САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2

Разработка одностраничного веб-приложения (SPA) с использованием фреймворка Vue.JS

Выполнил:
Ле Хоанг Чыонг

Группа:
K33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача: Мигрировать ранее написанный сайт на фреймворк Vue.JS.

Минимальные требования:

- Должен быть подключён роутер
- Должна быть реализована работа с внешним API
- Разумное деление на компоненты
- Использование composable

Ход работы

Развертывание маршрутизатора для проекта Vue:

```
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(),
  routes: [
    {
      path: "/",
      name: "home",
      component: () => import("../views/HomePage.vue"),
    },
    {
      path: "/login",
      name: "login",
      component: () => import("../views/LogInPage.vue"),
    },
    {
      path: "/signup",
      name: "signup",
      component: () => import("../views/SignUpPage.vue"),
    },
    {
      path: "/result",
      name: "result",
      component: () => import("../views/ResultPage.vue"),
    },
    {
      path: "/profile",
      name: "profile",
      component: () => import("../views/ProfilePage.vue"),
    },
```

Рисунок 1 - Роуты основные роуты приложения

Используйте app.use(router) для интеграции маршрутизатора в основное приложение.

```
import { createApp } from "vue";

import App from "@/App.vue";
import router from "@/router";
import pinia from '@/stores/index.js'
import CanvasJSChart from '@canvasjs/vue-charts';
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap";

const app = createApp(App);

app.use(pinia)
app.use(router);
app.use(CanvasJSChart);
app.mount("#app");
```

Рисунок 2 - app.use(router)

Развертывание работы с внешним API. Я создал класс под названием authApi, который имеет методы входа и регистрации для отправки POST-запросов к конечным точкам входа и регистрации. Этот класс предназначен для работы с экземпляром Axios (this.API) для обработки HTTP-запросов.

```
export class AuthApi {
  constructor(api) {
    this.api = api
  }

  async login(data) {
    return this.api({
      url: '/login',
      method: 'POST',
      data,
      headers: {
        'Content-Type': 'application/json'
      }
    })
  }

  async signUp(data) {
    return this.api({
      url: '/register',
      method: 'POST',
      data,
      headers: {
        'Content-Type': 'application/json'
      }
    })
  }

}
```

Рисунок 3 - authApi

Класс с именем расчетApi имеет методы getCalculations и postCalculation для отправки запроса GET, POST для получения всех расчетов или отправки его для сохранения.

```
export class CalculationApi {
  constructor(api) {
    this.api = api;
  }


  async getCalculations() {
    return this.api({
      url: `/calculations`,
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
      },
    });
  }


  async postCalculation(data) {
    return this.api({
      url: `/calculations`,
      method: 'POST',
      data,
      headers: {
        'Content-Type': 'application/json',
      },
    });
  }
}
```

Рисунок 3 - CalculationApi

Деление на компоненты:

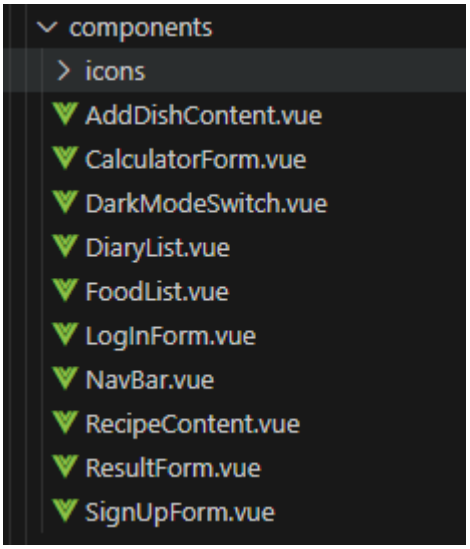
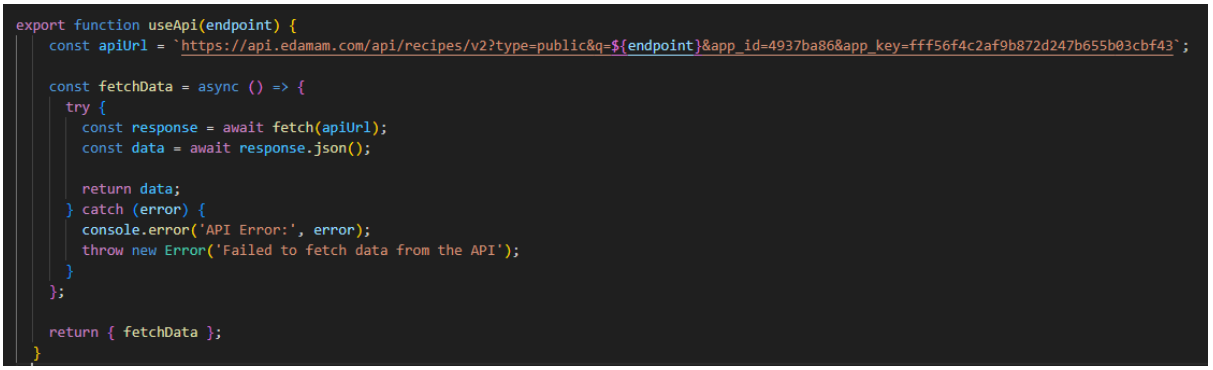


