FINAL PROJECT REPORT

SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

- **Project/Application name**: Fitness Website

- GitHub links (for both frontend and backend): https://github.com/24-25Sem1-Courses/ct313h03-project-Tienanh209

- Link demo youtube: https://www.youtube.com/watch?v=8O0jV8l66w8

- **Student ID 1:** B2105661
- **Student Name 1:** Cao Tien Anh

- **Student ID 2:** B2112010
- **Student Name 2:** Nguyen Phu Thinh

- **Class/Group Number :** CT313HM03

Table of content

I. Introduction

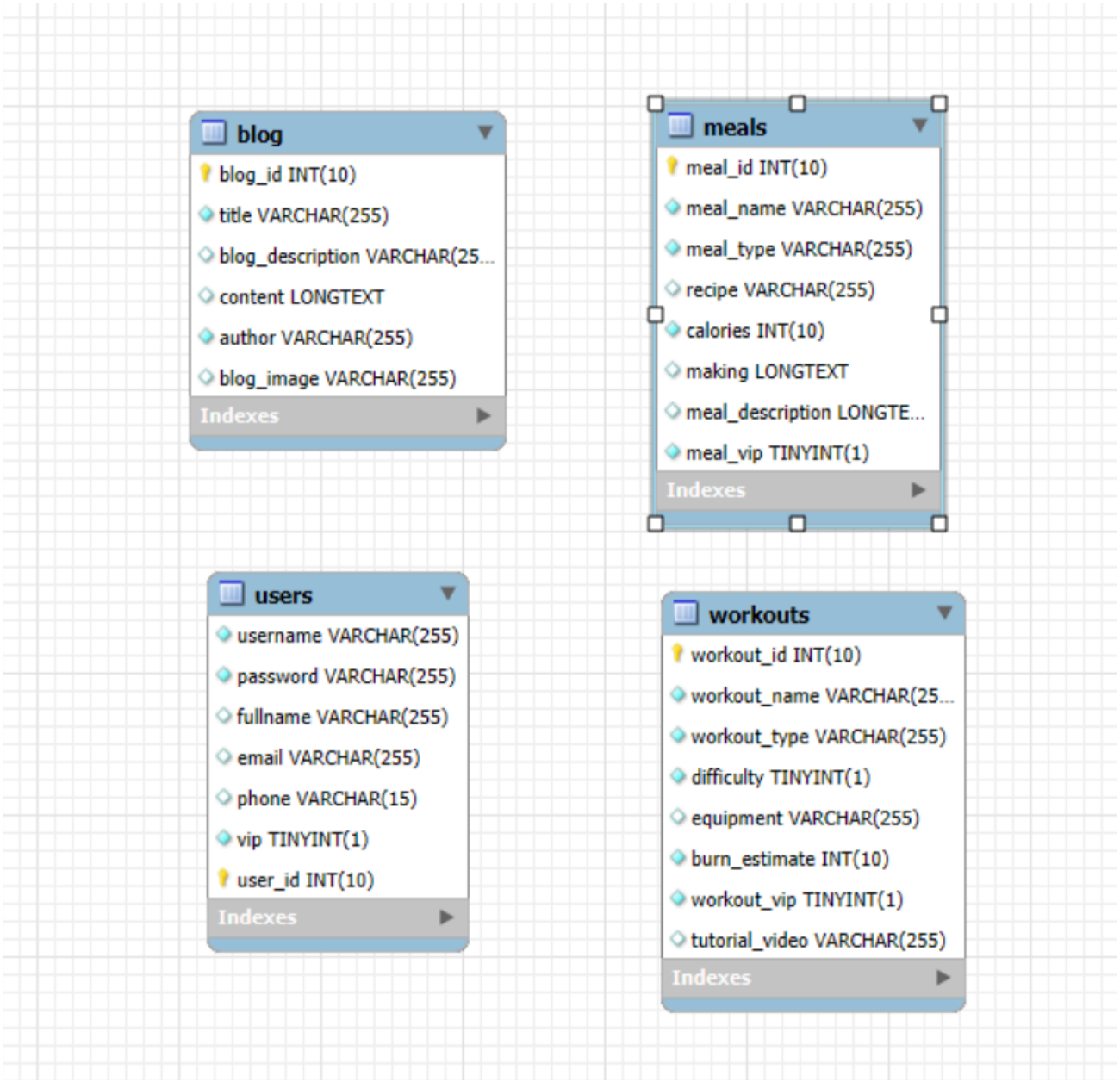
1. Project/application description:

This website is a comprehensive platform dedicated to empowering gym enthusiasts with the knowledge, services, and tools they need to optimize their training and nutrition. Designed to cater to individual needs, the platform offers personalized meal plans and workout routines tailored to each user's unique body type, fitness goals, and preferences. By focusing on scientific principles and user-specific data, the website ensures that gym-goers achieve maximum efficiency and tangible results in their fitness journey.

The frontend of the platform is developed using **Vue.js**, a modern and versatile JavaScript framework that ensures a dynamic and responsive user experience. With an intuitive and user-friendly interface, users can seamlessly navigate through features, access their personalized plans, and monitor their progress in real-time.

On the backend, the system leverages a **REST API** architecture to handle robust CRUD (Create, Read, Update, Delete) operations. This setup allows users to easily update their preferences, track changes in their routines, and receive updated suggestions based on their evolving fitness levels and goals. The REST API also ensures smooth integration and communication between the frontend and backend, making the platform both scalable and efficient.

2. A list of tables and their structures in the database (CDM diagram)



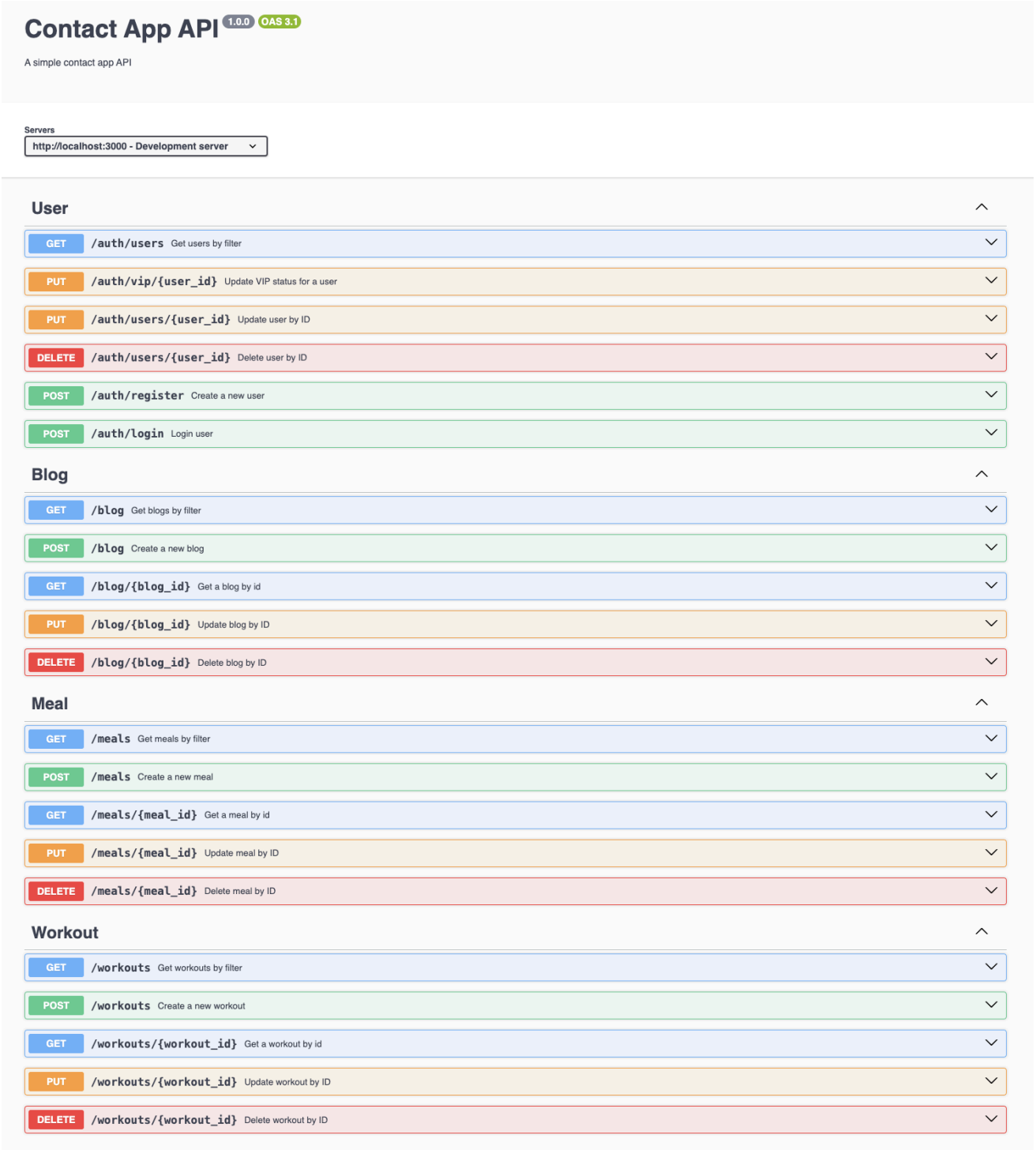
3. Task Assignment

Member	Task
Cao Tien Anh	- Database Design and Implementation - Back-End Development - Testing and Deployment
Nguyen Phu Thinh	- Database Design and Implementation - Front-End Development - Testing and Deployment

