Github Portfolio Website Development Log

Matthew Sylvester

January 2026 – current

1 Scaffolding

React + vite with tailwindcss for styling. See tutorial on setup & basic page structuring followed here.

1.1 React requirements

Installing requirements and setting up the React development workspace

- **Node.js**; Javascript runtime. Installed from https://nodejs.org/en/download
- **npm CLI**; package manager for Javascript, installed automatically with node.js but releases will not necessarily automatically update.
- **Vite**; build tool for web projects (web server & build command which bundles code with Rollup). Installed through npm CLI. Documentation: https://vite.dev/guide/

Using typescript as the default language for all pages, be aware that the tutorials I've followed are often for javascript/jsx so I may not be properly using typescript. default file extension is tsx unless required to be otherwise.

Begin by installing the Node.js runtime. Make sure npm package manager is included in the installer. Once Node.js is installed, open a terminal in the workspace for the project in the code editor and run

```
npm create vite@latest my-app -- --template react
```

Note that this command needs to be run only to set up vite and create the index.html file.

For the website, the options selected were

1. use rolldown-vite?: yes
2. hosted on http://localhost:5173/

use –host to expose.

When opening the project for a development session, you will need to start the dev server with the following command:

```
npm run dev
```

1.2 Styling

Lucide-react is installed for icons.

2 Deployment

Deployed as a github pages page in fourthstanza.github.io repo using default vite github pages workflow.

3 Structure

3.1 Navigation bar

Separated into its own component in *layout/navbar.tsx* for reusability across pages. Contains links to main sections of the site.

Has split styling for desktop and mobile views. Mobile view is a hamburger menu that expands to show links when clicked. See use of md: for handling switching between desktop and mobile views.

Animated with CSS styling for fade-in and fade-out when opening and closing the mobile menu. See utilities layer.

3.2 EN/FR button

Manual implementation of language switching button in the navbar. Implemented by switching between two dictionaries (and sometimes directly as a single string switch) in each page. Static elements not handled by the router have built-in language switching by analyzing the URL and changing which text is displayed based on the language prefix.

3.3 Main pages

Site is separated into sections via the *sections* folder. The sections are as follows.

- **about**; the index and primary page. Contains information about me and a summary of my experience/projects
- **projects**; summary of projects I'm currently working on. Contains additional project pages that go into further detail, links to other resources

- **gallery**; gallery of pictures I've taken
- **contact**; contact & location information (pull template & framework from somewhere?)

3.4 Subpages

Gallery and Projects pages set up with helper components to handle routing to subpages for each Projects and Gallery entry. Each individual gallery and project has its own page in the corresponding sections subfolder

3.5 Footer

Basic footer element contains blurb, duplicate nav, and social links.

4 Routing

Using HashRouter from react-router to handle routing between pages. Done because github pages do not support client-side routing without hashrouter.

5 Tools

Emails handled using email.js.
installed through npm with npm i @emailjs/browser.
Configured by adding keys to .env /(not pushed to the github repo/) which are imported and used when calling emailjs to send the email. Sends an email to my address using the template defined in the emailjs web interface and specified with an env variable.

6 Maintenance

6.1 Adding images to existing gallery

Find the gallery folder corresponding to the correct gallery in @/src/assets/images/. Place the images in the folder and resize them to be a few hundred kb. Imports are handled automatically by the corresponding galImages component.

6.2 Adding a new gallery

To add a new gallery, do the following:

1. Add a new gallery folder to the @/src/assets/images/ folder, and populate.
2. Add a new galImages component in @/src/sections/Galleries, and copy over the code, while changing relevant sections.

3. Add a new gallery component in @/src/sections/Galleries, and copy over the code, while changing relevant sections.

4. In @/src/sections/gallery.tsx,

    - add the new gallery to the pages array
    - Import the corresponding GalImages component
    - Add the new GalImages component to the images array
    - Add new english and french titles to the galleryTitles arrays
    - Add new english and french paragraphs tohe galleryParagraphs arrays

5. In @/src/sections/sections/galleryPages.tsx, import and add the new Gallery pages to the pages record