

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	2
1. Project/application description:	2
2. A list of tables and their structures in the database (CDM diagram)	3
3. Task Assignment	3
II. Details of implemented features	4
1. API-Docs	4
1.1 Register	5
1.2 Login	6
1.3 Update Vip (used to buy service)	7
1.4 Get Meals	8
1.3 Delete Workout	9
2. Website	9
2.1 Feature / Application HomePage	9
2.2 Feature / Application Services Page	12
2.3 Feature / Application About Us Page:	14

2.4 Feature / Application Pricing Page	14
2.5 Feature / Application Review	17
 Using Pinia	21

I. Introduction

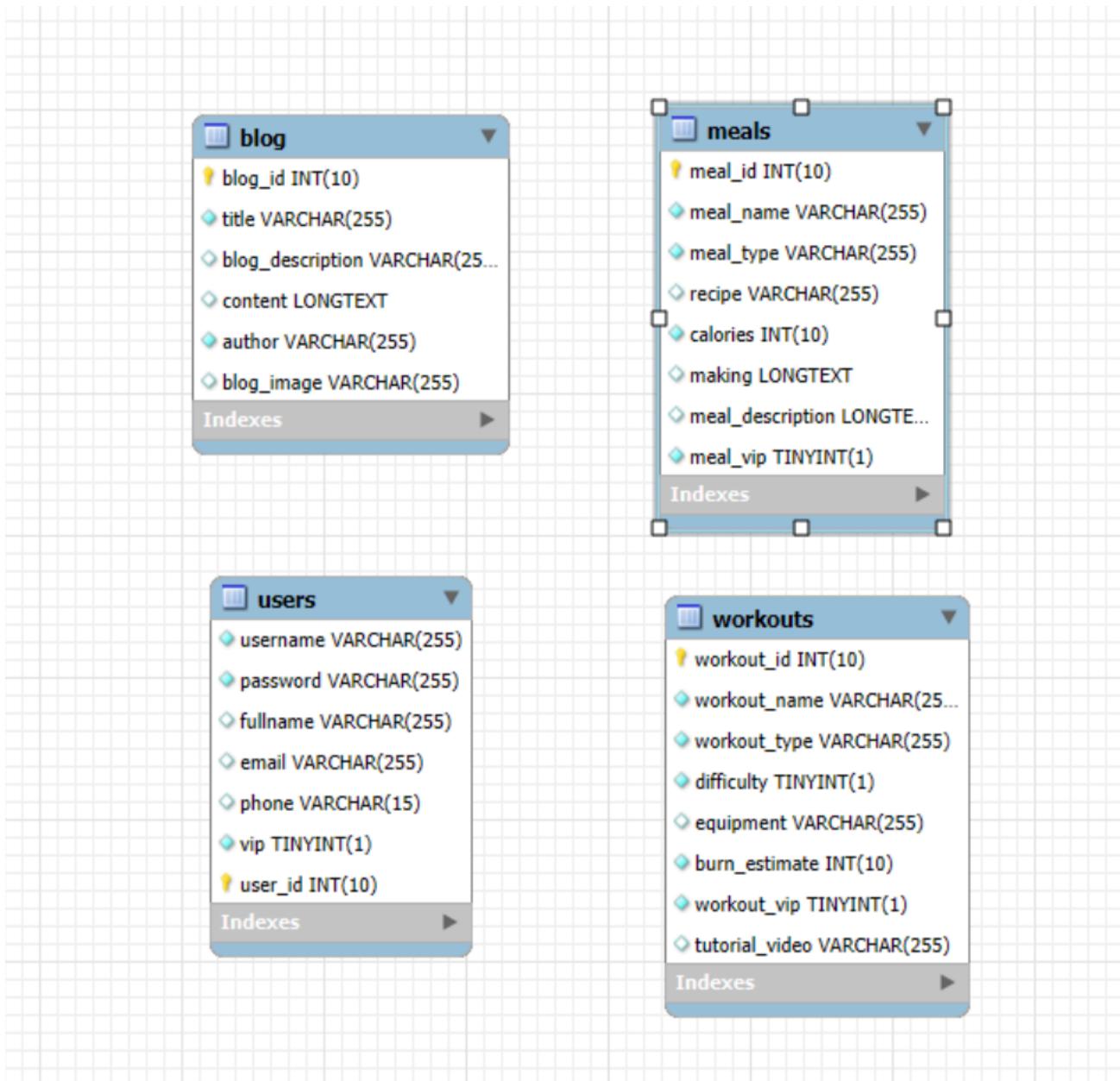
1. Project/application description:

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	<ul style="list-style-type: none">- Database Design and Implementation- Back-End Development- Testing and Deployment
Nguyen Phu Thinh	<ul style="list-style-type: none">- Database Design and Implementation- Front-End Development- Testing and Deployment

II. Details of implemented features

1. API-Docs

Overview

Contact App API 1.0.0 OAS 3.1

A simple contact app API

Servers [http://localhost:3000 - Development server](http://localhost:3000) ▾

User ^

- GET** /auth/users Get users by filter
- PUT** /auth/vip/{user_id} Update VIP status for a user
- PUT** /auth/users/{user_id} Update user by ID
- DELETE** /auth/users/{user_id} Delete user by ID
- POST** /auth/register Create a new user
- POST** /auth/login Login user

Blog ^

- GET** /blog Get blogs by filter
- POST** /blog Create a new blog
- GET** /blog/{blog_id} Get a blog by id
- PUT** /blog/{blog_id} Update blog by ID
- DELETE** /blog/{blog_id} Delete blog by ID

Meal ^

- GET** /meals Get meals by filter
- POST** /meals Create a new meal
- GET** /meals/{meal_id} Get a meal by id
- PUT** /meals/{meal_id} Update meal by ID
- DELETE** /meals/{meal_id} Delete meal by ID

Workout ^

- GET** /workouts Get workouts by filter
- POST** /workouts Create a new workout
- GET** /workouts/{workout_id} Get a workout by id
- PUT** /workouts/{workout_id} Update workout by ID
- DELETE** /workouts/{workout_id} Delete workout by ID

Below are some important methods of the website:

1.1 Register

POST /auth/register Create a new user

Create a new user

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{ "username": "string", "password": "string", "fullname": "string", "email": "user@example.com", "phone": "string", "vip": 0 }
```

Responses

Code	Description	Links
201	A new user created successfully	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{ "status": "success", "data": { "user": { "user_id": 0, "username": "string", "password": "string", "fullname": "string", "email": "user@example.com", "phone": "string", "vip": 0 } } }
```

1.2 Login

POST /auth/login Login user

Authenticates a user with username and password.

Parameters

No parameters

Request body required

```
{ "username": "tienanh", "password": "tienanh123" }
```

CANCEL Reset application/json

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: 194
content-type: application/json; charset=utf-8
date: Sun, 17 Nov 2024 10:31:32 GMT
etag: W/"58-admTood/k07k9lPxii0MrQ7hls"
keep-alive: timeout=5
x-powered-by: Express
```

1.3 Update Vip (used to buy service)

PUT /auth/vip/{user_id} Update VIP status for a user

Update the VIP status of a user by their user ID.

Parameters

Name	Description
user_id <small>* required</small>	The ID of the user to update

String (path) 2

Request body required

application/json

```
{ "vipStatus": 1 }
```

Servers

These operation-level options override the global server options.

http://localhost:3000 - Development server

Responses

Curl

```
curl -X 'PUT' \
'http://localhost:3000/auth/vip/2' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{ "vipStatus": 1 }'
```

Request URL

http://localhost:3000/auth/vip/2

Server response

Code **Details**

200 Response body

```
{ "status": "success", "data": { "user": { "user_id": 2, "username": "Coty_Mitchell", "password": "H6_Npuy5jd4Hlw", "fullname": "Bobby Olson DVM", "email": "Walter34@hotmail.com", "phone": "9169542608", "vip": 1 } } }
```

Download

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 195
content-type: application/json; charset=utf-8
date: Sun, 17 Nov 2024 10:33:30 GMT
etag: W/"c3-51h0h-Zg+vnv17rwFTsnauTnJCM"
keep-alive: timeout=5
x-powered-by: Express
```