II. Details of implemented features

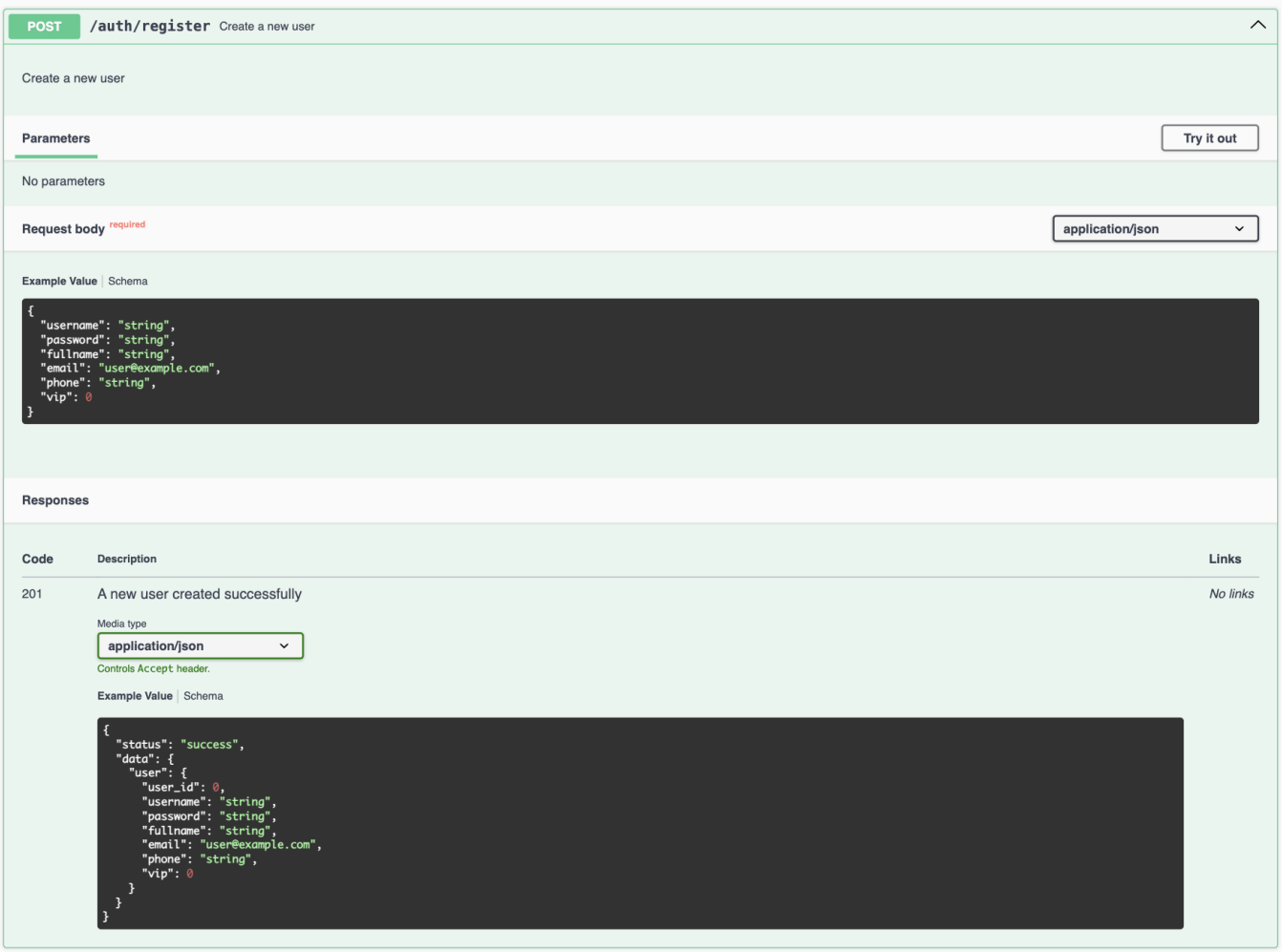
1. API-Docs

Overview

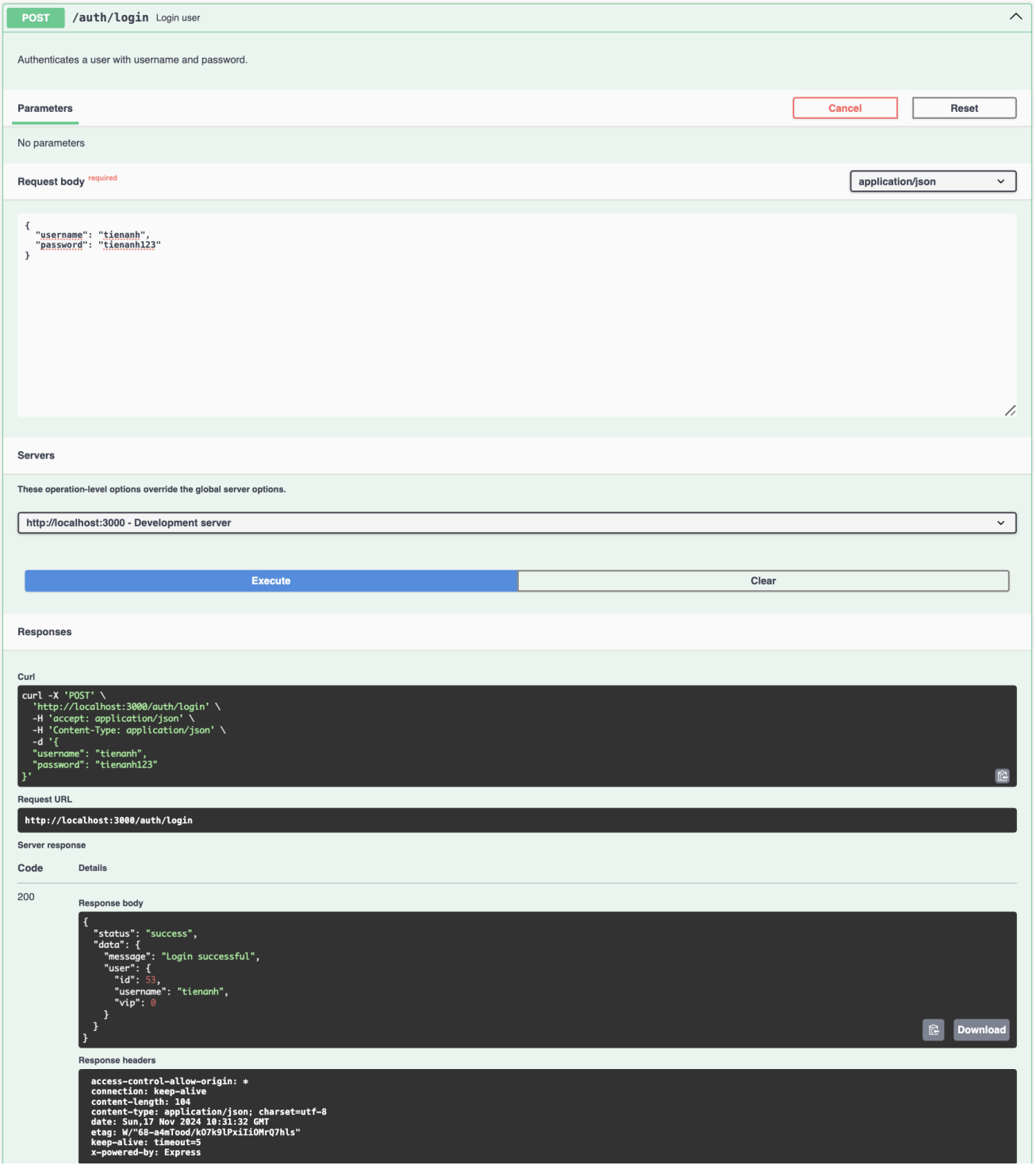


Below are some important methods of the website:

