# Contents

# 1 Abstract

This project is about designing and implementing the website of a nonprofit organization called Anemone, whose purpose is to assist women who are abused or in need of assistance. This document mainly introduces the technical aspects of the website, including the hosting service, structure, external libraries, and other relevant details.

## 1.1 Front end

Our website mainly use Vue.js 3 and Nuxt.js to create dynamic and responsive interfaces.

## 1.2 Back end

Our website uses Node.js as the server to implement SSR (Server-Side Rendering) and to provide REST API services.

# 2 Hosting Service

We have chosen Vercel and Supabase as our website and database hosting services.

## 2.1 Website service - Vercel

We opted for Vercel because it is easy to use and offers a free tier for small websites like ours. Vercel provides a seamless deployment process, automatic Git integration, and a global CDN, which ensures fast load times for our users. While Azure and AWS offer more extensive services and customization options, they are better suited for complex, large-scale, or highly customized applications. For our needs, Vercel's simplicity and efficiency make it the perfect choice.

## 2.2 Database service - Supabase

Similarly, we selected Supabase for our database hosting. Supabase provides the simplicity and ease of use we require, along with real-time capabilities and built-in authentication. It offers a comprehensive set of features that cater to our needs without the complexity of larger platforms. Like Vercel, Supabase has a free tier that is suitable for our project size, making it an excellent match for our requirements.

## 2.3 Render Mode - SSR

We adopt SSR in our project because it provides fully rendered HTML to search engines, allowing them to index the content more effectively. This is crucial for SEO optimization. In our project, we hope that more and more people can find it through search engines like Google and Bing to help hopeless women. Thus, SSR is the best choice compared to CSR and SSG.

# 3 Project Structure

The subsection introduces the project structure , pages directory structure and server endpoint.

## 3.1 Overview Structure

1. assets/: This folder is for uncompiled assets such as public js and css files.
2. components/: This folder is for Vue.js components. Components are reusable elements that you can use across different parts of your application.
3. layouts/: This folder contains layout components. Layouts are used to define the structure of your pages. For instance, you might have a default layout and a dashboard layout.
4. pages/: This folder contains your application's views and routes. Each file in this directory automatically becomes a route in your app.
5. plugins/: This folder is for JavaScript plugins that need to be run before instantiating the root Vue.js application.
6. public/: This folder contains static files used to serve publicly that you want to be publicly.
7. server/: This folder is used for server-side related code, such as API routes or middleware.
8. SQL/: This folder seems to be intended for database-related files, which might include SQL scripts or database schema files.

## 3.2 Page directory Structure

1. people/index.vue: This file defines the /people route, used to display a list of employees.
2. people/[id].vue: This dynamic route file handles URLs like /people/:id, where :id is a placeholder for a specific person's identifier. It is used to display the details of a specific employee.
3. projects/index.vue: This file defines the /projects route, used to show a list of projects in the website.
4. projects/[id].vue: This dynamic route file handles URLs like /projects/:id, where :id is a placeholder for a specific project's identifier. It displays the details of a specific project.
5. services/index.vue: This file defines the /services route, which typically shows a list of services offered.
6. services/[id].vue: This dynamic route file handles URLs like /services/:id, where :id is a placeholder for a specific service's identifier. It displays details about a specific service.

7. assistant.vue: This file defines the /assistant route, function as the chatbot in the website.

8. center.vue: This file defines the /center route, which describe the website center explanation.

9. request.vue: This file defines the /request route, which is used to handle help request form.

10. FAQ.vue: This file defines the /faq route, which is used to display the FAQ page.

11. contacts.vue: This file defines the /request route, which is used to display contacts page.

### 3.3 Server Endpoint

The website only shows the information including the employees, service and projects, thus all APIs are GET methods.

1. GET /api/employee?id={id}: Fetches the full content of a specific exployee identified by its ID. If no ID is provided, it returns all employees that match the query condition. Example:

```
{
 "data": [
 {
  "id": 1,
  "name": "Sofia Bianchi ",
  "role": "Director",
  "pic": "/sofia.jpg",
  "cv": "Sofia Bianchi, our esteemed Director...",
  "service": [],
  "project": []
 }
 ]
}
```

2. GET /api/project?id={id}: Fetches the full content of a specific project identified by its ID. If no ID is provided, it returns all projects that match the query condition. Example:

```
{
 data:[{
  id:1,
  name:"EmpowerHer",
  description:"The EmpowerHer program i... resilient communities.",
  tag:"Economic Empowerment Program",
  "pic": "/EmpowerHer.jpg",
 }]
}
```

3. GET /api/service?id={id}: Fetches the full content of a specific service identified by its ID. If no ID is provided, it returns all services that match the query condition. Example:


```
{
 "data": [
 {
  "id": 1,
  "name": "Work",
  "description": "Our Work Services...",
  "availability": "from 9:00 am to 5:00 pm",
  "pic": "...",
  "tag": "..."
 }
 ]
}
```


4. GET /api/testimonial?serviceID={id}: Fetches the full testimonial of a service identified by its ID. Example:


```
{
 "data": [
 {
  "id": 1,
  "serviceID": 1,
  "comment": "Anemone empowered me to restart my career safely",
  "name": "Arianna",
  "age": 24
 },
 ]
}
```

# 4 Implemented Components

Those components are made for better code organization.

