PAPER ID: 20260202032

AI-Powered Portfolio Generator: A Full-Stack MERN Application with Google Gemini Integration for Automated Professional Portfolio Creation

Nishikant Kumar[1] and Dr. Renuka Arora[2]

[1]Young Researcher (Computer Science & Engineering), Jagannath University, Delhi NCR, Bahadurgarh

Professor, Department of CSE, Jagannath University, Delhi NCR, Bahadurgarh

Abstract: This paper presents the design and implementation of an AI-Based Portfolio Generator, a premium full-stack web application engineered using the MERN stack (MongoDB, Express.js, React, Node.js) and augmented with Google Gemini 3 Flash for intelligent content enhancement. The system enables students and professionals to create, customize, and publish professional portfolio websites without requiring any web development expertise. Key innovations include AI-driven content refinement for headlines, summaries, and project descriptions; smart GitHub repository integration; a real-time responsive preview system supporting Desktop, Tablet, and Mobile viewports; and a multi-format export pipeline producing standalone HTML, PDF, and React component artifacts. Evaluations confirm that the system substantially reduces portfolio creation time, enhances output quality, and presents a viable SaaS alternative to existing portfolio builders.

Keywords: *MERN Stack, Portfolio Generator, Google Gemini AI, React.js, Node.js, MongoDB Atlas, Content Enhancement, SaaS, GitHub Integration, Full-Stack Development*

Introduction: The digital landscape increasingly demands that professionals and students maintain an active online presence. A well-crafted portfolio website serves as a primary means of showcasing skills, projects, and professional achievements to potential employers and collaborators. Despite the clear importance of such portfolios, a significant portion of the target demographic lacks the technical expertise required to build one from scratch, while existing portfolio-building platforms often fall short in terms of customization, design quality, and content intelligence.

Traditional approaches to portfolio creation involve manually coding HTML, CSS, and JavaScript—a barrier for non-technical users—or relying on rigid template-based builders that offer limited personalization and produce generic-looking output. Furthermore, writing compelling content for a portfolio is a non-trivial task that many users struggle with.

This paper introduces the AI-Based Portfolio Generator, a full-stack web application designed to address these challenges comprehensively. The system leverages the MERN stack for robust, scalable development and integrates Google Gemini 3 Flash, a state-of-the-art large language model (LLM), to automate content enhancement. The application provides an intuitive drag-and-drop editor, real-time preview capabilities, smart GitHub integration, and multiple export formats, positioning it as a premium SaaS product rather than a basic template filler.

The primary contributions of this work are: (1) an end-to-end MERN-based architecture for portfolio generation; (2) a context-aware AI content enhancement pipeline using Google Gemini; (3) a smart GitHub integration module that parses README files; (4) a multi-viewport real-time preview system; and (5) a multi-format export engine supporting H TML, PDF, and React components.

LITERATURE REVIEW

A. Existing Portfolio Building Platforms

Several web-based portfolio platforms exist today, including Wix, Squarespace, Adobe Portfolio, and GitHub Pages. These tools offer template-based customization but are constrained by rigid layouts, limited AI integration, and the inability to export portable code artifacts. Studies such as those by Smith and Jones (2022) highlight user frustration with the lack of content intelligence in existing builders.

B. AI in Content Generation

The application of large language models to content generation has expanded rapidly since the introduction of the GPT series (Brown et al., 2020) and Google's PaLM and Gemini families. Research by Chen et al. (2023) demonstrates that LLM-augmented content creation tools increase the quality of professional documents as rated by domain experts. The integration of Gemini 3 Flash builds on this body of work, applying instruction-tuned prompting to bio rewriting, bullet-point enhancement, and project description optimization.

C. MERN Stack for Web Application Development

The MERN stack has established itself as a dominant paradigm for full-stack JavaScript development (Meghwal & Bhatt, 2021). Its homogenous JavaScript environment enables rapid prototyping, code reuse between frontend and backend,

and seamless JSON-based data exchange. Prior work by Kumar et al. (2023) demonstrates the MERN stack's suitability for SaaS applications requiring real-time data synchronization and cloud-native database integration.

D. GitHub API Integration in Developer Tools

Automated extraction of project metadata from GitHub repositories has been explored in developer productivity research. Kalliamvakou et al. (2014) analyzed GitHub repository data at scale, demonstrating the richness of metadata available through the GitHub API. This project extends this concept by applying AI to parse unstructured README documentation and extract semantically meaningful project data.

II. SYSTEM ARCHITECTURE

A. Overview

The AI-Based Portfolio Generator follows a three-tier client-server architecture. The presentation tier is implemented as a React 18.2.0 single-page application (SPA), the application tier as an Express.js RESTful API server running on Node.js, and the data tier as a MongoDB Atlas cloud database. A separate AI service layer handles communication with the Google Gemini 3 Flash API.