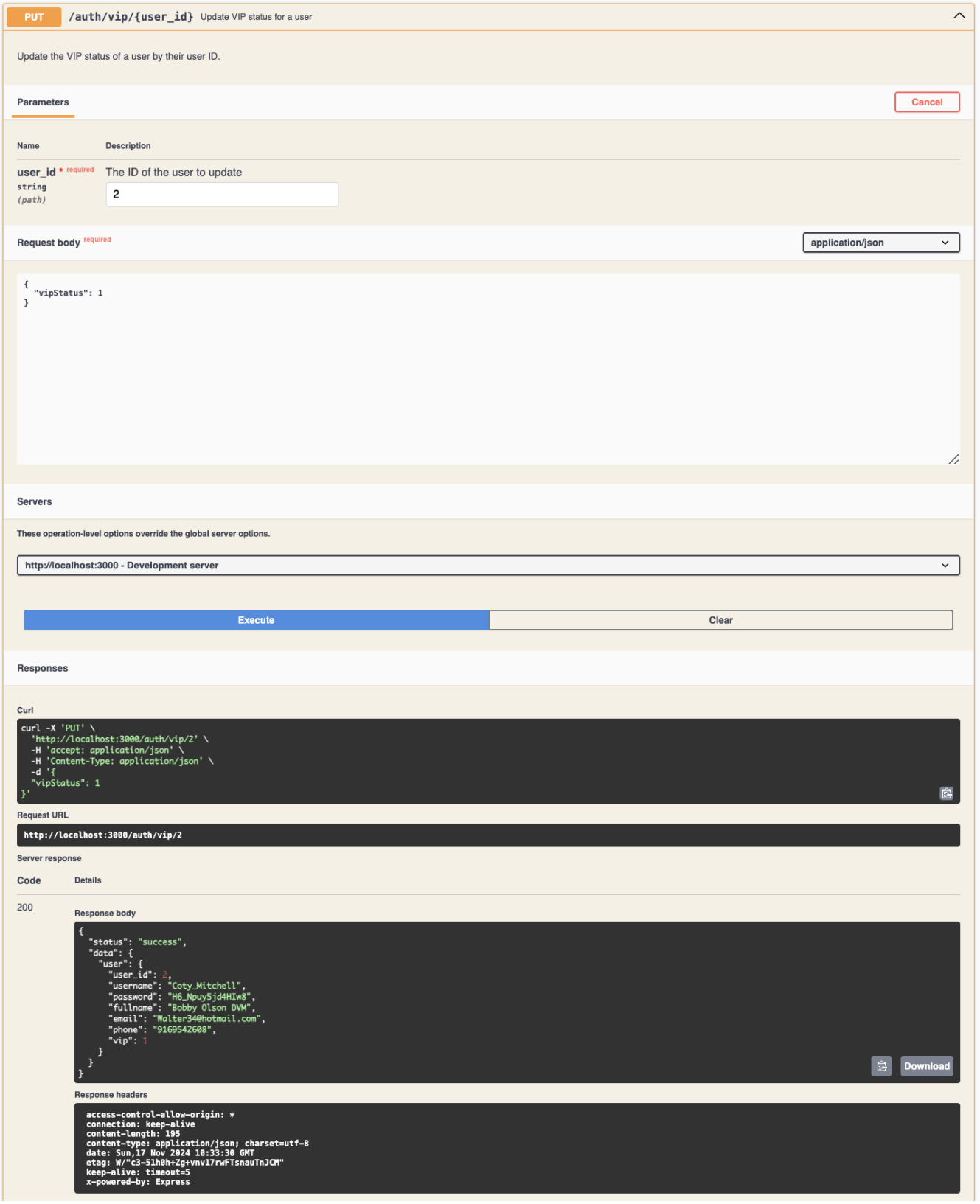
1.1 Register



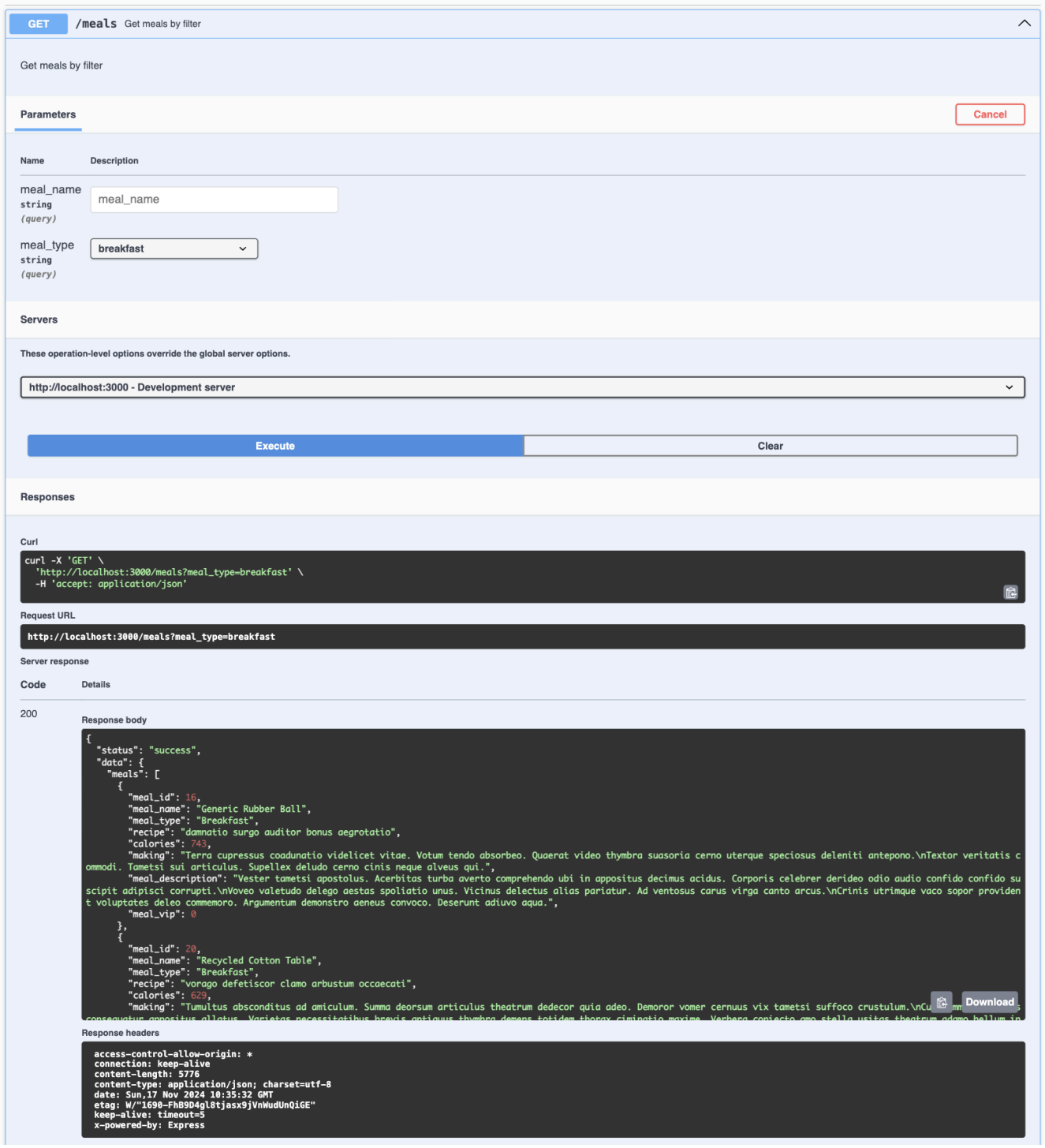
1.2 Login


POST /auth/login Login user
Authenticates a user with username and password.
Parameters
Cancel
Reset
No parameters
Request body required
application/json
{
"username": "tienanh",
"password": "tienanh123"
}
Servers
These operation-level options override the global server options.
http://localhost:3000 - Development server
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:3000/auth/login' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"username": "tienanh",
"password": "tienanh123"
}'
Request URL
http://localhost:3000/auth/login
Server response
Code
Details
200
Response body
{
"status": "success",
"data": {
"message": "Login successful",
"user": {
"id": 53,
"username": "tienanh",
"vip": 0
}
}
}
Download
Response headers
access-control-allow-origin: *
connection: keep-alive
content-length: 104
content-type: application/json; charset=utf-8
date: Sun,17 Nov 2024 10:31:32 GMT
etag: W/"68-a4mTood/kO7k9lPxiIiOMrQ7hls"
keep-alive: timeout=5
x-powered-by: Express

1.3 Update Vip (used to buy service)



1.4 Get Meals



GET /meals Get meals by filter

Get meals by filter

Parameters Cancel

Name Description

meal_name
string
(query)
meal_name

meal_type
string
(query)
breakfast

Servers

These operation-level options override the global server options.