1.4 Get Meals

GET /meals Get meals by filter

Get meals by filter

Parameters

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": "dominatio 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 sul articulus. Supellec deludo cerno cinis neque alveus qui.", "meal_description": "Vester tametsi apostolus. Acerbitas turba avertio comprehendo ubi in appositus decimus acidus. Corporis celebrer derideo odio audio confido confido scripti adipisci corrupti.\nVovo valetudo delego aestas spoliatio unus. Vicinus delectus alias pariatur. Ad ventosus carus virga canto arcus.\nCrinis utrimque vaco sopor providet t voluptates deleo commemoro. Argumentum demonstro aeneus convoco. Deserunt adiuv 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. Demorar vomer cernius vix tametsi suffoco crustulum.\nConsequatur concutus allatus. Vacantes necessitatibus brevis artificius thymbra demens totidem theatro cingitatio maxima. Verbera contactio amo stellata unitus theatrum admodum bellum in" } ] } }
```

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

The screenshot shows a REST API endpoint for deleting a workout. The URL is `DELETE /workouts/{workout_id}`. The description is "Delete workout by ID".
Parameters:
Name: `workout_id` (path) - required, type: integer, value: 3
Servers:
These operation-level options override the global server options.
Request URL: `http://localhost:3000` (Development server)
Responses:
Curl:

```
curl -X 'DELETE' \
'http://localhost:3000/workouts/3' \
-H 'accept: application/json'
```

Request URL:
`http://localhost:3000/workouts/3`
Server response:

Code	Details
200	Response body: <pre>{ "status": "success", "data": null }</pre> Download
	Response headers: <pre>access-control-allow-origin: * connection: keep-alive content-length: 32 content-type: application/json; charset=utf-8 date: Sun, 17 Nov 2024 10:37:37 GMT etag: W/"28-bff5r/a5MyHMyghjn8a8pOLDxA" keep-alive: timeout=5 x-powered-by: Express</pre>

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!



Build Your Dream Physique

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



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

Sign In

You don't have account?

Submit

2.2 Feature / Application Services Page:

The screenshot shows a dark-themed web application for 'Royal Fitness'. At the top, there's a navigation bar with links for Home, Services, About Us, Pricing, Review, and a 'Join Us' button. Below the navigation, there are two main service cards.

Workouts: Described as 'We will supply to you too much videos about fitness guide.' It features an image of a muscular torso and arms. Below the image, text specifies 'Difficulty: About your efforts' and 'Burn Estimate: Depending on your training time. kcal'. A 'Buy Service' button is at the bottom right.

Meals: Described as 'Advise you how to eat clean.' It features images of a bowl of meat and a bowl of mashed potatoes. The text includes 'TASTYTHIN Clean Eating Weekly'. Below the images, it says 'Calories: Depending on your stamina kcal' and 'Meal Type: Clean.'. A 'Buy Service' button is at the bottom right.

Below these cards, a section titled 'Discover More About Our Services' lists benefits: 'Customized workout plans tailored for each individual', 'Nutritionist-approved meal plans', and '24/7 customer support and consultation'.

A section titled 'Join Our Community' lists benefits: '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', and 'Personalized fitness consultations'.

At the very bottom left, there's a small link labeled 'Why Choose Us?'

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

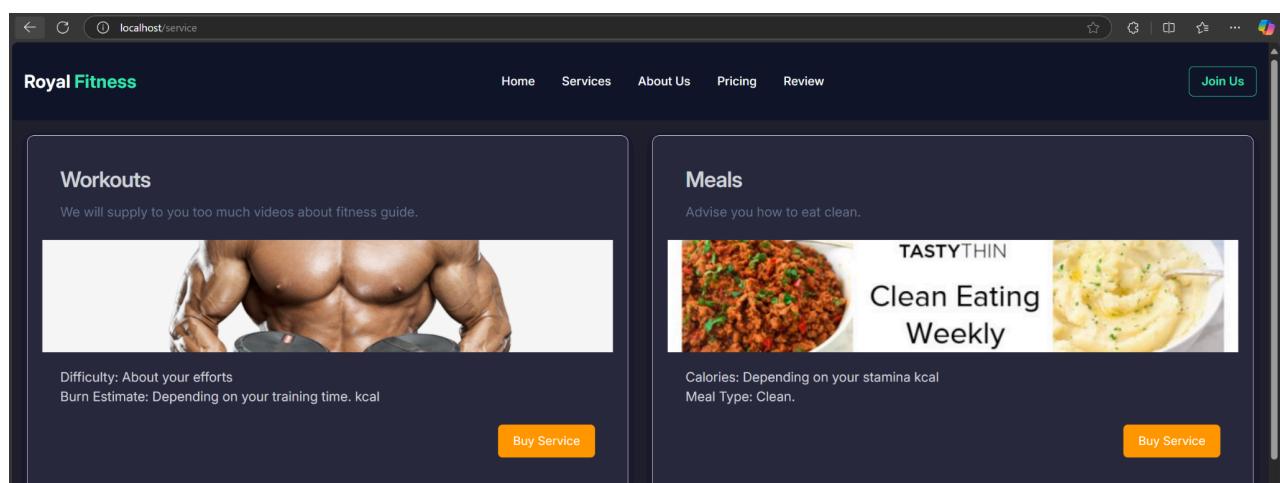
User Information: Reads the current user's data (`currentUserStore.currentUser`) to determine if the user has VIP access.

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:

