ChatGPT

Master Plan for the Vihara Avalokitesvara Pondok Cabe Web-Site Project

1 Introduction and Context

Vihara Avalokitesvara Pondok Cabe is a non-profit Buddhist temple in Indonesia that offers spiritual services, community activities and youth programs through its affiliated organizations (Persatuan Muda-Mudi Vihara — PMV, and Gelanggang Anak Buddhis Indonesia — GABI). The project aims to develop a comprehensive website that (1) informs the public about the temple, its location and programs; (2) fosters community engagement; (3) integrates information about PMV and GABI; and (4) embodies financial transparency. Because the site will be publicly available and handle sensitive financial data, the master plan must consider legal obligations, security, performance, accessibility and environmental sustainability. This document outlines the plan from requirements gathering to deployment and maintenance and assesses the suitability of JavaScript as the primary technology.

2 Project Goals

1. **Provide comprehensive information**: Publish clear descriptions of the Vihara's location, routine activities (puja, meditation, Dharma lectures), calendars of upcoming events, biographies of the board and staff, lists of sponsors, and contact details.
2. **Integrate PMV and GABI profiles**: Highlight the youth programs (PMV) and children's programs (GABI) with dedicated pages covering mission, training activities, social service initiatives, photo galleries and ways to get involved.
3. **Ensure financial transparency**: Make annual financial reports and budgets publicly accessible. Donorbox's guide stresses that making financial reports available on a website builds trust [1]. Annual or impact reports should detail the organization's impact, finances, board and staff members, programs and activities, and include photographs [2]. Provide downloadable PDF reports and interactive charts illustrating revenue, expenditures and sponsor contributions.
4. **Support accountability and governance**: Publish a list of board members and key staff to be transparent about who is making decisions [3]. Adopt and publish policies on conflicts of interest and privacy.
5. **Offer a user-friendly experience**: Ensure the site is responsive, accessible (WCAG-compliant) and localized (at least Indonesian and English), with intuitive navigation, search capability and accessible contact forms.
6. **Achieve sustainability**: Apply Web Sustainability Guidelines (WSGs) to minimize the site's environmental impact. The digital industry accounts for 3–5 % of global greenhouse-gas emissions [4], prompting W3C to publish sustainability guidelines [5]. Key goals include energy-efficient hosting, code and image optimization, accessibility, performance optimization, resource management and carbon reduction [6].
7. **Maintain high security**: Adhere to JavaScript security best practices. The New Stack article notes that common vulnerabilities include cross-site scripting (XSS), man-in-the-middle (MitM), denial-of-service (DoS), cross-site request forgery (CSRF) and session hijacking [7]. API endpoints should use HTTPS, access-control lists and authentication [8]; content security policies (CSPs) restrict what the browser may load [9]; user input should be sanitized and encoded to prevent XSS [10] [11]; and regular security audits should validate dependencies, implement security

headers (HSTS, X-Content-Type-Options, Permissions-Policy, Referrer-Policy) and centralize key functions [12].

3 Stakeholders

Stakeholder	Role/Interest
Temple board and management	Provide requirements and approve content; ensure compliance with religious ethos and financial transparency.
PMV leadership	Describe youth programs, training activities and event schedules; supply content and images.
GABI coordinators	Provide information on Sunday school, training, and annual sarasehan events; share educational materials.
Sponsors and donors	Expect transparent financial reporting and clear recognition for support.
Dev team (project manager, UX designer, front-end and back-end developers, content creators)	Design, develop and maintain the website; implement security and sustainability features.
Visitors (congregants, community members, prospective donors)	Want accessible information about services, events, programs, and financial stewardship.
Regulatory bodies	Ensure compliance with laws on nonprofit reporting, data protection and accessibility.