Рисунок 4 - Components

В папке «компоненты» я буду хранить исходный код небольших, автономных и многократно используемых компонентов интерфейса моего приложения. Каждый компонент может выполнять определенную функцию или отображать небольшую часть пользовательского интерфейса. Для моего проекта, например, форма регистрации, форма входа, интерфейс кнопки переключения темы, icons…

Composable

В моем проекте я создал компоновку, чтобы переписать логику, связанную с вызовами API. Функции useApi и useApiRecipe являются **composables**, предназначенными для взаимодействия с API для сбора данных о блюдах и их рецептах..



```
export function useApi(endpoint) {
    const apiUrl = `https://api.edamam.com/api/recipes/v2?type=public&q=${endpoint}&app_id=4937ba86&app_key=fff56f4c2af9b872d247b655b03cbf43`;

    const fetchData = async () => {
      try {
        const response = await fetch(apiUrl);
        const data = await response.json();

        return data;
      } catch (error) {
        console.error('API Error:', error);
        throw new Error('Failed to fetch data from the API');
      }
    };

    return { fetchData };
  }
```

Рисунок 5 - useApi

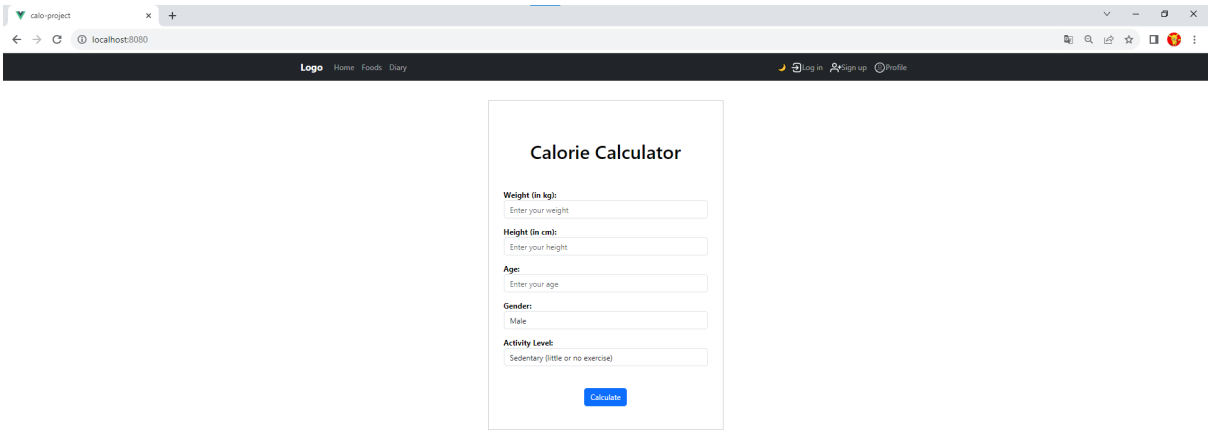
```
export function useApiRecipe(endpoint) {
    const apiUrl = `https://api.edamam.com/api/recipes/v2/${endpoint}?type=public&app_id=4937ba86&app_key=fff56f4c2af9b872d247b655b03cbf43`;
    console.log(endpoint)
    const fetchRecipe = async () => {
        try {
            const response = await fetch(apiUrl);
            const data = await response.json();

            return data;
        } catch (error) {
            console.error('API Error:', error);
            throw new Error('Failed to fetch data from the API');
        }
    };

    return { fetchRecipe };
}
```