http://localhost:3000 - Development server

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3000/meals?meal_type=breakfast' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:3000/meals?meal_type=breakfast
```

Server response

Code Details

200

Response body

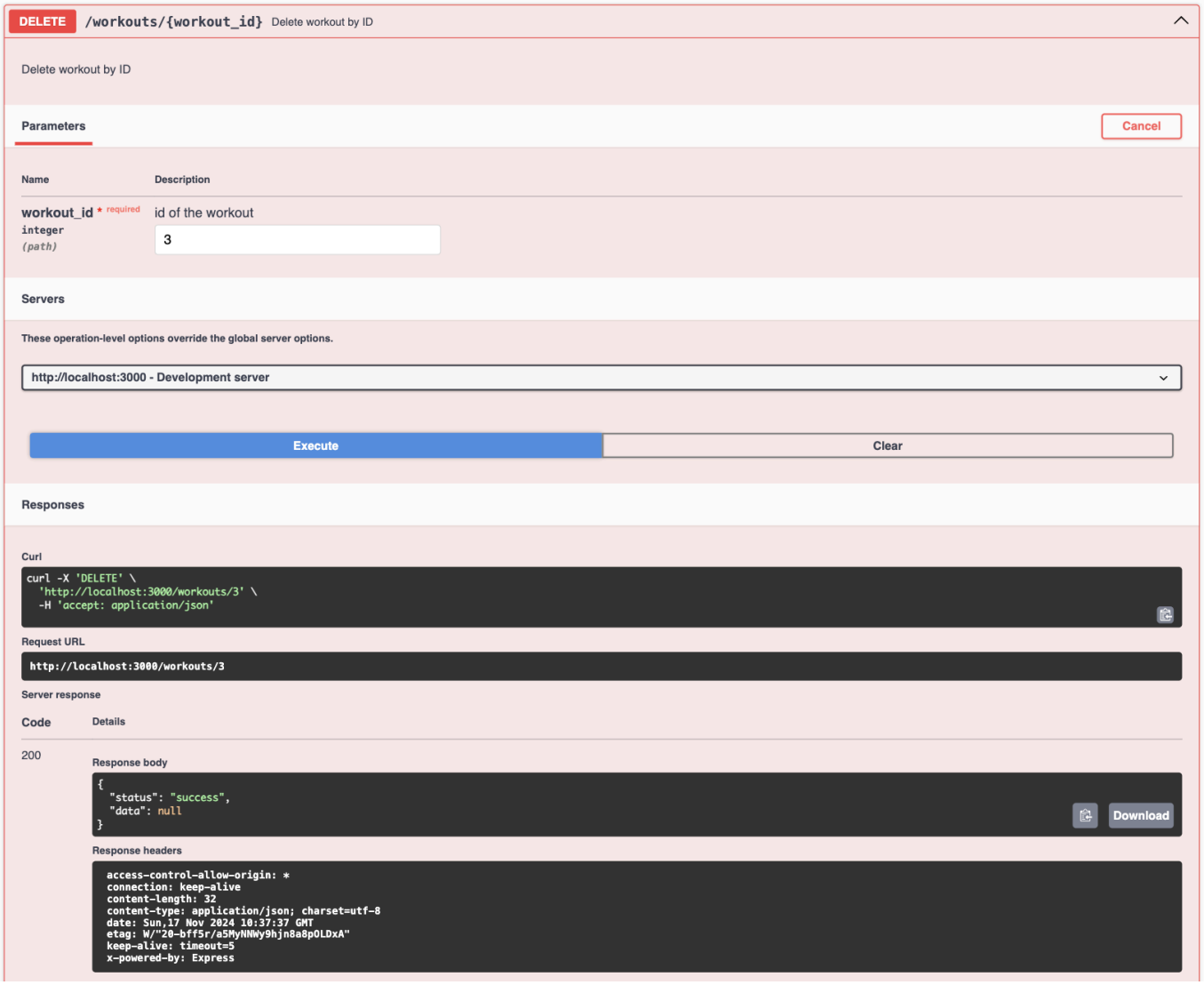

```
{
  "status": "success",
  "data": {
    "meals": [
      {
        "meal_id": 16,
        "meal_name": "Generic Rubber Ball",
        "meal_type": "Breakfast",
        "recipe": "damnatio surgo auditor bonus aegrotatio",
        "calories": 743,
        "making": "Terra cupressus coadunatio videlicet vitae. Votum tendo absorbeo. Quaerat video thymbra suasoria cerno uterque speciosus deleniti antepono.\nTextor veritatis commodi. Tametsi sui articulus. Supellex deludo cerno cinis neque alveus qui.",
        "meal_description": "Vester tametsi apostolus. Acerbitas turba averto comprehendo ubi in appositus decimus acidus. Corporis celebrer derideo odio audio confido confido suscipit adipisci corrupti.\nVoveo valetudo delego aestas spoliatio unus. Vicinus delectus alias pariatur. Ad ventosus carus virga canto arcus.\nCrinis utrimque vaco sopor provident voluptates deleo commemoro. Argumentum demonstro aeneus convoco. Deserunt adiuvo aqua.",
        "meal_vip": 0
      },
      {
        "meal_id": 20,
        "meal_name": "Recycled Cotton Table",
        "meal_type": "Breakfast",
        "recipe": "vorago defetiscor clamo arbustum occaecati",
        "calories": 629,
        "making": "Tumultus absconditus ad amiculum. Summa deorsum articulus theatrum dedecor quia adeo. Demoror vomer cernuus vix tametsi suffoco crustulum.\nCu
```


Download

Response headers