4 Requirements and Features

4.1 Informational Pages

1. **Home page**: Summary of the temple's mission, welcome message and navigation to key sections.
2. **About Vihara**: History, vision, mission, location map, operating hours, contact details and directions.
3. **Routine activities**: Weekly puja schedule, meditation sessions, Dharma classes and volunteer opportunities.
4. **Event calendar**: An interactive calendar listing upcoming events, ceremonies and festivals with registration options. Use server-side logic to allow administrators to add or update events.
5. **PMV page**: Mission statement, leadership profile, training programs, social-service projects, membership information, event photos and contact details.
6. **GABI page**: Description of Sunday school activities, educational resources, annual sarasehan programs, photos, registration forms and parent guidance.
7. **Sponsor and donor recognition**: List of sponsors with logos and descriptions. Include donation acknowledgements and a call-to-action for new sponsors.
8. **Management and governance**: Board structure, staff roles and bios; include contact email addresses for accountability [3].
9. **Financial reports**: Downloadable annual financial statements, budgets and impact reports. Provide summary charts and the ability to filter by year. According to Donorbox's guide, making

financial reports available on the website demonstrates transparency [1] and increases donor trust [2].

10. **Policies**: Publish a conflict-of-interest policy, privacy policy and other governance documents to reassure donors of ethical operations [13].
11. **Contact and feedback**: Provide a contact form (with spam protection) and links to official social-media channels. Include directions via Google Maps, WhatsApp numbers and an FAQ page.

4.2 Functional Features

1. **Content Management System (CMS)**: Choose a headless CMS (e.g., Strapi, Keystone) or build a lightweight custom CMS in Node.js to allow authorized staff to update content without coding. Access must be restricted using role-based authentication.
2. **Multilingual support**: Implement language toggle (Indonesian/English). Use translation files and design pages to accommodate text expansion.
3. **Search**: Provide a site-wide search that indexes pages, events and reports.
4. **Accessibility**: Follow WCAG 2.1 guidelines. The WSG emphasises inclusive design, alt text, keyboard navigation and compliance with WCAG [14].
5. **Responsive design**: Use responsive CSS frameworks (e.g., Tailwind CSS or Bootstrap) to ensure usability on mobile devices.
6. **Interactive calendars and forms**: Use JavaScript components for event calendar and registration forms, but ensure that functionality degrades gracefully when JavaScript is disabled.
7. **Analytics**: Integrate privacy-respecting analytics (e.g., Matomo) to monitor traffic without compromising user privacy.
8. **Server-side rendering and caching**: Use frameworks like Next.js or SvelteKit to enable server-side rendering (SSR) for performance and SEO. Implement caching strategies (e.g., static site generation for content that rarely changes) to reduce server load.
9. **Donation integration**: Provide secure donation links or embed third-party donation forms (e.g., using Donorbox) that comply with local laws. Ensure receipts are issued as recommended [15].

5 Technology Evaluation

5.1 JavaScript and Node.js

Advantages:

- **Unified language**: Using JavaScript both on the client and server (with Node.js) simplifies the technology stack and reduces context switching for developers.
- **Rich ecosystem**: The Node.js ecosystem offers numerous libraries and frameworks (Express, Next.js) that expedite development and provide built-in features (routing, API integration, SSR).
- **Community and long-term support**: Node.js is widely adopted and maintained by a large open-source community, making it a sustainable choice.
- **Integration with headless CMS and front-end frameworks**: Many modern headless CMSs are built in JavaScript; frameworks like React/Next.js or Vue/Nuxt provide excellent support for accessibility and internationalization.

Concerns:

- **Security vulnerabilities**: JavaScript applications are susceptible to XSS, CSRF, session hijacking, API misuse and other attacks [7]. Developers must implement protective measures—HTTPS,

ACLs, authentication [8], CSPs [9], input sanitization and encoding [10] [11], and regular security audits [12].

