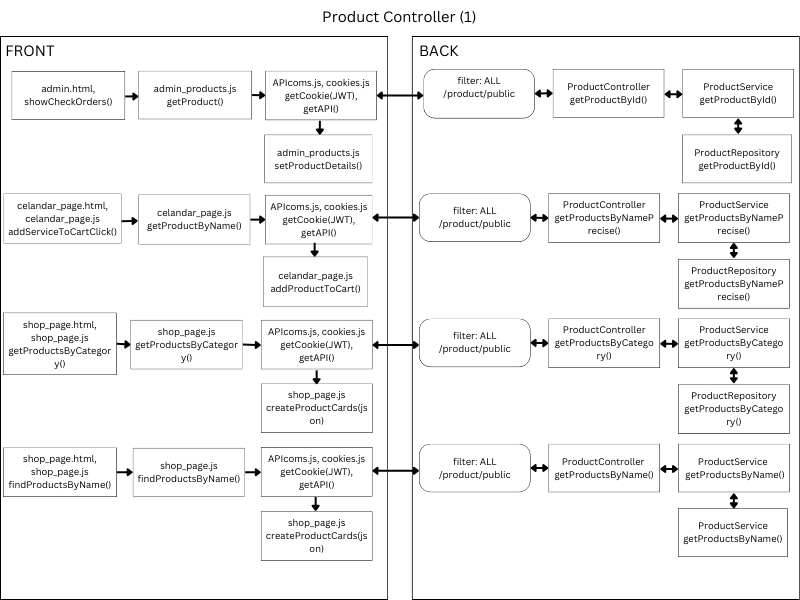
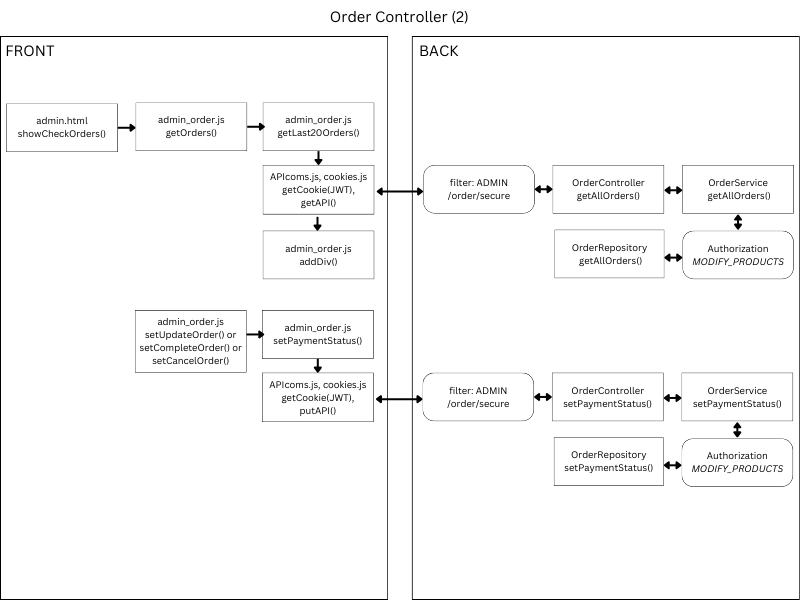
Figure 10,11. Order and product controllers schematics.

