Hello! I need help with my personal portfolio and blog website. It is built using the static site generator Eleventy. Here is a complete overview of the project structure, systems, and configuration.

1. Goal

The goal is to create a personal portfolio website for a Data Analyst named Farman Khan. The site will feature a homepage, portfolio, blog, about page, and contact page. It needs to be dynamic, pulling content from Markdown files, and should be fully responsive with a light/dark theme.

2. File Structure

The project is organized as follows:

- _includes/: Contains reusable templates.
  - layouts/base.html: The master layout file for all pages.
  - partials/header.html: The site's navigation header.
  - partials/footer.html: The site's footer.
- pages/: The main directory for all site content.
  - home.html: The homepage.
  - portfolio.html: The main listing page for all portfolio projects.
  - blog.html: The main listing page for all blog posts.
  - about.html, contact.html, 404.md: Static content pages.
- pages/posts/: A dedicated folder where all blog post .md files will be stored.
- pages/projects/: A dedicated folder where all project .md files will be stored.
- css/style.css: The single stylesheet for the entire site.
- img/: A folder for all images, organized into subfolders like posts/ and projects/.
- .eleventy.js: The main Eleventy configuration file.
- package.json: Manages Node.js dependencies.

3. Core Systems & Logic

- **Templating:** The site uses the **Nunjucks** templating language.
- **Collections:** Eleventy automatically creates two main collections:
  1. collections.posts: From all .md files inside pages/posts/.
  2. collections.projects: From all .md files inside pages/projects/.
- **Dynamic Listing Pages:**
  - portfolio.html and blog.html loop through their respective collections (collections.projects and collections.posts) to generate a grid of clickable cards.
  - home.html displays the **two most recent** items from each collection by using a custom limit filter: collections.projects | limit(2).
- **Error Handling:** The loops are wrapped in {% if collections.projects and collections.projects.length %} blocks to prevent the site from crashing if the collections are empty.

4. Styling and Features

- **CSS:** The site uses a single style.css file with CSS variables for easy theming (:root for light theme, .dark for dark theme). It is fully responsive using a media query for screens under 768px.
- **JavaScript:** All client-side JavaScript is located in a <script> tag in base.html and handles:
  - A light/dark theme toggle that saves the user's preference in localStorage.
  - A responsive mobile "hamburger" menu.
  - A "scroll-to-top" button.

5. Configuration (.eleventy.js)

The .eleventy.js file is configured to:

- Copy the css, js, and img directories to the final build folder (_site).
- Define a custom limit filter to slice arrays, which is essential for the homepage. The filter code is:
  ```
  eleventyConfig.addFilter("limit", (arr, limit) => (arr || []).slice(0, limit));
  ```

6. Content Structure (Markdown Front Matter)

All dynamic content is driven by front matter in the .md files. Each file must have the following structure:

- **For a project (pages/projects/some-project.md):**
  ```
  ---
  layout: layouts/base.html
  title: Name of the Project
  date: YYYY-MM-DD
  summary: "A short, one-sentence summary for the card view."
  tags: ["Tool 1", "Tool 2"]
  featuredImage: /img/projects/project-image.jpg
  featuredImageAlt: "A descriptive alt text for the image."
  ---

  Full project case study content in Markdown...
  ```
- **For a post (pages/posts/some-post.md):**
  ```
  ---
  layout: layouts/base.html
  title: Title of the Blog Post
  date: YYYY-MM-DD
  excerpt: "A short excerpt for the card view."
  featuredImage: /img/posts/post-image.jpg
  featuredImageAlt: "A descriptive alt text for the image."
  ```

```
---

Full blog post content in Markdown...
```

My task is to continue building and refining this website based on the established structure.