The screenshot shows the 'About Us' section of the Royal Fitness website. At the top, there is a navigation bar with links to Home, Services, About Us, Pricing, Review, and a 'Join Us' button. Below the navigation, there are two profile cards side-by-side.

Profile 1: Nguyen Phu Thinh

Profile 2: Cao Tien Anh

Profile 1 Bio:

Full Name: Nguyen Phu Thinh
Date of Birth: 11/13/2003
Gender: Male
Zodiac Sign: Scorpio
Hobbies: Listening to music, gaming, learning web development from Teacher Bao, badminton, and gym with the guy on the right
Thoughts: I believe that working out at the gym is truly an effective way to improve health. I've been going to the gym for a year, and I think we all need to stay committed to exercise to achieve the health and body we desire. This is one of the main reasons why the two of us created this website – for gym enthusiasts to share experiences, learn from one another, and enjoy the benefit of our curated workout and meal plans that help support everyone's fitness journey. Let's keep grinding!

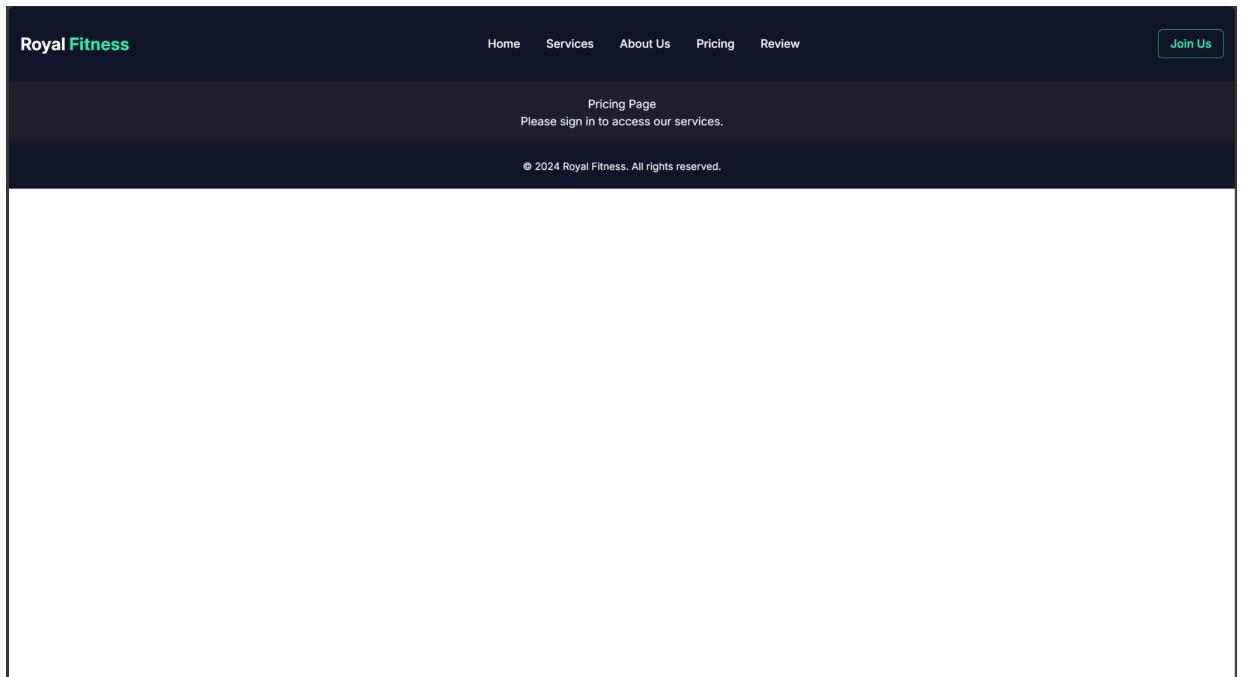
Profile 2 Bio:

Full Name: Cao Tien Anh
Date of Birth: 20/09/2003
Gender: Male
Zodiac Sign: Virgo
Hobbies: Football, Badminton, Gym, Code, and Music
Thoughts:

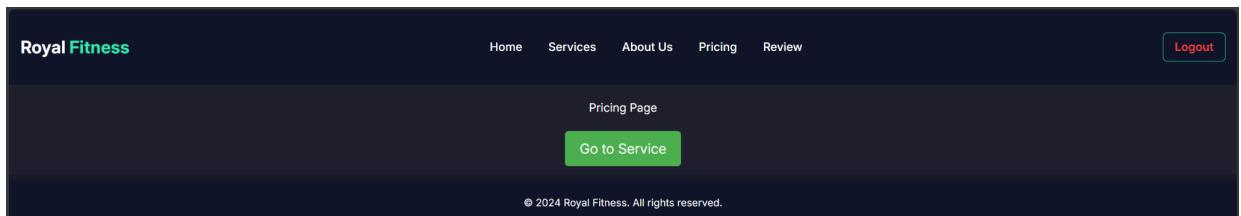
On the AboutUs page we post information about the two people who created this website and their pictures.

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

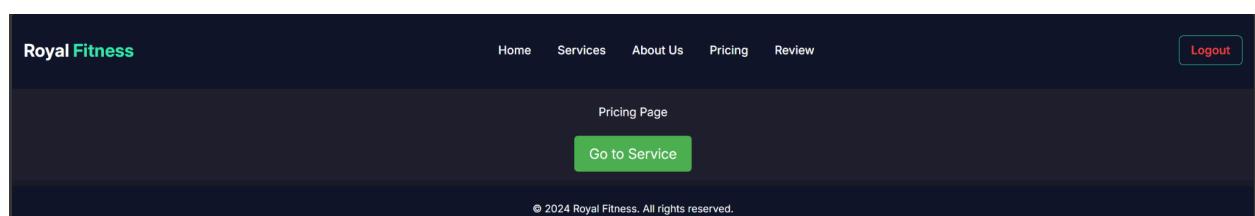
Store Data

- 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.



Royal Fitness

Home Services About Us Pricing Review Logout

Workouts

We will supply to you too much videos about fitness guide.



Difficulty: About your efforts
Burn Estimate: Depending on your training time. kcal

[Let's do it](#)

Meals

Advise you how to eat clean.



TASTYTHIN
Clean Eating Weekly

Calories: Depending on your stamina kcal
Meal Type: Clean.

[Let's do it](#)

2.5 Feature / Application Review

localhost/review

Royal Fitness

Home Services About Us Pricing Review Logout

Add New Blog Search by title or author... Search

Sharing Our Experienced About Fitness

BackUp
Author: thinh1t1
ádasd123



CHEST WORKOUT
(HIGH VOLUME)

PLAT DE PRESS 2x12 2x12 2x12
CABLE CROSS OVERS 3x15 3x15 3x15
PRESS UPS 2x12 2x12 2x12

INCLINE DE PRESS 2x12 2x12 2x12
DE PULLOVER 3x10 3x10 3x10

[More Details](#)

Chest
Author: thinh1t1
123123

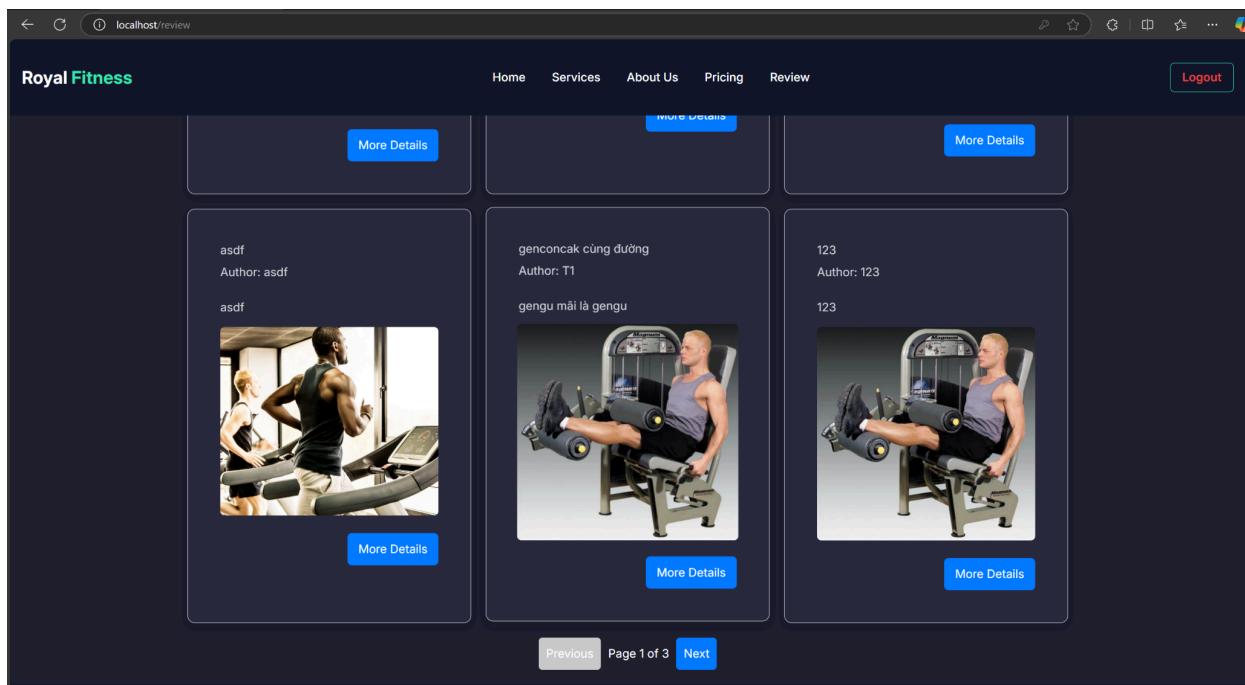


[More Details](#)

Leg
Author: thinh1t1
sadf



[More Details](#)



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.