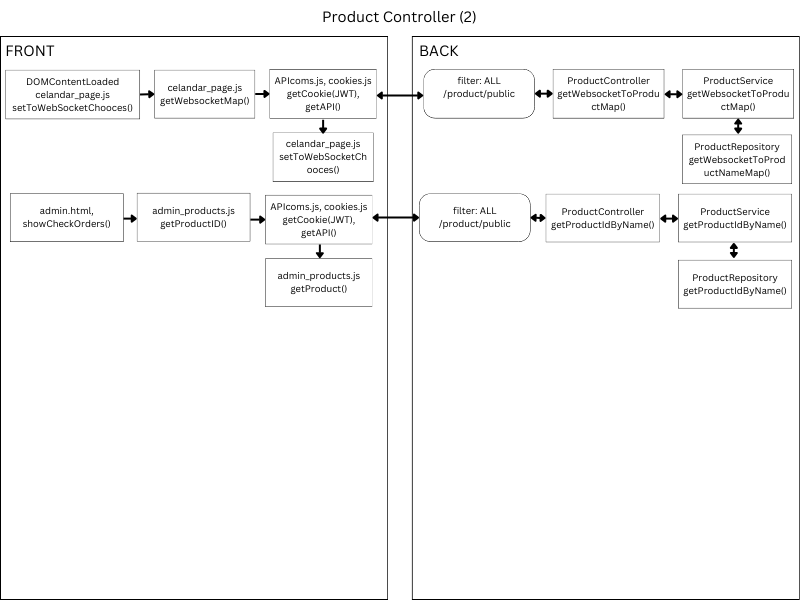
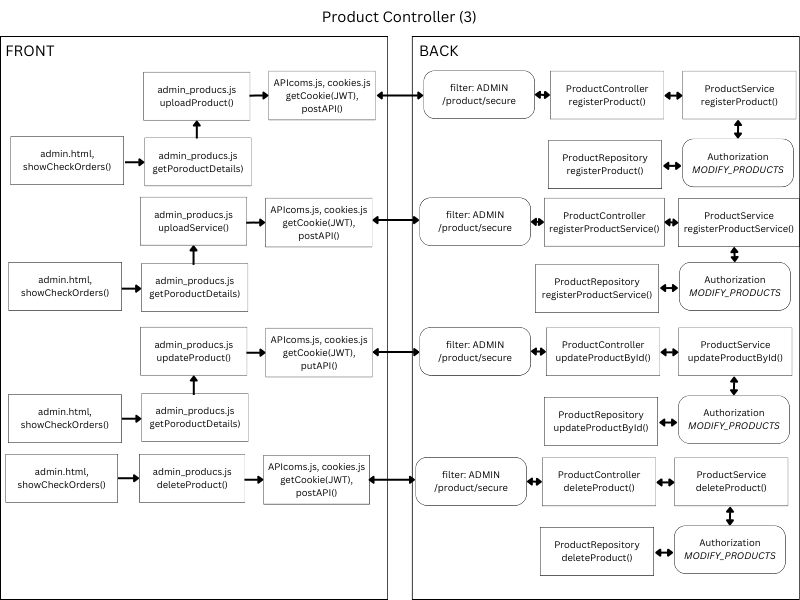
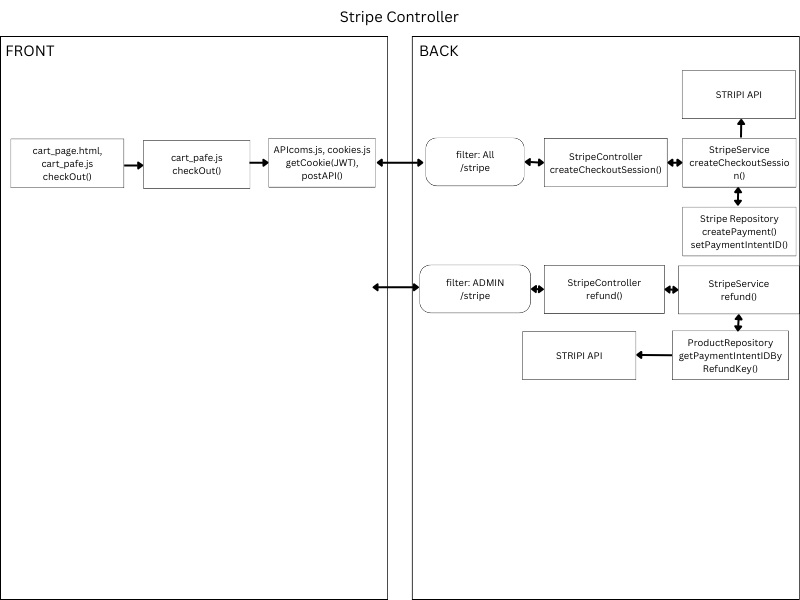
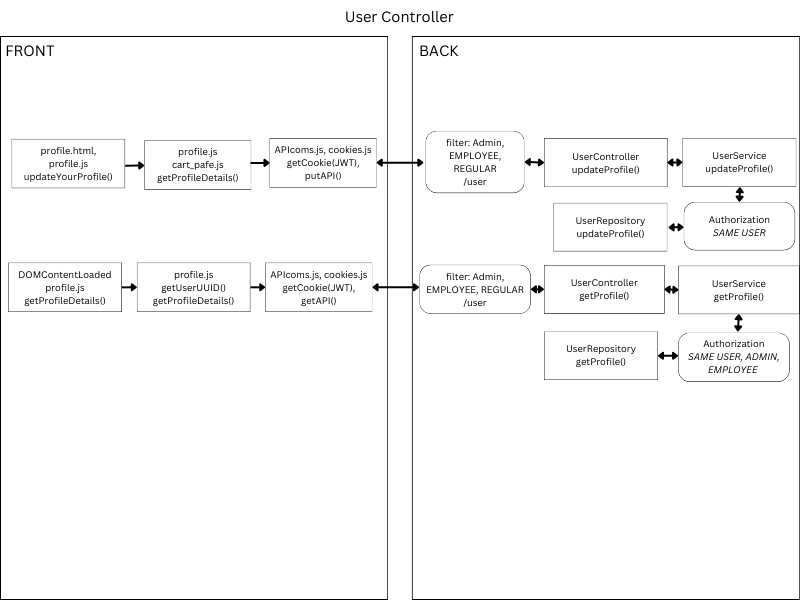
Figure 12,13. Product controllers schematics.