Рисунок 6 - useApiRecipe

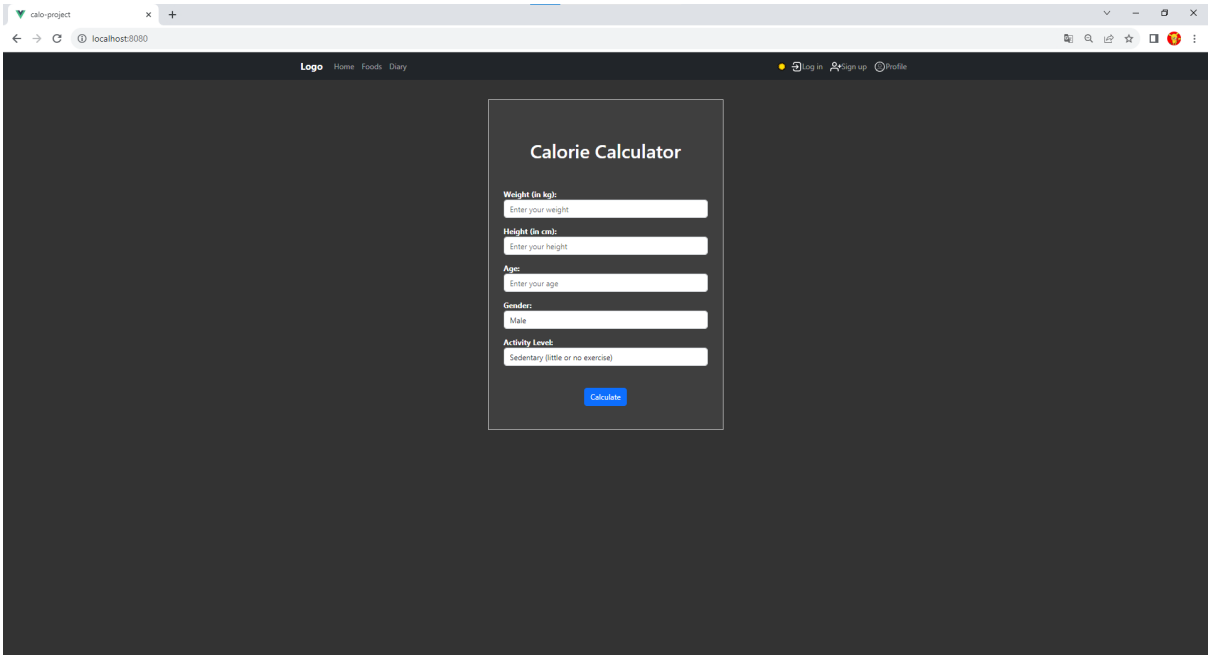
Сайты после завершения

Home Page

Light theme

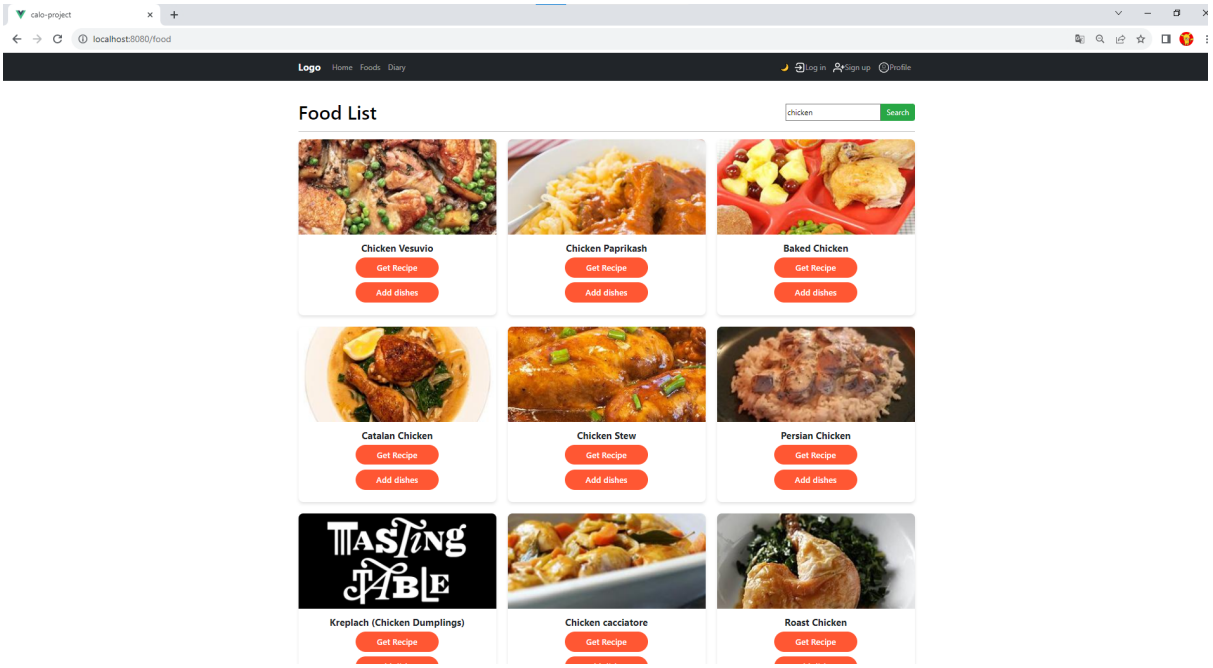


Dark Theme

