Archives available at journals.mriindia.com

International Journal on Advanced Computer Engineering and Communication Technology

ISSN: 2278-5140

Volume 15 Issue 01, 2026

An Intelligent AI-Driven Platform for Automated Portfolio Generation and Real-Time Publishing

[1]Abhay Gaidhani, [2]Yashraj Thube, [3]Kalpesh Ghodekar, [4]Rishikesh Marathe, [5]Shreyash Agare

[1] Assistant Professor, Department of Computer Engineering, Sandip Institute of Technology and Research Centre (SITRC), Nashik, India

[2,3,4,5] B.E Students, Department of Computer Engineering, Sandip Institute of Technology and Research Centre (SITRC), Nashik, India

Email: abhay.gaidhani@sitrc.org, [2] yashraj07thube@gmail.com, [3]kalpeshghodekar2004@gmail.com, [4] rishikeshmarathe04@gmail.com, [5]shreyashagare437@gmail.com

Peer Review Information

Submission: 18 March 2026

Revision: 03 April 2026

Acceptance: 22 April 2026

Keywords

SaaS Platform, Portfolio Generation, Large Language Model, Schema-Driven Architecture, Template Rendering, React, Spring Boot, FastAPI, Ollama, Placeholder Mapping

Abstract

This paper presents Portiva, a cloud-based Software-as-a-Service (SaaS) platform designed to automate the generation, deployment, and management of professional portfolio websites. The system collects structured user data through a guided multi-step onboarding process and utilizes a locally hosted Large Language Model (LLM) via Ollama to generate high-quality, context-aware content. The generated portfolio is dynamically rendered and published on a unique public URL within minutes.

The primary contribution of this work is a schema-driven placeholder mapping architecture, where each template explicitly defines its data requirements using a structured placeholder_schema stored in the database. The AI engine interprets this schema to generate only the required content fields, eliminating hardcoded mappings and enabling seamless extensibility of templates without modifying application logic. The platform is implemented using a modern full-stack architecture comprising React 19, Spring Boot 3.2, FastAPI, PostgreSQL, and Redis. It incorporates production-grade features such as asynchronous AI processing, Redis-based caching, soft deletion, and subscription management via Razorpay. Experimental evaluation demonstrates sub-200ms API response times and complete end-to-end portfolio generation within three minutes on commodity hardware, highlighting the system's efficiency and scalability.

Introduction

The rapid expansion of the digital economy has significantly increased the importance of personal branding, particularly for professionals in software engineering, design, and other technology-driven domains. A well-structured portfolio website serves as a crucial medium for showcasing technical expertise, projects, and professional achievements. However, the process of designing, developing, and maintaining such portfolios remains complex, requiring expertise in web development, UI/UX design, hosting infrastructure, and content creation.

Traditional portfolio creation approaches rely on manual website builders or custom development workflows, which are often time-intensive and require significant effort. Consequently, many individuals, especially students and early-career professionals, either delay creating portfolios or

produce suboptimal representations of their skills.

Recent advancements in Artificial Intelligence (AI), particularly Large Language Models (LLMs), present an opportunity to automate and enhance portfolio generation. However, existing AI-based systems lack structured data integration, dynamic adaptability, and a formal contract between generated content and visual templates. To address these limitations, this paper introduces Portiva, a schema-driven SaaS platform that automates portfolio creation, content generation, and live deployment through an intelligent and scalable architecture.

1. AI-Driven Personalized Portfolio Generation

Recent developments in Machine Learning (ML) and Large Language Models (LLMs) have enabled the generation of high-quality, context-aware textual content based on structured inputs. In the context of portfolio systems, these technologies can transform user-provided data such as skills, experience, and projects into professionally written summaries and descriptions.

Unlike conventional systems that rely on manual content creation, Portiva employs a structured AI pipeline where only specific content fields are generated dynamically. By constraining the generation process through predefined schemas, the system ensures relevance, consistency, and contextual accuracy while minimizing unnecessary computation.

This approach significantly reduces user effort while improving the overall quality and coherence of generated portfolio content.

2. System Architecture and Data Integration

Portiva adopts a microservices-based architecture integrating multiple technologies to ensure scalability, modularity, and performance. The system consists of a React-based frontend for user interaction, a Spring Boot backend for orchestration and business logic, and a FastAPI-based AI engine for content generation.

Data is stored in PostgreSQL, while Redis is used for caching, asynchronous processing, and performance optimization.

User inputs collected through a multi-step onboarding form are stored as flexible JSON objects, enabling schema evolution without requiring database migrations. The backend aggregates user data and template schemas and forwards them to the AI engine, which generates structured content based on predefined requirements.

This architecture enables efficient handling of both structured and semi-structured data while ensuring seamless communication between system components.

3. Automated Template Rendering and Content Generation

Portiva automates the generation of complete, production-ready portfolio websites by integrating structured data processing with dynamic template rendering. The system collects detailed user information, including skills, experience, projects, education, and achievements, and maps this data to predefined templates.

A key innovation is the schema-driven placeholder mapping system, where each template explicitly defines the content it requires. The AI engine interprets this schema and generates only the necessary fields, ensuring precise alignment between content and presentation.

Additionally, the platform supports dynamic template switching without requiring users to re-enter data. This feature enhances flexibility and user experience, allowing seamless experimentation with different designs while preserving content consistency.