Figure 14, 15. Stripe and user controllers schematics.

It is important to note that the schematic diagrams (1–15) included in the documentation are intended to serve as high-level references for understanding the general flow of operations within the system. These diagrams do not depict all the functions or interactions in detail. A comprehensive functional map, along with the full technical specifications for all features and processes, will be provided in the technical documentation. This ensures that while the diagrams offer an overview, the complete details will be available for deeper technical analysis and implementation guidance.

**3. Installation instructions**

To develop and run the project effectively, several essential tools and technologies are required. These include **MySQL**, a reliable relational database management system, and **MySQL Workbench**, a user-friendly interface for managing databases. For coding and project development, **IntelliJ IDEA**, a powerful integrated development environment (IDE), and **Visual Studio Code (VS Code)**, a versatile code editor, are indispensable. The project also leverages **Docker** for containerization, providing isolated environments for seamless deployment. Additionally, **Redis** is used for caching frequently accessed data to enhance performance, while **RabbitMQ** serves as a message broker, optimizing backend processes like request counting and filtering. These tools together create a robust and efficient development ecosystem.

**MySQL:**

1. Visit the official: <https://dev.mysql.com/downloads/>
2. Select **MySQL Community Server**.
3. Choose the appropriate installer for your operating system:

* **Windows**: Download the MySQL Installer (.msi file).
* **Linux**: Use the repository setup or download the package.
* **macOS**: Download the DMG archive.

1. In the same download section, locate and download **MySQL Workbench**.
2. Run Installer

**IntelliJ IDEA:**

1. Visit the officialhttps://www.jetbrains.com/idea/download/?section=windows
2. Select the **Community Edition** (free version).
3. Choose the installer for your operating system:

* **Windows**: .exe file.
* **macOS**: .dmg file.
* **Linux**: .tar.gz file.

1. Run Installer

**VisualCode**

1. Go to thehttps://code.visualstudio.com/
2. Click the download button for your operating system:

* **Windows**
* **macOS**
* **Linux**

1. Run Installer
2. Go to apps
3. Install live Server

**Docker:**

1. Visit the official https://www.docker.com/
2. Choose the appropriate Docker installer for your operating system:

* **Windows**: Docker Desktop.
* **macOS**: Docker Desktop.
* **Linux**: Docker Engine package or Docker Desktop

1. Run Installer

**Redis**

*PowerShell:*

*docker pull redis:latest*

*docker run -d --name redis-container -p 6379:6379 redis:latest*

*docker ps*

**RabbitMQ**

*PowerShell:*

*docker pull redis:latest*

*docker run -d --name rabbitmq-container -p 5672:5672 -p 15672:15672 rabbitmq:management*

*docker ps*

**4. User guide**

The Healthy Life Website User Guide is designed to help users navigate and utilize the platform effectively. It provides step-by-step instructions for browsing and purchasing healthy products, registering for consultations, and calculating daily caloric needs using the built-in tools. Users can learn how to create and manage their profiles, access past orders, and save preferences. For admins and employees, the guide explains how to manage products, users, and orders through advanced panels. Additionally, it covers features like real-time chat, video consultations, and leveraging AI-powered nutrition tools for meal planning. The guide ensures that users can seamlessly explore and benefit from the platform’s features.