1. navbar(components/anemoneNavbar.vue): This component is used to render the header section of the website, including the logo and navigation menu. It works responsively in PC and mobile views.

2. footer(components/anemoneFooter.vue): This component renders the footer section, providing links to important pages like the privacy policy, terms of service, and contact information.

3. loading(components/LoadingPlaceholder.vue): This component works when the page or content is rendering.

4. request button(components/RequestButton.vue): This component works when clicking the request button in index page.

# 5 External Libraries

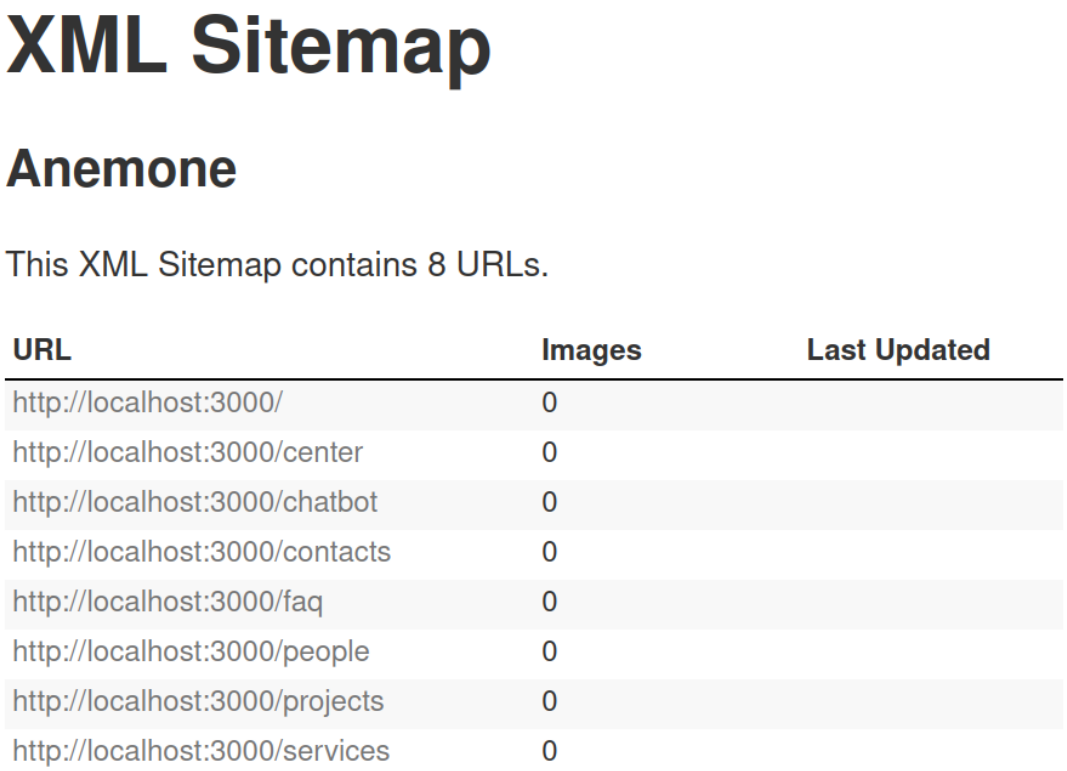
Apart from Vue.js 3 and Nuxt.js, the following external libraries were also imported:

1. @nuxtjs/supabase: It is primarily used to connect to and interact with the Supabase database, and it works on our Node.js server.
2. less.js: It's used to compile LESS code into CSS, enabling easier and more efficient styling for the website, we use it to organize our stylesheet in creating CSS-like components, enabling modular styling.
3. Tailwind.css: It's used to quickly style the website using utility-first CSS classes, allowing for rapid and responsive design without writing custom CSS.
4. @nuxtjs/sitemap: It's used to enhance SEO by helping search engines better crawl and index our website's pages.
5. vue3-carousel: It's used to support carousel function in service, project and people pages.
6. nuxt-phosphor-icons: It provides extra icons used in the website.

# 6 SEO and Accessibility

## 6.1 SEO

Apart from using Server-Side Rendering (SSR), we also use dynamic titles, meta keywords, and a sitemap to boost SEO.



XML Sitemap

Anemone

This XML Sitemap contains 8 URLs.

| URL | Images | Last Updated |
|---|---|---|
| http://localhost:3000/ | 0 | |
| http://localhost:3000/center | 0 | |
| http://localhost:3000/chatbot | 0 | |
| http://localhost:3000/contacts | 0 | |
| http://localhost:3000/faq | 0 | |
| http://localhost:3000/people | 0 | |
| http://localhost:3000/projects | 0 | |
| http://localhost:3000/services | 0 | |

**Figure 1**

The sitemap of our website.

## 6.2 Accessibility

We did some work on accessibility:

1. Provide Alternative Text: Ensure all images have descriptive alt text so screen readers can convey the information to visually impaired users.
2. Use Semantic HTML: Use semantic HTML tags like header, nav, main, article, and section to help screen readers understand the structure of the page.
3. Add ARIA (Accessible Rich Internet Applications) Tags