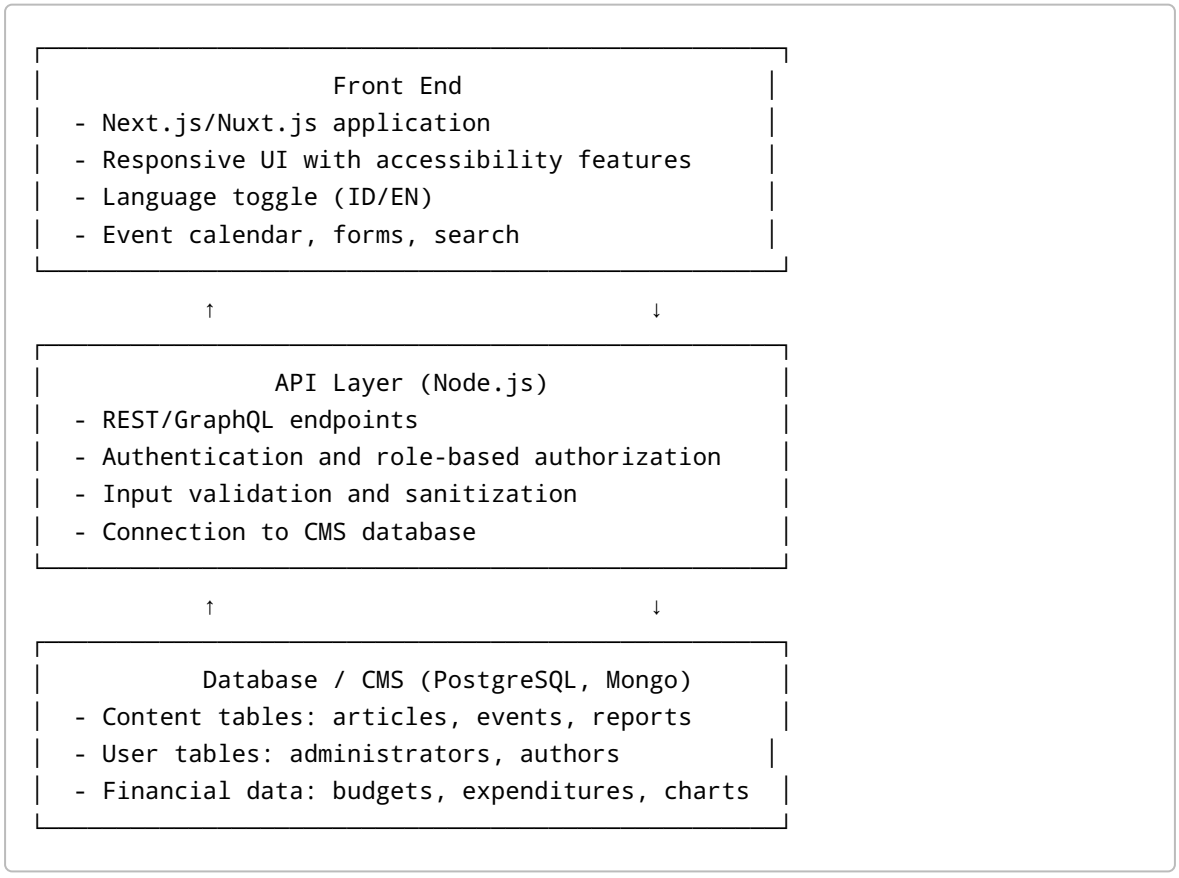
- **Performance overhead**: Client-heavy JavaScript can slow down load times, increasing energy consumption. To align with WSG goals, scripts should be minified and loaded lazily [16], and heavy logic should be executed server-side.
- **Maintainability**: The temple may rely on volunteers with varying technical skill. Excessive reliance on complex frameworks may hinder future maintenance.

5.2 Alternative Technologies

1. **Static Site Generators (SSGs)** (e.g., Hugo, Jekyll, Eleventy): Provide high performance and lower security risks due to pre-rendered pages. However, dynamic features (events calendar, search, multilingual content) become harder to implement, requiring client-side JavaScript or third-party services.
2. **PHP-based CMS (e.g., WordPress)**: Familiar to non-technical users and offers plug-ins for donations and events. But typical implementations are heavier, require regular patching, and can become bloated, violating sustainability and performance goals.
3. **Python frameworks (Django, Flask)**: Provide robust security and scalability; but the temple's volunteers may be more familiar with JavaScript.

Given the benefits and the need for dynamic features, the plan recommends a **JavaScript-centric stack** using **Node.js** on the back end with **Next.js** or **Nuxt.js** on the front end. However, developers must rigorously apply security and performance best practices.

