ChatGPT

Product Requirements Document (PRD): Al-Ameen Matriculation Higher Secondary School Website Redesign

Overview

Al-Ameen Matriculation HSS's existing site is only half-built and visually bland. The redesign will transform it into a professional, polished, production-ready website that reflects the school's identity. We aim for a modern design that remains culturally respectful, integrating subtle Islamic-inspired motifs and a sophisticated color palette. The site must be fully static and responsive, ensuring fast load times and compatibility across devices. Accessibility (WCAG 2.1 AA) is mandatory to accommodate diverse users [1]. Development will use semantic HTML5, modular CSS (BEM), and lightweight JavaScript for interactivity.

Visual Design and Branding

The aesthetic will incorporate subtle Islamic design elements such as geometric latticework, calligraphic flourishes, and arch motifs. Geometric patterns are a hallmark of Islamic art and architecture [2], emphasizing abstraction and unity. Colors will draw from a muted yet refined Islamic-inspired palette: shades of green, blue, ivory, sand, and complementary neutrals. In Islamic architecture, green and blue often convey spiritual symbolism, and gold and white highlight purity and light [3]. Overall the interface should feel clean and welcoming, while retaining clarity and modern simplicity.

The layout will leverage arches or minaret-like shapes sparingly as visual accents (e.g. in hero images, section dividers, or iconography). A tiled arch (like those in traditional mosque facades) can inspire both shape and color. Textures and patterns (e.g. stone screens or tile motifs) may appear as subtle backgrounds or borders to add cultural depth. Typography will be clean and legible; for instance, pair a clear sans-serif body font with a classic serif or Naskh-style heading font. Components should feel polished and harmonious with the color scheme.

Site Architecture and Content Structure

- **Home:** A welcoming landing page featuring the school logo, tagline, and quick links to main sections. Include a prominent hero area (image or slider), highlights (news, announcements, or featured programs), and clear call-to-action buttons. Possibly showcase upcoming events or a welcome video. Navigation must be prominent and intuitive.
- **About Us:** A section with three sub-sections:
- **Our History:** Brief narrative of the school's founding, evolution, and milestones.
- **Mission & Vision:** Clear statements of educational goals and values.
- **Our Trust:** Information about the management/trustee body overseeing the school. Include relevant images (e.g. historic photo, trust logo).
- **Academics:** Overview of academic programs, curriculum highlights, streams of study, and achievements. This could include links to curriculum guides or department pages.
- **Campuses:** Two distinct campus pages or a combined page distinguishing each campus. Provide addresses (with embedded maps), photos, and unique features of each location. Visually differentiate them (e.g. use different accent colors or icons per campus) and include Google Maps embeds for each.
- **Admissions:** Information on enrollment procedures, eligibility, and key dates. Provide links to download admission forms (PDF/Word) for each relevant class or campus. Ensure forms are clearly labeled and accessible.
- **Community:** Section for parent-teacher association, alumni network, or community outreach programs. Could include news, announcements, or resources for stakeholders.

- **Events:** Interactive event calendar (e.g. embedded JavaScript calendar) showing upcoming school events, holidays, and important dates. Events data can be stored in a static JSON or iCal feed, rendered client-side.
- **Gallery:** A photo gallery showcasing school life, events, and facilities. Images should be high-quality and organized by category (e.g. Academics, Sports, Cultural).

- **Contact:** School's contact information and addresses. Include an embedded Google Map for each campus. No contact form is used; instead list email addresses, phone numbers, and mailing addresses for relevant offices.

- **Navigation Bar:** A responsive top navbar with the school logo and menu links (Home, About, Academics, etc.). On mobile, it should collapse into a hamburger menu. Support multi-level dropdowns if needed (for subsections like those under About Us).

- **Footer:** Persistent footer on all pages. Include school address(es), contact info, social media icons/links, copyright notice, and quick links (e.g. privacy policy).
- **Mobile/Responsive UI:** Ensure all layouts are mobile-friendly. Breakpoints and components should adapt gracefully (use Bootstrap's grid and utility classes). Menu buttons and taps should be large enough for touch.
- **Other Standard Features:** Include a "skip-to-content" link for accessibility, and breadcrumb navigation if the page hierarchy is deep. An optional search bar could be added for site search functionality.