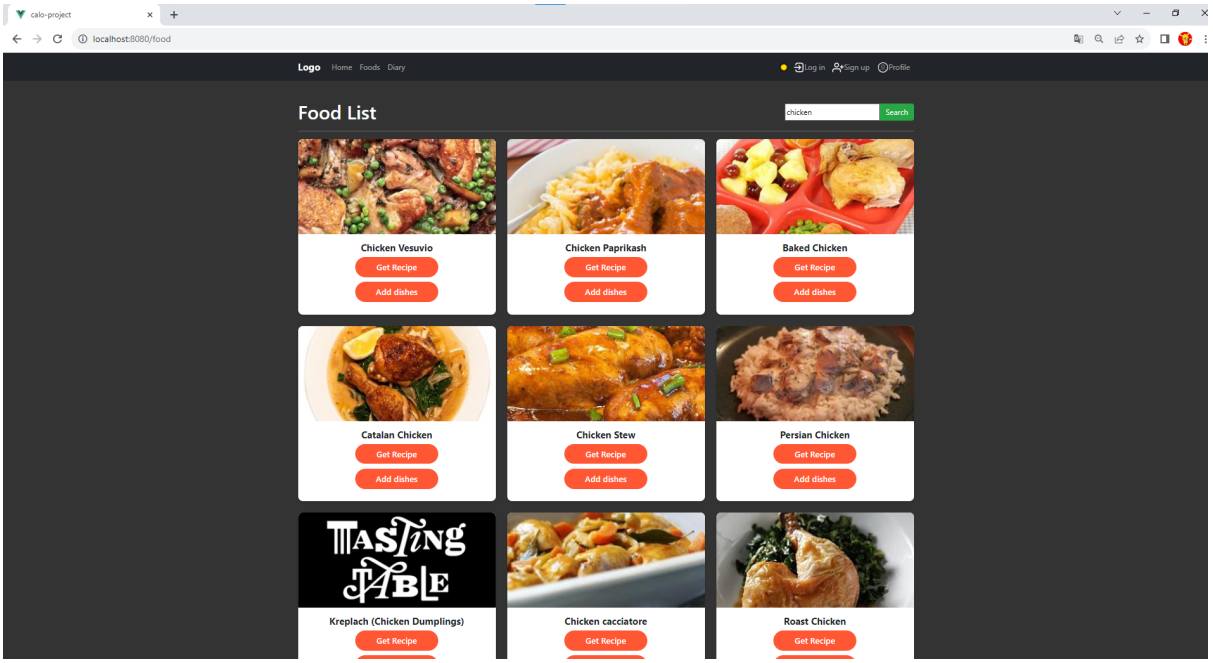


Food Page

Light theme

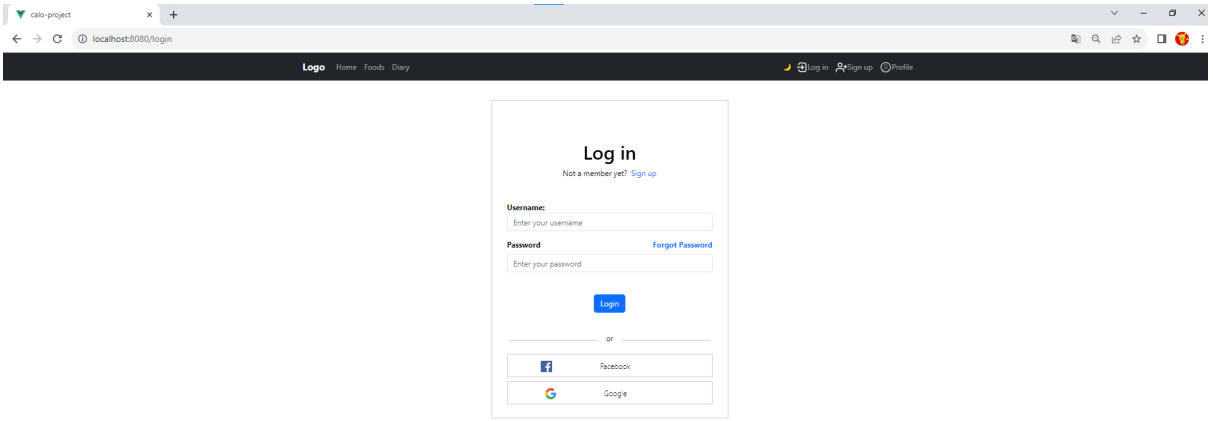


Dark Theme

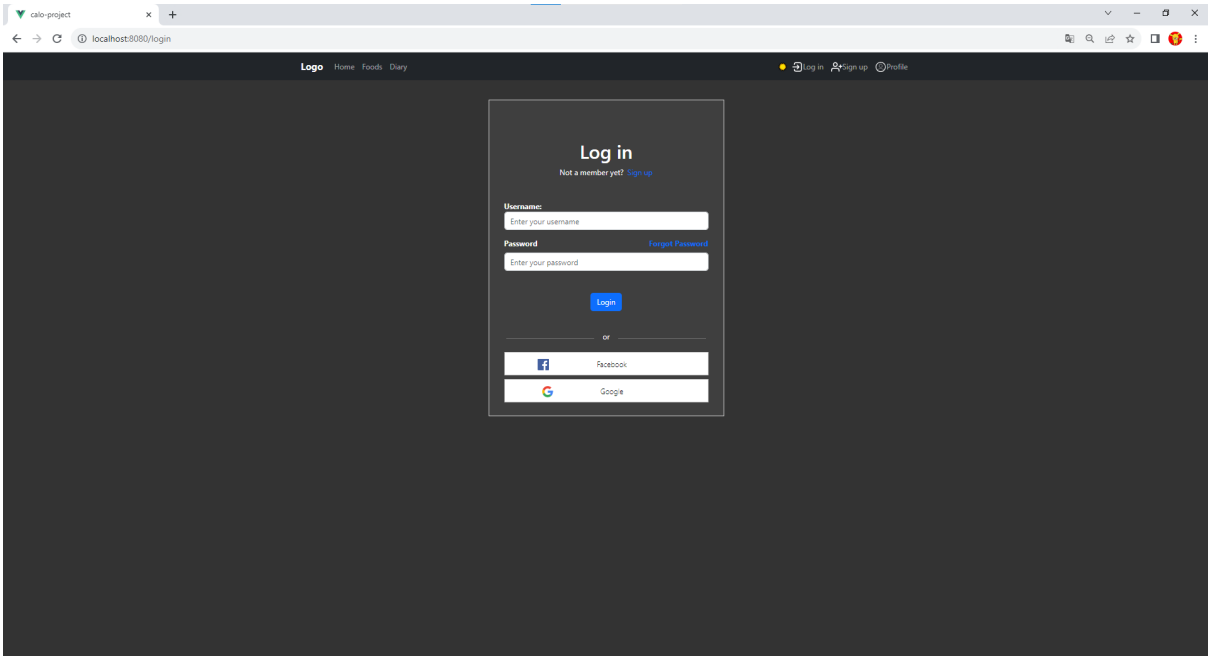


Login Page