6 Site Architecture and Information Flow



7 Security Strategy

1. **HTTPS and certificates**: Enforce HTTPS across the site and API to prevent eavesdropping and tampering.
2. **Access control and authentication**: Restrict CMS to authorized users with strong passwords and multi-factor authentication. Use role-based access control and implement ACLs [17].
3. **Input validation and sanitization**: Validate all user input (forms, API data) and escape special characters to prevent XSS and injection attacks [18]. Use server-side validation in addition to client-side checks.
4. **Content Security Policy (CSP)**: Define CSP headers to restrict the domains from which scripts, styles and media can be loaded [9].
5. **CSRF protection**: Use anti-CSRF tokens and HTTP-only cookies [11].
6. **Secure session management**: Use HTTP-only, secure cookies; regenerate session IDs after login; implement proper logout.
7. **Regular security audits**: Periodically check for outdated dependencies, implement security headers, centralize sensitive functions and use tools like Snyk and OWASP ZAP to identify vulnerabilities [12].
8. **Backup and disaster recovery**: Regularly back up databases and reports, store backups offsite, and establish a recovery plan.

8 Sustainability and Performance Plan

The WSG emphasises reducing environmental impact while improving user experience [19]. Key measures:

1. **Energy-efficient hosting**: Choose data centres powered by renewable energy and supporting energy-efficient servers [20]. Use CDN providers with green credentials.
2. **Code and image optimisation**: Write clean, efficient JavaScript and CSS; compress images; use modern formats (WebP); and remove unused libraries [21].
3. **Caching and CDNs**: Implement browser caching and CDN distribution to reduce latency and energy use [22] [23].
4. **Minification and lazy loading**: Minify CSS and JS files and use lazy loading to defer non-critical resources [24].
5. **Resource management**: Reduce HTTP requests; minimize data transfer; avoid bloatware and unnecessary plug-ins [25].
6. **Green hosting and carbon offsets**: Select hosting providers that use renewable energy and consider carbon offset programs [26]. Highlight this commitment on the website to demonstrate social responsibility.
7. **Accessibility and inclusive design**: Ensure that the site is accessible, including alt text, keyboard navigation and WCAG compliance [14]. Sustainable design goes hand in hand with accessibility by preventing resource waste due to user frustration.

9 Development Phases and Timeline (approx. 6 months)

Phase	Duration	Key Tasks	Deliverables
1 Initiation & Planning	2 weeks	Define scope and goals; identify stakeholders; gather requirements from temple board, PMV, GABI; assess content inventory; determine budget and resources.	Requirements document; project charter; risk register.
2 Architecture & Design	4 weeks	Design site map; information architecture; wireframes and mockups; choose technology stack; establish security and sustainability guidelines; plan database schema; define APIs.	Site architecture document; design prototypes; technology decision log.
3 Development	10 weeks	Set up repository and CI/CD; implement back-end (API layer, database); integrate headless CMS; build front-end components (pages, calendar, forms, search); implement multilingual support; integrate security measures (authentication, CSP, sanitization); implement performance optimizations (lazy loading, caching).	Minimum viable product (MVP) with core features; user authentication; CMS configured.
4 Content Ingestion & Testing	4 weeks	Migrate existing content (history, programs, sponsors); prepare financial reports and annual reports for upload; run unit and integration tests; test accessibility (WCAG guidelines); perform security audits [12]; load testing and performance profiling; incorporate sustainability optimizations [6].	Fully populated staging site; test reports; bug list and fixes.
5 UAT & Deployment	3 weeks	Conduct user acceptance testing with temple board, PMV, GABI; refine based on feedback; deploy to production server with HTTPS and CDN; monitor initial usage; fix any issues.	Live website; training manuals; deployment checklist.
6 Training & Handover	1 week	Train administrators on CMS and analytics; document update procedures; provide security and privacy training; hand over system documentation and credentials.	User manuals; technical documentation; training sessions.
7 Maintenance & Continuous Improvement	Ongoing	Monitor site performance and security; update dependencies and frameworks; publish new reports; incorporate feedback; expand features (e.g., donation integration, mobile apps).	Quarterly updates; security audit reports; sustainability impact metrics.