Technical Specifications

- **Static Site:** The site must be fully static (no server-side rendering). All pages and assets will be hosted on GitHub Pages [4].
- **Bootstrap 5:** Use Bootstrap 5's grid and component framework for rapid, responsive UI development [5].
- **Semantic HTML5:** Mark up content with semantic tags (header, nav, main, section, article, footer, etc.) to improve accessibility and SEO.
- **Modular CSS (BEM):** Follow the BEM (Block Element Modifier) naming convention to create reusable, modular styles. BEM is a methodology that helps build reusable components and maintainable code [6] [7].
- **Accessibility (WCAG 2.1 AA):** All pages must meet WCAG 2.1 Level AA standards (e.g. color contrast, keyboard navigation, ARIA labels, alt text). WCAG are the internationally recognized accessibility standards [1].
- **Images & Media:** Use appropriately optimized images. Include descriptive alt text for images and ensure any embedded video has captions or transcripts.
- **Performance:** Leverage Bootstrap's utility classes to minimize custom CSS. Lazy-load non-critical images, and minify CSS/JS. Keep external dependencies (fonts, libraries) to a minimum for fast load times.
- **Efficient JS:** Use lightweight JavaScript for interactivity. If a calendar library is used, it should be fully client-side (no backend). Avoid heavy frameworks; vanilla JS or small libraries are acceptable.
- **Deployment:** The site will be deployed via GitHub Pages. Developers will push changes to the GitHub repository, and the site will publish automatically [4].

AI Development Process Requirements

- **Requirement 1.1 (Incremental Design Methodology):** Adopt an iterative approach. Break the redesign into incremental phases (e.g. layout and navigation first, then Home page, then remaining pages). Each iteration should deliver a working subset of features.
- **Requirement 1.2 (Task Prioritization):** Prioritize tasks by importance and dependencies. Core layout and navigation are highest priority, followed by content pages and secondary features.
- **Requirement 1.3 (Git Version Control):** Use Git for all development. Every change or addition must be committed with a clear message.
- **Requirement 1.4 (Frequent Commits):** Commit and push changes frequently (e.g. after completing each task or daily) to document progress and enable continuous review.
- **Requirement 1.5 (Pre-redesign Backup):** Before starting the redesign, create a backup branch or tag of the existing site codebase to preserve the current state.

AI Agent Prompting and Management

- **Single-Goal Prompts:** When using GitHub Copilot (Agent mode), ensure each prompt targets one specific task or file to avoid confusion. For example, "Generate HTML for the Home page hero section with Bootstrap 5 structure."
- **File-Specific Context:** Include relevant code snippets or filenames in prompts to provide context. Reference the exact file in question (e.g. "in `index.html`").
- **Custom Instructions:** Maintain a `copilot-instructions.md` file documenting project-specific guidelines or coding conventions. Use these custom instructions to guide the AI's output.
- **Recovery from Context Loss:** Periodically verify Copilot's context. If it diverges or loses focus, reset context by clarifying the next single goal or reloading the relevant file.
- **Scoped, Repeatable Prompts:** Keep prompts consistent and narrowly scoped so results are repeatable. Avoid vague or multi-part queries that could cause drift.

Review and QA Requirements

- **Code Reviews:** After each AI-assisted task or major change, perform a manual code review. Check the generated code for correctness, cleanliness, and adherence to standards before merging.
- **Visual Testing:** Verify the design by viewing pages in browsers. Ensure layout, typography, and images appear correctly on various screen sizes.
- **Functional Testing:** Test interactivity (e.g. navigation links, calendar component) to confirm everything works. Ensure downloadable forms open and display correctly.
- **Accessibility Checks:** Use accessibility tools or manual checks to confirm WCAG compliance (e.g. check color contrast, keyboard navigation, ARIA labels).
- **Developer Oversight:** A developer should oversee all AI outputs. Manual intervention is expected to fix or improve any AI-generated code as needed.

Risk Management Protocols

- **Git Rollback Options:** Maintain the ability to revert changes via Git. Use branches or tags to isolate work. If an AI-generated change causes issues, revert or checkout the previous commit.
- **Agent Restart Procedures:** If Copilot becomes unresponsive or produces off-track content, restart the agent or reload its session. Keep the environment and context clear for a fresh start.

- **Limitations Acknowledgment:** Recognize that AI outputs may be imperfect. Have fallback strategies: consult documentation or proceed with manual coding if the agent repeatedly errs or stalls.
- **Backup of Critical Assets:** Regularly push to the remote repository so work is not lost. Ensure assets (images, forms, etc.) are backed up in Git or other storage.

Definition of Done

Each feature or component is considered done only when: 1. The code has been peer-reviewed and merged. 2. It has been tested (visual and functional) on multiple devices/browsers. 3. It meets all stated requirements (design, accessibility, performance). 4. All changes are committed and documented in Git with descriptive messages.

---

1 WCAG 2.1 AA Guide for Beginners or Experts (Plain English) | Accessible.org
https://accessible.org/wcag-2-1-aa-guide-for-beginners-or-experts-plain-english/

2 Geometric Patterns in Islamic Art - The Metropolitan Museum of Art
https://www.metmuseum.org/essays/geometric-patterns-in-islamic-art

3 The Symbolic Use of Color in Islamic Architecture | ArchDaily
https://www.archdaily.com/1004972/the-symbolic-use-of-color-in-islamic-architecture

4 GitHub Pages | Websites for you and your projects, hosted directly from your GitHub repository. Just edit, push, and your changes are live.
https://pages.github.com/

5 Understanding Bootstrap 5 Layout
https://designmodo.com/bootstrap-5-layout/

6 7 BEM — Block Element Modifier
https://getbem.com/