Light Theme

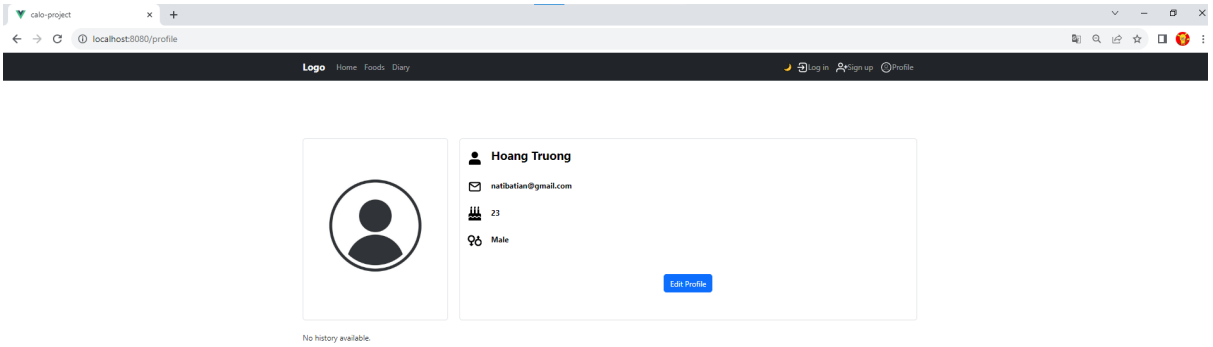


Dark Theme

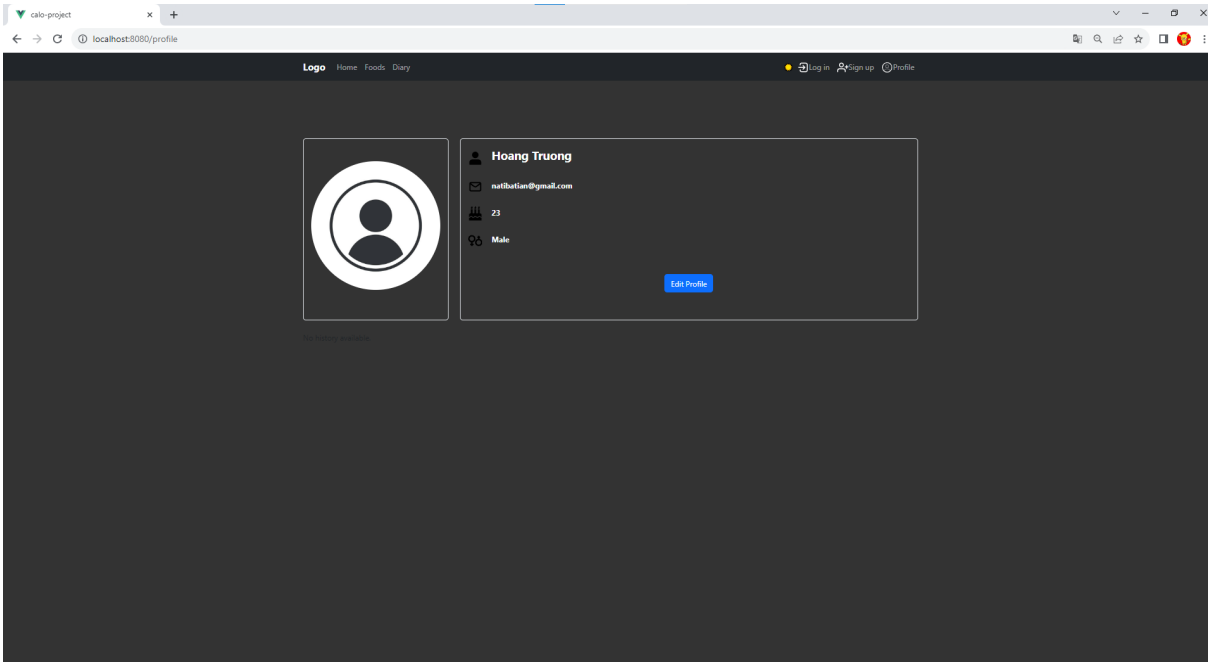


Profile Page

Light theme



Dark theme



Вывод

В ходе лабораторной работы №3 я научился развертывать веб-сайты с помощью платформы VueJS, научился развертывать маршрутизаторы, компонуемые объекты и работать с внешними API.