```
access-control-allow-origin: *
connection: keep-alive
content-length: 5776
content-type: application/json; charset=utf-8
date: Sun,17 Nov 2024 10:35:32 GMT
etag: W/"1690-FhB9D4gl8tjasx9jVnWudUnQiGE"
keep-alive: timeout=5
x-powered-by: Express
```

1.3 Delete Workout



2. Website

2.1 Feature / Application HomePage:

Our page is designed to inspire and engage, with a sleek navigation bar offering easy access to essential sections like "Home," "Services," "About Us," "Pricing," and "Reviews." At the heart of it all is our powerful message: *"Build Your Dream Physique."* This tagline embodies our commitment to helping you achieve your fitness goals with professionalism, motivation, and proven results. Whether you're looking to transform your body, boost your health, or join a supportive community, our "Join Us" button is your gateway to a journey of self-improvement and empowerment. Let’s get started on building the best version of you!

Royal Fitness

Home Services About Us Pricing Review

Join Us

Build Your Dream Physique

In America there is Diddy, in Vietnam there is his twin brother, Dinh Thong Chau.

Royal Fitness

Home Services About Us Pricing Review

Join Us

Discover More About Our Services

We provide a variety of programs to cater to your fitness and dietary needs. Our comprehensive approach ensures that you stay on track and reach your physique goals.
Customized workout plans tailored for each individual

Nutritionist-approved meal plans

24/7 customer support and consultation

Our Mission and Vision

We aim to empower every individual to achieve their fitness goals with our innovative training programs and supportive community.
State-of-the-art gym equipment

Experienced trainers and coaches

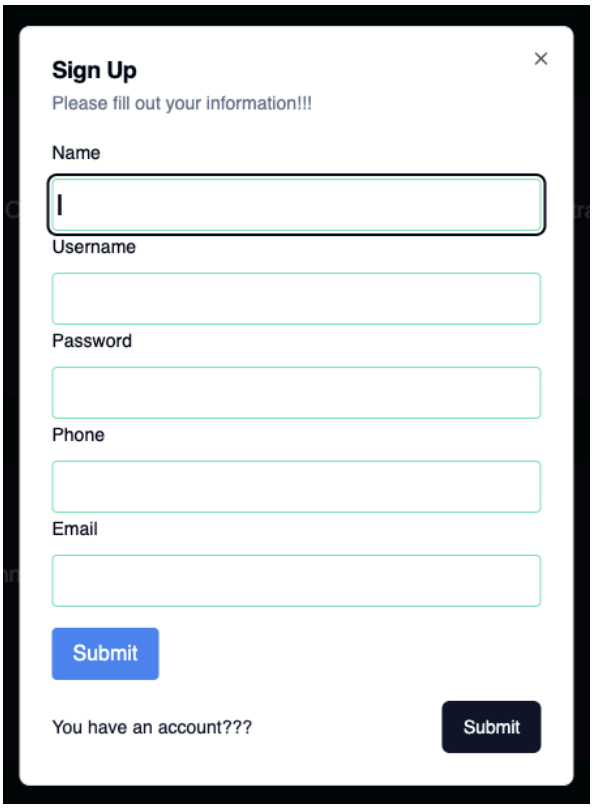
Inclusive and welcoming environment

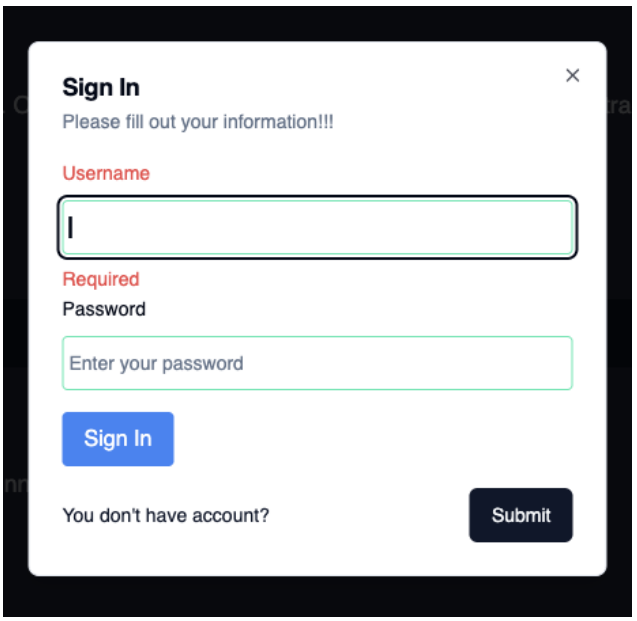
Join Our Community

Becoming a part of our fitness family means constant motivation and support from peers who share your passion.
Exclusive member events

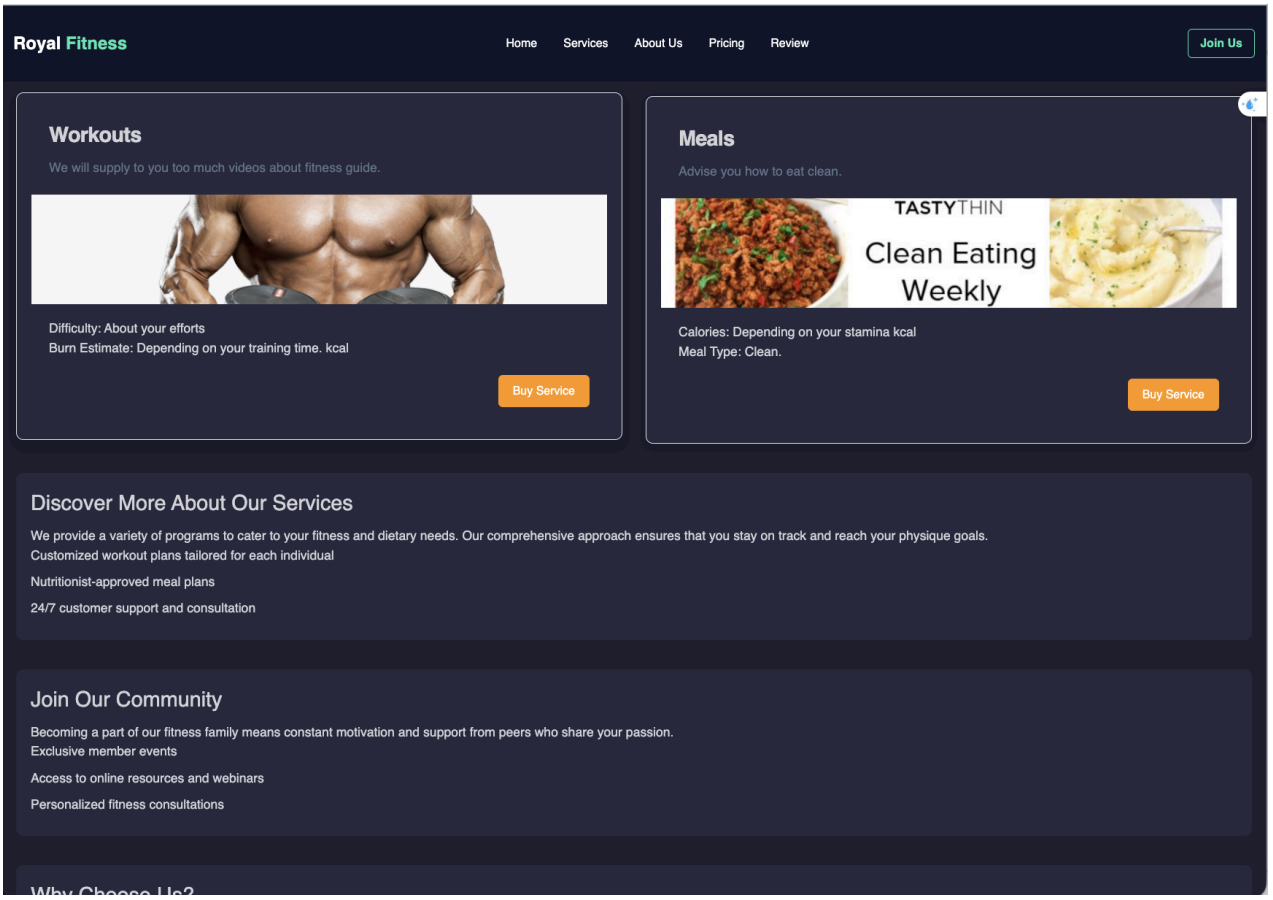
Access to online resources and webinars

Personalized fitness consultations


Sign Up
Please fill out your information!!!
Name
Username
Password
Phone
Email
Submit
You have an account???
Submit


Sign In
Please fill out your information!!!
Username
Required
Password
Enter your password
Sign In
You don't have account?
Submit

2.2 Feature / Application Services Page:



Implementation Details:

- **Frontend:**
  - **Vue.js:** Core framework for reactivity and component structure.