10 Resource and Budget Considerations

- **Human resources**: Project manager (part-time), UX/UI designer, two front-end developers, one back-end developer, one content writer/editor (preferably bilingual), one translator, and a part-time finance liaison to provide accurate financial data.
- **Infrastructure**: Domain registration; green hosting plan; CDN subscription; SSL certificates; headless CMS (open-source or subscription); analytics tool (self-hosted or third-party); offsite backup storage.
- **Training and documentation**: Budget for training sessions and creation of documentation (user manuals, coding standards, security guidelines).

11 Risk Analysis and Mitigation

Risk	Impact	Likelihood	Mitigation
Content delays	High; delays the launch.	Medium	Start content gathering early; assign dedicated content managers in the temple, PMV and GABI; set clear deadlines; allow placeholder content for initial testing.
Security breaches	High; could compromise personal and financial data.	Medium	Implement rigorous security measures (HTTPS, input sanitization, CSP, audits) [27]; keep dependencies updated; schedule regular penetration tests.
Technical complexity	Medium; volunteers may struggle to maintain.	Medium	Use well-documented frameworks; provide comprehensive training; document processes; consider contracting periodic maintenance.
Regulatory changes	Medium; may require additional reporting or privacy measures.	Low	Monitor relevant regulations; design architecture to be adaptable; maintain contact with legal advisors.
Lack of engagement	Low; features may be underused.	Medium	Promote the site through social media; involve community in content creation; gather feedback to improve usability.
Environmental goals unmet	Medium; site may not meet sustainability targets.	Low	Monitor carbon footprint using tools (e.g., the Green Web Foundation's checker); adjust hosting and code optimization accordingly.

12 Recommendations and Concluding Remarks

1. **Adopt transparent financial reporting**: Make financial statements, budgets and impact reports downloadable and easy to understand. Donorbox notes that making financial reports available builds donor trust [1] and annual reports should highlight impact, finances, board and programs [2].

2. **Publish governance information**: List board members and key staff with bios and contact information to enhance accountability [3].
3. **Apply security best practices**: Use HTTPS, ACLs, authentication, CSP, input sanitization, anti-CSRF tokens, secure cookies and regular security audits [28]. Provide training for staff on safe content publishing.
4. **Design for sustainability and accessibility**: Follow the WSG's energy efficiency, performance optimization, resource management, and carbon footprint guidelines [6]. Use green hosting, optimize code and images, minimize HTTP requests and data transfer, adopt lazy loading and caching, and ensure the site is accessible to users with disabilities [14].
5. **Evaluate the JavaScript stack periodically**: While JavaScript provides a unified stack, it also introduces security and performance concerns. The temple should periodically assess whether alternative technologies (e.g., static site generators or modern frameworks like Astro) could meet needs with lower complexity.
6. **Engage the community**: Encourage PMV and GABI members to contribute content (blogs, photos, event recaps). Integrate feedback loops and analytics to improve the site and maintain relevance.
7. **Plan for scalability**: The architecture should support future expansion such as online courses, live-streamed ceremonies, or membership portals.

By following this master plan, Vihara Avalokitesvara Pondok Cabe will gain a modern, secure and sustainable website that fosters community engagement, showcases its youth and children's programs, and builds trust through transparent governance and financial reporting.

---

[1] [2] [3] [13] [15] How to Improve Nonprofit Transparency | A Complete Guide
https://donorbox.org/nonprofit-blog/nonprofit-transparency

[4] [5] [6] [14] [16] [19] [20] [21] [22] [23] [24] [25] [26] Web sustainability guidelines | Blog | Extramile Digital
https://www.extramiledigital.com/web-sustainability-guidelines-what-they-mean-for-your-b2b-website/

[7] [8] [9] [10] [11] [12] [17] [18] [27] [28] 5 JavaScript Security Best Practices for 2024 - The New Stack
https://thenewstack.io/5-javascript-security-best-practices-for-2024/