```

Fetched Blogs: ▼ Object ⓘ
  ▼ blogs: Array(6)
    ▼ 0:
      author: "thinhhtt"
      blog_description: "adasd123"
      blog_id: 163
      blog_image: "/public/uploads/1731698030893-531248886.jpg"
      content: "siuuuuu123"
      title: "Backup"
    ► [[Prototype]]: Object
    ▶ 1: {blog_id: 164, title: 'Chest', blog_description: '123123', content: '123123', author: 'thinhhtt', ...}
    ▶ 2: {blog_id: 165, title: 'Leg', blog_description: 'asdf', content: 'adf', author: 'thinhhtt', ...}
    ▶ 3: {blog_id: 170, title: 'asdf', blog_description: 'asdf', content: 'asdf', author: 'asdf', ...}
    ▶ 4: {blog_id: 171, title: 'genconcak cùng đường', blog_description: 'gengu mãi là gengu', content: '88848', author: 'T1', ...}
    ▶ 5: {blog_id: 172, title: '123', blog_description: '123', content: '123', author: '123', ...}
      length: 6
    ► [[Prototype]]: Array(0)
  ▶ metadata: {totalRecords: 14, firstPage: 1, lastPage: 3, page: 1, limit: 6}
  ► [[Prototype]]: Object

```

Add Blog

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