The **Shop Page** allows users to explore products efficiently with intuitive filtering and searching options. Users can filter products by category, selecting from predefined groups such as beverages or snacks, or search for specific products using a search bar by entering a product name or keyword. Once users find a product they like, they can add it to their cart by clicking the **Add to Cart** button. The page supports infinite scrolling, enabling users to load and view more products as they scroll down. Access to these services is restricted to logged-in users, ensuring personalized experiences and secure transactions.

The **Calculator Page** is a powerful tool available exclusively to registered users, designed to help manage calorie intake effectively. By providing personal details such as weight, height, age, gender, and activity level, users can calculate their daily calorie requirements by clicking the **Calculate** button. Additionally, the page offers a **Check Product Calories** feature, enabling users to search for specific products and view their calorie content. If a product is found, users can input the amount in grams and click **Add** to include it in a calorie table. The table dynamically calculates the total calories based on the selected products and quantities, providing a comprehensive overview for better dietary management.  
 The **Register for Consultation** page allows users to book appointments with health experts easily. To access this feature, users need to have an account. Once logged in, they can select a preferred date and time for the consultation in real-time, ensuring up-to-date availability. After choosing the time slot, users can press the **Continue** button, which will seamlessly add the consultation to their shopping cart. From there, they can proceed to checkout alongside other products or services they wish to purchase, making the process efficient and user-friendly.

To register for a consultation, start by clicking on the **hamburger menu** located in the corner of the website. From there, select the **Sign In** option. If you don’t have an account, you’ll be guided to a registration page where you can fill in your details to create one. Once signed in, navigate to the **Register for Consultation** page, select your preferred date and time for the consultation in real-time, and press **Continue**. This will add the consultation to your cart, where you can proceed to checkout along with any other selected products or services.

The **hamburger menu** serves as the central navigation hub for the website, offering quick access to various pages and features. It includes links to the **Profile**, **Employee**, and **Admin** pages:

1. **Profile Page**: Allows users to manage their account details, view past orders, save preferences, and access personalized features.
2. **Employee Page**: Specifically designed for employees, this page enables them to manage consultation service orders. Employees can view assigned orders, update their statuses (e.g., "In Progress" or "Completed"), and streamline the service delivery process.
3. **Admin Page**: A powerful toolset for administrators, offering extensive management capabilities. Admins can:
   * Add, modify, or remove users.
   * Manage the product catalog, including adding or deleting items.
   * Oversee orders, track statuses, and handle complex order processes.
   * Update or add food items to the calorie database, ensuring accurate and up-to-date nutritional information.

This structured menu simplifies navigation and ensures that users, employees, and admins can efficiently access the tools they need.  
 The **Cart Page** provides a clear overview of all selected products and services. Users can review their items and make adjustments to product quantities directly on the page. However, services, such as consultations, remain fixed at a quantity of one to ensure accurate scheduling and processing. Once the cart is finalized, users can proceed by clicking the **Checkout** button. This action redirects them to the **Stripe payment gateway**, where they can securely complete their purchase. The streamlined process ensures an intuitive and hassle-free shopping and booking experience.

**5. User guide**

**Models**

Class BillingOrder

private int id;

private String customerName;

private String customerEmail;

private String customerAddress;

private String orderCart;

private PaymentStatus paymentStatus;

private String promoCode;

private String customerPhone;

private String userUUID;

Class CartItem

private int productId;

private String category;

private String name;

private int quantity;

Enum PaymentStatus

PENDING,

COMPLETED,

PAID,

FAILED,

CANCELLED,

REFUNDED;

Abstract class User

private int ID;

private String UUID;

private String Username;

private String Password;

private Role role;

Class RegularUser extends User

private Boolean subscription;

Class Employee extends User

private List<Specialization> specializations;

Class Admin extends User

private List<Privileges> privileges;

Enum Specialization

DIETITIAN,

TRAINER;

Enum Role

ADMIN,

GUEST,

REGULAR,

EMPLOYEE;

Enum Privileges

MODIFY\_USERS,

MODIFY\_PRODUCTS,

MODIFY\_DIET\_PLANS,

MODIFY\_FOOD\_ITEMS;