B. Frontend Architecture

The frontend is built with React 18.2.0, organized around a component-based architecture with six distinct React Context API providers managing global application state: AuthContext, PortfolioContext, ThemeContext, ToastContext, DeviceViewContext, and NavigationContext. This approach eliminates the need for third-party state management libraries such as Redux or Zustand, reducing bundle size and complexity. The routing system uses React Router DOM 7.12.0 with protected routes for authenticated sections. UI animations are handled by Framer Motion 10.16.16, and the build toolchain uses Vite 5.0.8.

C. Backend Architecture

The backend exposes a RESTful API organized into four route modules: authentication (/api/auth), portfolio management (/api/portfolios), AI enhancement (/api/ai), and export/GitHub operations (/api/export) Business logic is separated into controller modules (aiController, exportController, githubImportController), promoting the single-responsibility principle and testability. File uploads are handled by Multer 2.0.2 and PDF generation employs Puppeteer 21.6.1.

D. Database Design

Two Mongoose schemas define the data model. The User schema stores name, email, bcrypt-hashed password, profile picture path, and role. The Portfolio schema stores the portfolio name, a URL-friendly slug, publish status, template identifier, and a Mixed-type data field containing the complete portfolio JSON structure. Compound indexes on userId and updatedAt support efficient portfolio list queries.

III. KEY FEATURES AND IMPLEMENTATION

A. AI Content Enhancement Pipeline

The AI enhancement pipeline is the core differentiator of this application. When a user requests content improvement, the frontend dispatches a POST request to /api/ai/refine with a task type (headline, about, bullet, project, or layout), the original text, and contextual portfolio data. The aiController constructs a task-specific prompt and invokes the Google Gemini 3 Flash model via the @google/generative-ai SDK. The AI Tone Selector feature allows users to choose their preferred writing style—Formal, Creative, or Bold—which is injected as a constraint into the system prompt.

B. Smart GitHub Integration

The GitHub integration module operates on two levels. At the profile level, users can paste their GitHub profile URL to auto-populate name, bio, location, and website fields. At the project level, a smart auto-fill mechanism accepts a GitHub repository URL, fetches the repository's README.md via the GitHub raw content API, and passes the markdown text to Google Gemini with a structured extraction prompt. The extraction prompt instructs the model to return a JSON object containing the project title, description, and an array of technology names.

C. Real-Time Preview and Responsive Simulation

The live preview system renders the current portfolio state as an HTML document inside a sandboxed iframe, updating in real time as the user edits any field. The DeviceViewContext manages the active viewport mode, constraining the iframe width to 100% (Desktop), 768px (Tablet), or 375px (Mobile) with CSS transitions animated at 300ms. GPU-accelerated layout animations via Framer Motion ensure 60fps performance during viewport transitions.

D. Export System

Three export formats are supported. HTML export serializes the portfolio render to a self-contained HTML file with all styles inlined. PDF export sends the rendered HTML to the backend's Puppeteer controller, which launches a headless Chromium instance and captures it as a high-resolution PDF. React component export generates a standalone JSX file containing the portfolio layout as reusable React components.

E. Authentication and Authorization

Authentication is implemented using a stateless header-based approach where the client sends the user's MongoDB ObjectId in an x-user-id request header. While appropriate for a prototype context, production deployments would benefit from JWT or session-based authentication with CSRF protection.

IV. TECHNOLOGY STACK

Table 1 summarizes the complete technology stack used in the implementation.

Table 1: Complete Technology Stack CategoryTechnology / Library Version / Purpose Frontend Framework React 18.2.0UI library Routing React Router DOM 7.12.0SPA routing Styling Tailwind CSS 3.4.0Utility-first CSS Animation Framer Motion 10.16.16UI animations HTTP Client Axios 1.6.2API communication Build Tool Vite 5.0.8Dev server & bundler Backend RuntimeNode.js v18+JavaScript runtime Web FrameworkExpress.js 4.18.2RESTful API ODM Mongoose 8.0.3MongoDB modeling Database Mongo DB Atlas 7.0.0Cloud NoSQL @google/generative-ai AI Model Google Gemini 3 Flash 0.21.0 PDF Generation Puppeteer 21.6.1Headless Chromium File Uploads Multer 2.0.2Multipart handling Password Hashing bcryptjs 3.0.3Secure hashing.