```

▼ 5:
  author: "123"
  blog_description: "123"
  blog_id: 172
  blog_image: "/public/uploads/1731784312622-797078787.jpg"
  content: "123"
  title: "123"

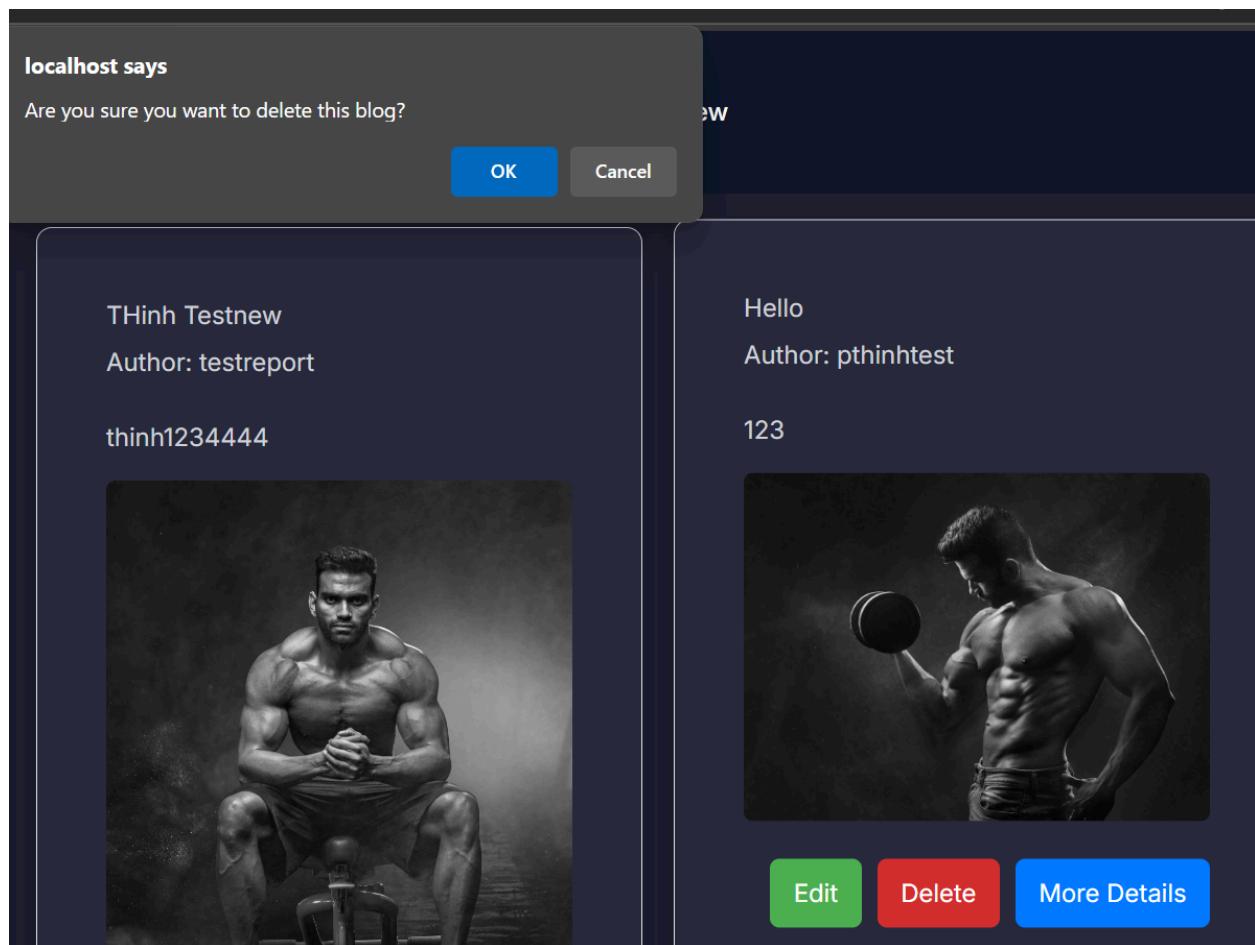
```

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.