CLass Profile

private int ID;

private String userUUID;

private String email;

private String phoneNumber;

private String address;

private String firstName;

private String lastName;

private LocalDate dateOfBirth;

private Double height;

private Double weight;

Class FoodItem

private int id;

private String food;

private Double calories;

private Double proteins;

private Double fats;

private Double carbohydrates;

private Double fibers;

Class Product

private int id;

private String name;

private String description;

private BigDecimal price;

private String category;

private String imageUrl;

Class TimeSlotOrderDTO

private LocalDateTime timeSlot;

private String employeeUuid;

private String userUUID;

private int orderId;

private String customerName;

private String customerPhone;

private String customerEmail;

private int productId;

private String paymentStatus;

Class WebSocketDTO

private LocalDateTime reservedTime;

private String userUUID;

**Java Back-end dependencies:**

1. spring-boot-starter-data-jpa -
2. okhttp - 4.10.0
3. spring-boot-starter-web -
4. spring-boot-starter-validation -
5. spring-boot-starter-test -
6. spring-boot-devtools -
7. h2 -
8. mysql-connector-java - 8.0.31
9. jjwt-api - 0.11.5
10. jjwt-impl - 0.11.5
11. jjwt-jackson - 0.11.5
12. amqp-client - 5.16.0
13. jedis - 4.2.3
14. sendgrid-java - 4.10.1
15. stripe-java - 22.12.0
16. stripe-java - 20.98.0 (duplicate)
17. jbcrypt - 0.4
18. jackson-datatype-jsr310 - 2.15.2
19. spring-boot-starter-websocket -
20. selenium-java - 4.15.0
21. testng - 7.9.0
22. junit -
23. mockito-core -

**JS Front-end dependencies:**

1. <https://js.stripe.com/v3/>
2. <https://cdn.jsdelivr.net/npm/sockjs-client@latest/dist/sockjs.min.js>
3. <https://cdnjs.cloudflare.com/ajax/libs/stomp.js/2.3.3/stomp.min.js>

**Spring Boot properties documentation:**

1. spring.application.name=shop  
   Sets the name of the Spring Boot application.
2. jar.java.url=http://localhost:8080  
   URL for the backend Java service.
3. website.url=http://127.0.0.1:5501  
   URL for the frontend website.
4. cors.allowed.origins  
   Defines allowed origins for cross-origin resource sharing (CORS).
5. spring.datasource.url  
   JDBC URL for connecting to the MySQL database.
6. spring.datasource.username  
   Username for the MySQL database.
7. spring.datasource.password  
   Password for the MySQL database.
8. redis.host=localhost  
   Hostname for the Redis server.
9. redis.port=6379  
   Port number for the Redis server.
10. rabbit.host=localhost  
    Hostname for the RabbitMQ server.
11. jwt.key  
    Secret key for signing JSON Web Tokens (JWT).
12. jwt.token.maxsize=2048  
    Maximum size of the JWT token in bytes.
13. jwt.token.expiration=300  
    Expiration time for JWT tokens in seconds.
14. ip.request.maxsize.ban=200  
    Maximum allowed IP requests before banning.
15. ai.key  
    API key for AI-related services.
16. stripe.api.key  
    API key for Stripe payment integration.

**6. Maintenance and Support**

Regular maintenance and support are essential to ensure the smooth operation of the project. The food calorie calculator relies on AI subscription service to function, and this cost must be accounted for as part of the ongoing operational expenses. Additionally, the backend and frontend must run concurrently and remain synchronized for the system to operate effectively. Keeping dependencies updated and incorporating user feedback will further enhance the reliability, security, and overall user experience of the system.

**Potential Updates**

The platform has significant potential for future updates and feature expansions. One possible enhancement is the addition of a diet plan page, accessible through the user's hamburger menu, allowing for a more streamlined user experience. The platform could also be expanded to include workout plans and workout plan calendars, providing users with a comprehensive health and fitness solution. Furthermore, a user information page could be introduced to collect details about medical conditions, enabling the generation of more precise and personalized diet plans. This sensitive data would be encrypted and securely managed in the Java backend and stored in a MySQL database, ensuring both privacy and compliance with data security standards.

**Potential Issues and Solutions**

Currently, no significant potential issues have been identified in the project configuration or dependencies.