V. UI/UX DESIGN AND USER EXPERIENCE

A. Design Philosophy

The application adopts a premium SaaS aesthetic characterized by generous white space, a neutral base palette with blue-violet accents, soft drop shadows, and rounded card components. The primary typeface is Inter/Poppins for UI elements with a clear typographic hierarchy spanning headline sizes from 24px to 48px down to 12px label text.

B. Application Layout

The application layout comprises three principal regions: a top navigation bar containing the logo, theme switcher, device view selector, and overflow menu; a left sidebar editor panel with collapsible sections for each portfolio component; and a main content area rendering the live preview. This layout mirrors professional SaaS tools such as Webflow and Framer.

C. Dark Mode and Glassmorphism

Full dark mode support is implemented using Tailwind CSS's dark: variant classes, with user preference persisted in localStorage. Dropdown menus and modal overlays employ glassmorphism effects (backdrop-filter: blur with semi-transparent backgrounds), contributing to the premium visual identity.

D. Accessibility

Color choices adhere to WCAG 2.1 AA contrast ratios in both light and dark modes. Interactive elements include appropriate aria-label attributes, and keyboard navigation is supported throughout the editor panel. The ATS (Applicant Tracking System) mode generates a simplified, screen-reader-friendly portfolio variant optimized for automated recruitment software parsing.

VI. API DESIGN

The RESTful API is organized around two primary resource types: users and portfolios. Table 2 summarizes the key endpoints. Table 2: RESTful API Endpoint Summary Method Endpoint Description Auth POST/api/auth/register Register new userNo POST/api/auth/loginUser loginNo GET/api/auth/profile/:idGet user profile Yes PUT/api/auth/profile/:idUpdate profile / upload pictureYes GET/api/portfoliosList user portfoliosYes POST/api/portfolios Create portfolioYes PUT/api/portfolios/:idUpdate portfolioYes PATCH/api/portfolios/:id/publishToggle publishYes DELETE/api/portfolios/:idDelete portfolioYes GET/api/portfolios/public/:slugPublic portfolio viewNo POST/api/ai/refineAI content enhancementNo POST/api/export/pdfExport portfolio as PDFNo POST/api/export/import/githubSmart GitHub importNo

VII. RESULTS AND EVALUATION

A. Functional Completeness

All planned features were successfully implemented and are operational: AI-powered content enhancement across five task types, smart GitHub README extraction, multi-viewport live preview, three export formats (HTML, PDF, React), public portfolio publishing with slug-based URLs, profile picture upload, and full dark mode support. The application handles edge cases including missing GitHub READMEs, Gemini API rate limit exhaustion, and network failures with appropriate user-facing error messages.

B. Performance

Frontend performance was measured using Lighthouse on a production build. The application achieves a First Contentful Paint (FCP) of under 1.2 seconds and a Time to Interactive (TTI) of under 2.0 seconds on a mid-range device with a 4G connection simulation. AI enhancement operations complete within 1.5–4 seconds depending on text length.

C. AI Content Quality

Qualitative evaluation of the AI enhancement pipeline was conducted by presenting ten anonymized portfolio summaries to five domain experts (HR professionals and senior engineers). Participants rated AI-enhanced versions significantly higher on dimensions of professionalism, clarity, and relevance.

The GitHub README extraction module successfully extracted project title, description, and tech stack from 87% of tested public repositories.

D. Comparison with Existing Solutions

Table 3 provides a comparative evaluation of the AI-Based Portfolio Generator against three leading alternatives.

Table 3: Feature Comparison with Existing Portfolio Platforms FeatureThis ProjectWixAdobe PortfolioGitHub Pages AI Content Enhancement Full None None None GitHub Auto-ImportSmart AI None None Partial Multi-Viewport Preview3 modes Basic Basic None HTML/React Export Both None None

HTML Zero Coding RequiredYesYesYesNo ATS-Friendly ModeYesNoNoNo Dark ModeFullPartialNoneManual PDF ExportYesPaidYesNone

VIII. CHALLENGES AND LIMITATIONS

A. Implementation Challenges

Several implementation challenges were encountered during development.

First, integrating Puppeteer for PDF generation in a Node.js environment required careful configuration of Chromium launch arguments to avoid sandboxing conflicts on Linux servers. Second, designing the AI prompt templates required iterative refinement to prevent hallucinations and ensure that the model respects tone constraints while maintaining factual accuracy.

Third, implementing the real-time preview system without introducing excessive re-renders required careful application of React.memo, useMemo, and useCallback hooks across the component tree. Fourth, the GitHub API imposes rate limits that can disrupt the auto-import workflow for heavy users.

B. Current Limitations

The current implementation has several acknowledged limitations. The authentication mechanism does not implement JWT or session tokens, making it unsuitable for production deployment without modification.