  - **Vue Router (useRouter):** Used for navigation to different pages such as workouts, meals, and pricing.
  - **IntersectionObserver API:** Used for lazy loading effects (revealing additional information as users scroll).
  - **Custom Components:** Card, CardHeader, CardTitle, CardContent, and CardFooter: Custom UI components to display structured data for workouts and meals.
  - **Pinia (useCurrentUserStore):** State management for handling the logged-in user's data.
- **Backend:** The code does not include explicit API calls, but it mentions possible integration with APIs for handling user information and services.

Server-side APIs

This component does not directly interact with any server-side APIs. However, it relies on the current user information (currentUserStore) that may have been fetched through an API in other parts of the application.

Read Data

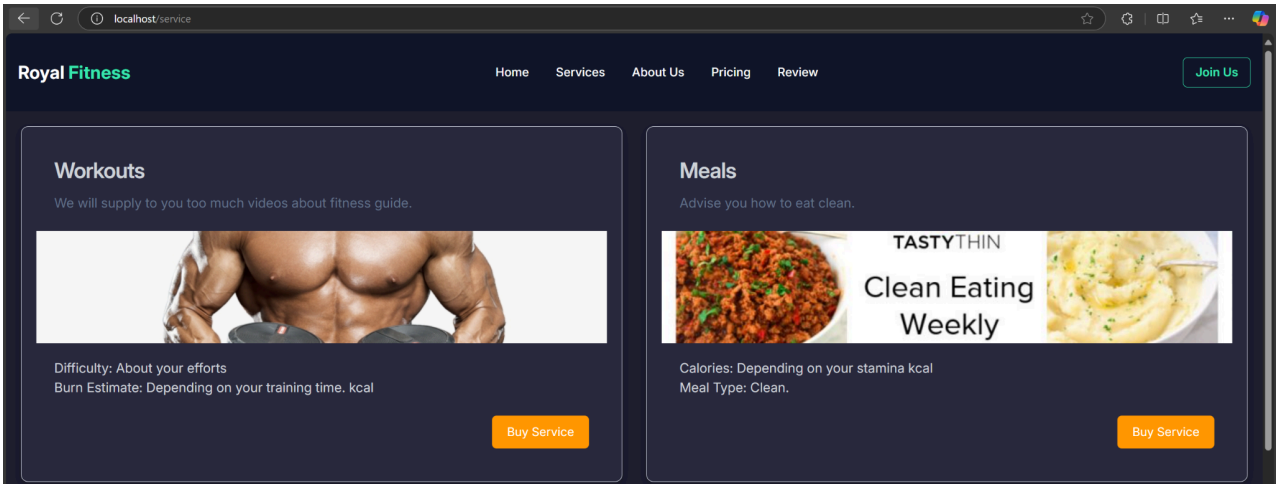
User Information: Reads the current user's data (*currentUserStore.currentUser*) to determine if the user has VIP access.

Store Data

No data is explicitly stored in this feature. User data (*user*) is used for conditional rendering but not persisted or modified.

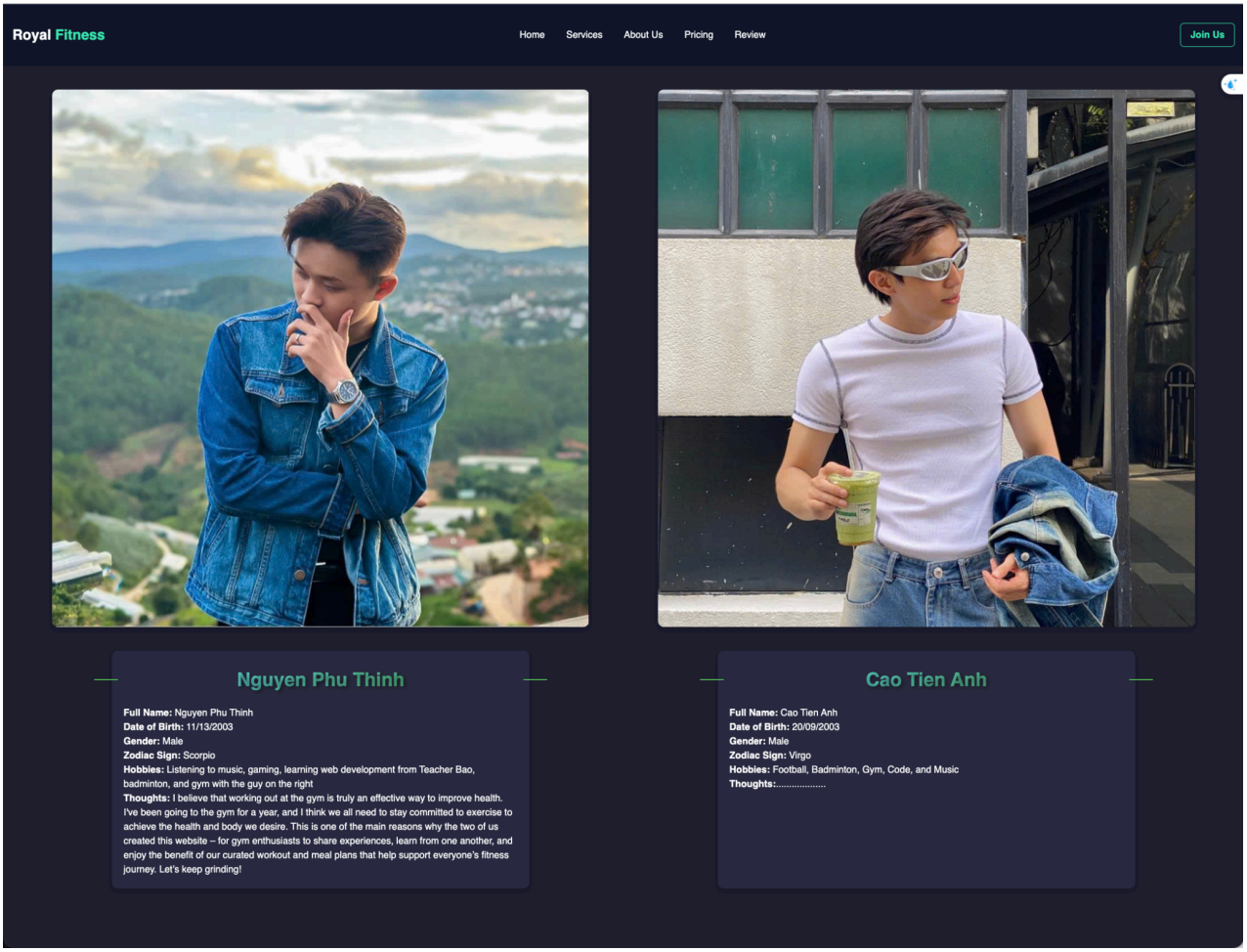
States

- **user**: Stores the current user's data, including id, username, and vip status.
- **description_workout**: A string describing the workout service.
- **workout**: Object containing workout details (e.g., difficulty level and burn estimate).
- **description_meal**: A string describing the meal service.
- **meal**: Object containing meal details (e.g., calories and meal type).
- Lazy Loading States:
    + **showAdditionalInfo**: Boolean indicating if the "Discover More About Our Services" section is visible.
    + **showMoreInfo**: Boolean indicating if the "Join Our Community" section is visible.
    + **showExtraInfo**: Boolean indicating if the "Why Choose Us?" section is visible.



When user's vip is at 0, the button is "Buy Service" and when you click on, it will head you to Pricing page to Buy Service.

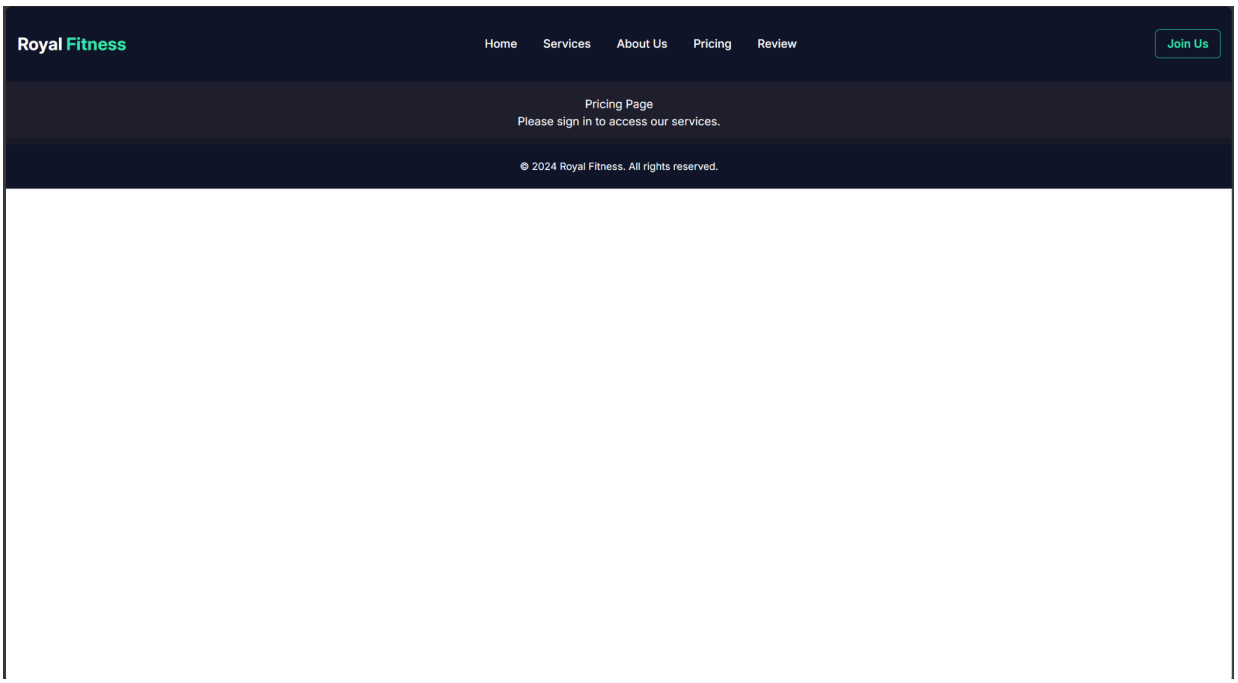
2.3 Feature / Application About Us Page:



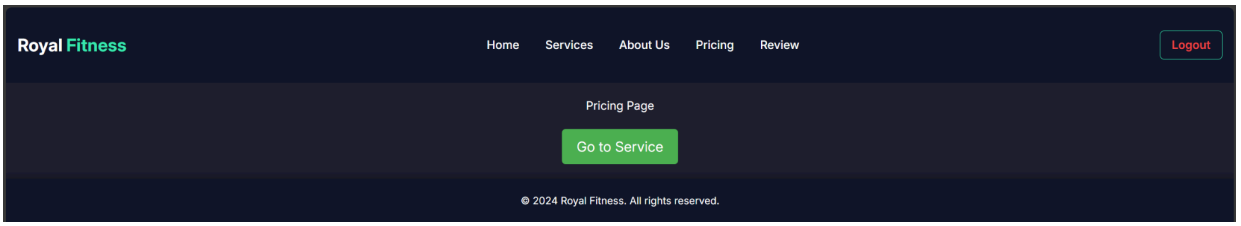
On the AboutUs page we post information about the two people who created this website and their pictures.

2.4 Feature / Application Pricing Page

Before login:



After login:



Implementation details

- **Frontend:**
  - **Vue.js:** Core library for building the frontend.
  - **Vue Router (*vue-router*):** Used for navigation between pages.
  - **Pinia (*@/stores/currentUserStore*):** State management for user data.
