Full Portfolio Maintenance & Update Guide

Author: Syed Hassan Ahmed

1. Project Overview

This portfolio is a full-stack application consisting of a Flask backend and a static frontend. The backend serves APIs and hosts the frontend. The frontend consumes APIs to render dynamic content.

2. Folder Structure & Responsibilities

Root Structure:

```
Portfolio/ ■■■ Backend/ ■ ■■■ app.py ■ ■■■ requirements.txt ■ ■■■ data/ ■■■ Frontend/ ■■■ index.html ■■■ css/ ■■■ js/ ■■■ pages/ ■■■ assets/
```

3. Backend (Flask API & Hosting)

Backend/app.py

Controls API endpoints and serves frontend files. Any data-driven content is fetched from JSON files located in Backend/data.

- To add a new section (e.g., Volunteer): add a JSON file in Backend/data and create a new @app.get API route.
- To change API behavior: modify app.py.
- To deploy correctly on Render: Root Directory must be 'Backend'.
- Gunicorn entry point: gunicorn app:app.

Backend/data/

Each JSON file represents a section of the portfolio.

- profile.json → Name, headline, social links
- summary.json → Summary bullets on home page
- projects.json → All project cards and categories
- experience.json → Work experience timeline
- skills.json → Skill categories and chips
- certifications.json → Certifications list
- achievements.json → Achievements & awards
- volunteer.json → Volunteer work section

4. Frontend (UI & Interaction)

Frontend is a static site rendered by Flask. All visual and interaction logic lives here.

Frontend/index.html

Main landing page. Controls section order and which sections appear on homepage.

- To change section order: rearrange blocks.
- To hide a section from homepage: remove or comment its section block.
- Hero (name + resume) lives here.

Frontend/pages/

Contains standalone pages like about.html, projects.html, experience.html.

- Each page loads layout.js and pages.js.
- Used for navbar navigation and deep linking.
- If something shows on page but not homepage → index.html missing that section.

Frontend/js/

pages.js

Handles API calls and DOM rendering.

- To add a new section: create a render function here.
- Connect new API endpoints here.
- Filtering logic for projects lives here.

Frontend/css/style.css

Controls all UI styling: layout, spacing, colors, cards, grids.

- Overlapping issues → check .container display:flex and gap.
- Card spacing → .card margin or container gap.
- Icons styling → .about-links, svg size.

Frontend/assets/

Static assets like resume PDF and SVG icons.

- Resume PDF → Frontend/assets/docs/resume.pdf
- Icons → Frontend/assets/icons/*.svg
- Broken icons → path or case mismatch.

5. Common Change Scenarios

- Change name/title → profile.json
- Add project → projects.json
- Add project category → projects.json + filter button HTML + JS filter logic
- Add volunteer work → volunteer.json + API + JS render
- Fix styling issues → style.css
- Change navigation → layout.js + HTML
- Deploy errors → Render settings + app.py paths

6. Deployment (Render)

- Root Directory: Backend
- Build: pip install -r requirements.txt
- Start: gunicorn app:app
- Frontend is served automatically by Flask

This document is your single source of truth for maintaining and extending your portfolio.