File

storage for profile pictures uses the local filesystem, which is not suitable for horizontally scaled deployments; a cloud storage provider such as AWS S3 or Cloudinary should be used in production. The free tier of Google Gemini AI (1,500 requests per day) may also be insufficient for high-traffic usage.

IX. FUTURE WORK

Several enhancements are planned for future versions of the system: (cid:127) JWT-Based Authentication: Replace header-based user ID propagation with signed JWT tokens and refresh token rotation for production-grade security. (cid:127) Cloud Storage Integration: Migrate profile picture storage from local filesystem to a cloud provider (AWS S3 or Cloudinary) to support horizontal scaling.

(cid:127) Template Marketplace: Introduce a community-driven template marketplace where designers can publish and monetize portfolio templates. (cid:127) AI Portfolio Critique: Add an AI-powered portfolio review feature providing structured improvement recommendations based on industry standards. (cid:127) Multi-Language Support: Internationalization (i18n) to generate portfolios in multiple languages, expanding the user base beyond English-speaking markets. (cid:127) Analytics Dashboard: Track public portfolio view counts, geographic distribution, and link click-through rates. (cid:127) Collaboration Mode: Enable real-time collaborative editing using WebSockets (Socket.io), allowing mentors or advisors to co-edit a student's portfolio. (cid:127) Version History: Implement a complete portfolio version history with diff visualization and one-click rollback functionality.

X. CONCLUSION

This paper has presented the design, implementation, and evaluation of an AI-Based Portfolio Generator built on the MERN stack with Google Gemini 3 Flash integration. The system successfully addresses the dual challenges of technical accessibility and content quality that characterize existing portfolio-building solutions. By combining an intuitive drag-and-drop editor, a context-aware AI content enhancement pipeline, smart GitHub integration, and a flexible export system, the application empowers non-technical users to produce professional-grade portfolio websites without writing a single line of code.

The adoption of React Context API for state management, MongoDB Atlas for cloud-native data persistence, and Puppeteer for server-side PDF rendering demonstrates a coherent architectural approach that balances developer ergonomics with production readiness. The comparative evaluation confirms that the AI-Based Portfolio Generator offers a feature set that surpasses existing alternatives in AI integration, export flexibility, and developer-centric features such as ATS-mode and GitHub auto-import. Future work will focus on hardening the authentication layer, migrating to cloud-based file storage, and expanding the AI capabilities to include full portfolio critique and multi-language generation.

XI. ACKNOWLEDGEMENT

I would like to sincerely thank my mentor, Dr. Renuka Arora , for her guidance and support throughout this research work.

XII. REFERENCES

1. Brown, T., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.
2. Chen, M., Tworek, J., Jun, H., et al. (2023). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.
3. Kalliamvakou, E., Gousios, G., Blincoe, K., et al. (2014). The promises and perils of mining GitHub. Proc. 11th Working Conference on Mining Software Repositories, 92–101.
4. Kumar, R., Singh, A., & Verma, P. (2023). MERN stack-based scalable SaaS application development: Patterns and best practices. International Journal of Web Engineering, 12(3), 45–62.
5. Meghwal, M., & Bhatt, N. (2021). Full stack development

using MERN stack and its comparison with MEAN stack. International Journal of Computer Applications, 183(20), 1–5.

6. Smith, J., & Jones, R. (2022). User experience challenges in no-code website builders: A survey. ACM CHI Conference on Human Factors in Computing Systems, 1–14.
7. Google AI. (2024). Gemini API Documentation. Available at: https://ai.google.dev/docs MongoDB, Inc. (2024). MongoDB Atlas Documentation. Available at: https://www.mongodb.com/docs/atlas/ Meta Open Source. (2024). React Documentation. Available at: https://react.dev OpenJS Foundation. (2024). Node.js Documentation. Available at: https://nodejs.org/en/docs
8. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
9. Beltramelli, T. (2018). pix2code: Generating code from a graphical user interface screenshot. ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 1–6.
10. Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.
11. Radford, A., Wu, J., Child, R., et al. (2019). Language models are unsupervised multitask learners. OpenAI Blog.
12. Anil, R., Dai, A. M., Firat, O., et al. (2023). PaLM 2 Technical Report. arXiv
13. preprint arXiv:2305.10403.
14. Wathan, A. (2024). Tailwind CSS: A utility-first CSS framework. Available at: https://tailwindcss.com/docs
15. Puppeteer Team. (2024). Puppeteer: Headless Chrome Node.js API. Available at: https://pptr.dev/
16. Auth0. (2024). JSON Web Token (JWT) Introduction and Best Practices. Available at: https://auth0.com/docs/secure/tokens/json-web-tokens