- **Backend:**
  - ***UserService*** handles API calls, which is likely a custom wrapper for HTTP requests (*using Axios*).

API Endpoint

- ***UserService.updateVip(user.value.id)*** is called when the user upgrades to VIP.

API Description

- **Endpoint**: Likely a REST API endpoint such as ***/api/users/{id}/vip*** (not explicitly provided in the code).
- **Method**: PUT or POST.
- **Request Data Format:** JSON payload likely includes the user ID to identify the account to be upgraded: ***"id":<user_id>***
- **Response Data Format:** Response status: HTTP 200 OK on success. JSON response confirming the update: ***"status": "success","vip": 1***

Read Data

- Reads the current user data ***(currentUserStore.currentUser)*** likely from a users

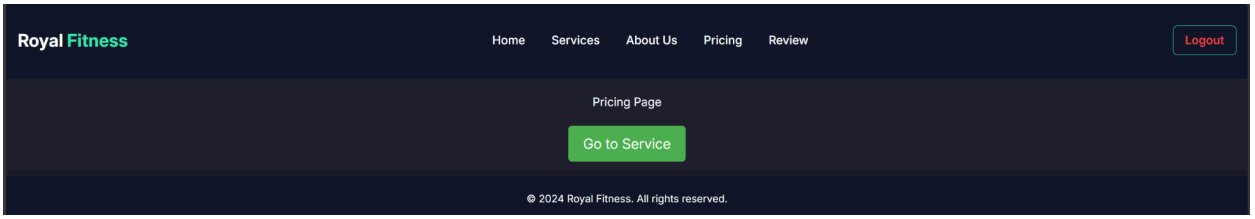
Store Data

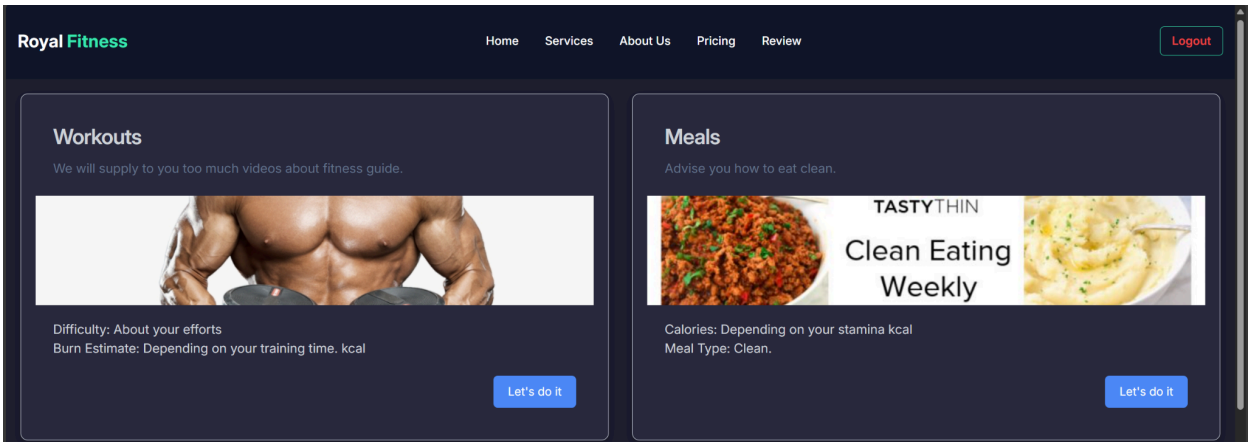
- Updates the ***vip*** status of the ***user*** in the same users table.

Client-side states

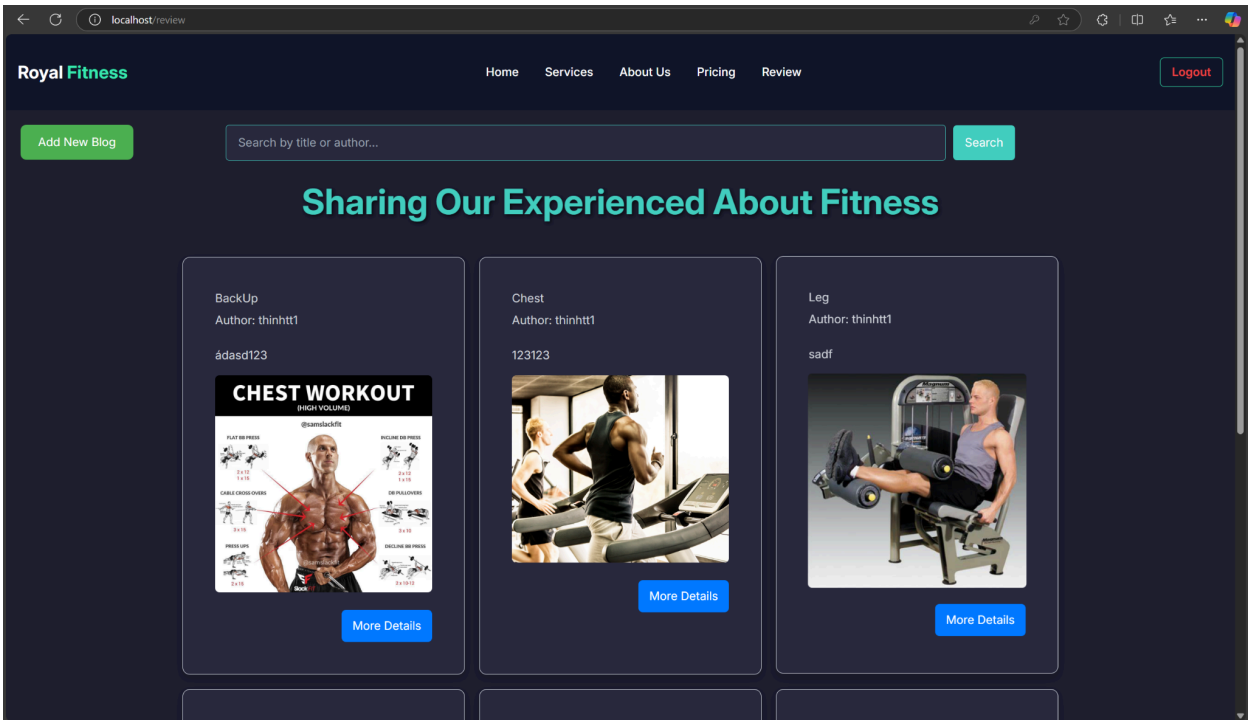
- **user:** Local reactive state in the **PricingPage** component, initialized from ***currentUserStore.currentUser.***
- **currentUserStore.currentUser:** Shared state managed by Pinia, representing the logged-in user's data.
- **user.vip:** Used to conditionally render buttons and check VIP status:
  + 0: Show "Click to Buy Service" button.
  + 1: Show "Go to Service" button.
- **Router state (useRouter):** Manages page navigation (navigating to /service).

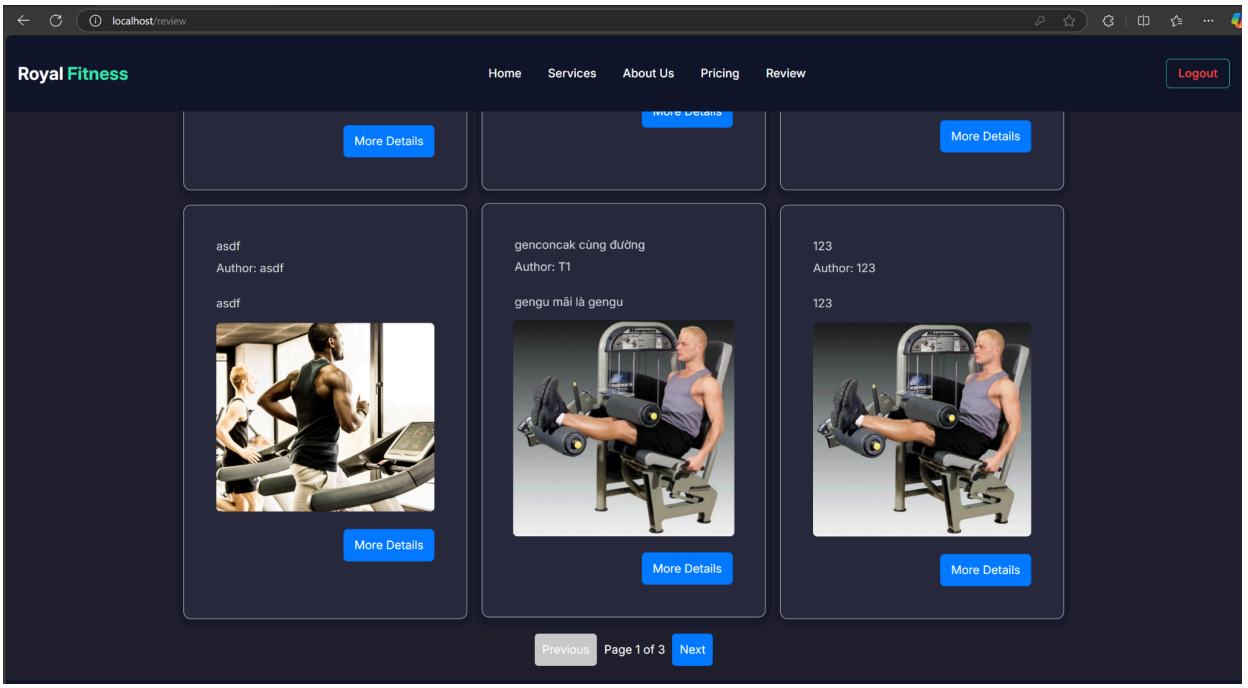
When the user got vip 0, “Click to buy Service” change to “Go to Service” and you can see the “Let’s do it” button to go to Workouts and Meals page.





2.5 Feature / Application Review





Implementation Details

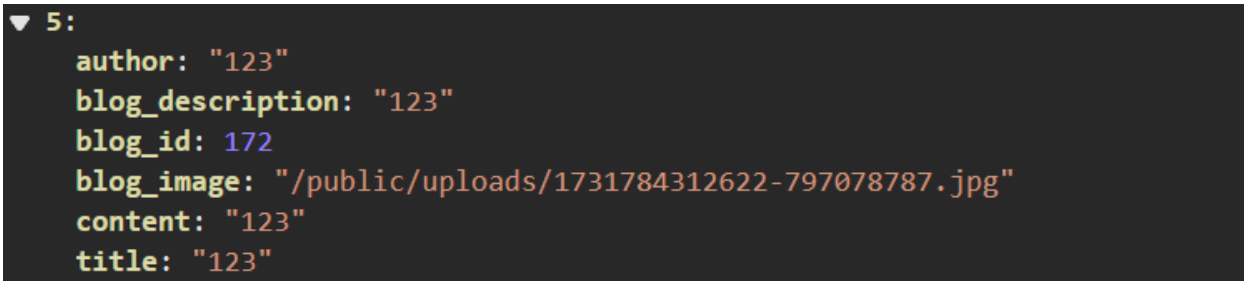
- **Frontend:**
  - **Vue.js:** Core library for building the component.
  - **Pinia (@/stores/currentUserStore):** Used for state management of the current user's data.
  - **Custom Components:** Card, CardHeader, CardContent, CardFooter (UI components for displaying blogs), Pagination (for handling paginated data), SearchBar (for searching blogs).
  - **BlogService:** A custom service for API calls to interact with blog data.
- **Backend:** Not explicitly provided in the code, but the BlogService likely uses an HTTP library (Axios) to communicate with APIs.

Fetch Blogs

- **Endpoint:** Likely ***/api/blogs?page={page}&limit={limit}&search={query}.***
- **Method:** GET.
- **Data Format Sent:** Query parameters for pagination and search.

Add Blog

- **Endpoint:** /api/blogs.
- **Method:** POST.
- **Data Format Sent:**

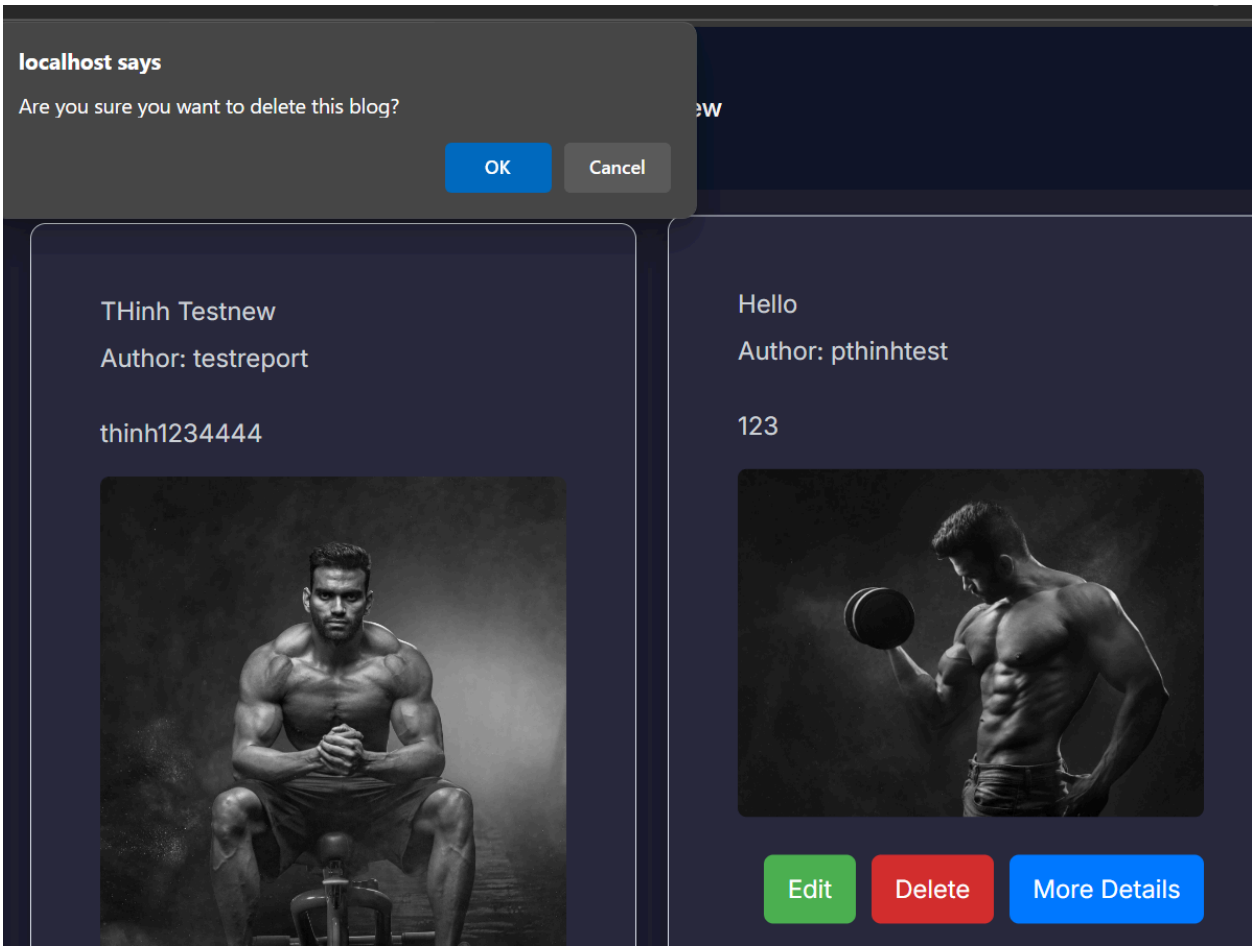


Update Blog

- **Endpoint:** */api/blogs/{blog_id}*.
- **Method:** PUT.
- **Data Format Sent:** Similar to Add Blog but includes the *blog_id*.

Delete Blog

- **Endpoint:** */api/blogs/{blog_id}*.
- **Method:** DELETE.
- **Data Format Sent:** *blog_id* as a path parameter.



Read Data

- Reads data from a blogs table or similar that includes the following columns:
  + blog_id (Primary Key)
  + title
  + blog_description
  + content
  + author
  + blog_image

Store Data

- Adds, updates, and deletes rows in the blogs table based on user actions.

States

- **posts:** Stores the list of blogs fetched from the API.
- **page:** Current page number for pagination.
- **limit:** Number of blogs per page.

- **totalPages:** Total number of pages available (calculated from the API response).
- **searchQuery:** Query string for searching blogs.
- **selectedPost:** Stores the details of a blog selected for viewing in a modal**.**
- **showAddPostModal:** Boolean to toggle the "Add Blog" modal.
- **showEditPostModal:** Boolean to toggle the "Edit Blog" modal.
- **newPost:** Form data for adding a new blog.
- **editPost:** Form data for editing an existing blog**.**
- **currentUser:** Logged-in user's data fetched from the store**.**
- **currentUserStore:** Pinia store for globally managing user data.

Using Pinia

- Pinia is used as a state management library to handle the global state of the currently logged-in user (*currentUser*). The *currentUser* store tracks user information (e.g., *username, VIP status*) and synchronizes it with *localStorage* to persist data across page reloads. This store includes actions like set *CurrentUser* to update the user state and save it to *localStorage*, and *clearUser* to remove the user data from both the store and *localStorage*. Pinia is initialized in the main.ts file, making the state globally accessible. In components like review.vue, the store is used to check if the logged-in user is the author of a post to display edit and delete options. In service-page.vue, it determines user permissions (e.g., VIP status) to render appropriate buttons such as "Let's do it" or "Buy Service." Pinia simplifies state management by offering a clean, reactive API without boilerplate, and its integration with *localStorage* enhances the user experience by retaining session data across reloads. This ensures a consistent, global state for user-related actions while avoiding prop drilling or redundant event handling.