4. Research Challenges and Gaps

Despite significant progress in AI-driven website generation, several limitations remain in existing systems. These include the lack of structured integration between content generation and template rendering, over-reliance on unstructured prompts leading to inconsistent outputs, inefficient AI pipelines with high computational overhead, and limited adaptability and personalization.

Furthermore, concerns related to data privacy, system transparency, and user control over generated content remain critical challenges in AI-driven platforms.

Portiva addresses these challenges through a schema-driven architecture, a two-phase AI generation pipeline, and an asynchronous processing model. These design choices enable efficient, scalable, and reliable portfolio generation while maintaining high-quality output and system performance.

Literature Survey

Over the past decade, portfolio generation systems have evolved from manual coding approaches to automated and AI-assisted platforms. Traditional solutions rely heavily on static templates and manual content input, requiring significant technical expertise.

Recent AI-based systems have introduced automated content generation; however, they often lack structured data integration and fail to

establish a formal relationship between generated content and template requirements. Most systems rely on unstructured prompts, resulting in outputs that are difficult to modify or reuse.

This highlights the need for schema-driven architectures that combine structured data collection with controlled AI generation. Portiva addresses this gap by introducing a system that tightly integrates templates, schemas, and AI pipelines into a unified framework.

1. Generic Website Builders

Traditional website builders such as Wix, Squarespace, and WordPress provide drag-and-drop interfaces for creating portfolio websites. While these platforms offer flexibility and ease of use, they rely heavily on manual content entry and design decisions.

Users must write their own content, structure layouts, and manage updates independently. Additionally, these systems do not dynamically adapt to individual user profiles, making it difficult for users without design expertise to create professional-quality portfolios.

2. AI-Based Website Generation Platforms

AI-powered website builders such as Framer AI and Durable.co generate websites based on short textual prompts. While these platforms reduce the effort required for content creation, they lack structured data collection and fail to provide deep personalization.

Furthermore, there is no formal mapping between AI-generated content and template placeholders, making it difficult to modify or reuse generated content across different designs.

3. Role of AI and LLMs in Content Generation

Large Language Models (LLMs) have demonstrated strong capabilities in generating human-like text for various applications. In portfolio generation, LLMs can transform structured data into meaningful, professional content.

However, unstructured use of LLMs often leads to inefficiencies and inconsistent outputs. A structured approach, where only specific fields are generated based on predefined schemas, improves both efficiency and output quality.

4. Benefits of Personalization in Portfolio Systems

Personalized digital content significantly improves user engagement and perceived value. In portfolio systems, personalization ensures alignment between user identity, skills, and presentation.

Such systems reduce user effort while enhancing the overall effectiveness of portfolio communication.

5. Contribution of the Portiva System

Portiva introduces a novel approach that integrates structured data collection, schema-driven template design, and AI-based content generation within a unified framework.

It establishes a formal contract between templates and AI generation through a database-stored placeholder schema, enabling precise and efficient content generation.

Additionally, the two-phase generation pipeline improves performance by separating deterministic data mapping from AI-based content generation, reducing computational overhead.

System Design and Architecture

1. Frontend Design

The frontend of the Portiva system is developed using React 19, leveraging Vite as the build tool to achieve fast development and optimized production performance. Tailwind CSS is utilized to ensure a responsive, consistent, and visually appealing user interface across different devices and screen sizes.

The user interface is designed around a guided multi-step onboarding workflow, which systematically collects structured professional data, including personal details, technical skills, work experience, projects, and educational background. This step-by-step interaction model reduces cognitive load on users, ensures completeness of input data, and maintains consistency across all generated portfolios.

To enhance usability, the frontend also provides a centralized dashboard that enables users to manage their portfolios, preview generated outputs, switch between templates, and update their information dynamically. Real-time feedback and intuitive navigation further improve the overall user experience.

A key architectural decision is the implementation of a separate lightweight rendering layer for public portfolio pages. This layer operates independently of the main application state management (such as Redux), ensuring minimal load time and improved performance. By isolating the public rendering pipeline, the system achieves faster page delivery, reduced resource consumption, and enhanced scalability, particularly under high user traffic conditions.

2. Backend Framework

The backend of the Portiva system is implemented using Spring Boot 3.2, providing a

robust, scalable, and enterprise-grade framework for handling business logic, API communication, and system orchestration. The backend exposes a set of RESTful APIs that support core functionalities such as user authentication, portfolio creation and management, template handling, and subscription services.

Acting as the central orchestrator, the backend coordinates interactions between the frontend and the AI engine. Upon receiving user input, the backend performs validation, persists the data in the database, and initiates the AI generation process asynchronously. It also manages the lifecycle of portfolio generation by tracking job status and updating the frontend through periodic status checks.

To ensure high performance and scalability, the backend incorporates several advanced mechanisms:

- Asynchronous processing (@Async) to handle long-running AI tasks without blocking API responses
- Scheduled polling (@Scheduled) to retrieve results from the AI engine at defined intervals
- Redis-based caching to reduce database load and improve response times

This architecture allows the system to efficiently handle concurrent user requests while maintaining low latency and high throughput, making it suitable for production-scale deployment.

3. AI Generation Engine

The AI generation engine is implemented as an independent microservice using FastAPI and is powered by a locally hosted Large Language Model (LLM) through Ollama. This component is responsible for both template selection and intelligent content generation, forming the core of the system's automation capability.

The AI pipeline is structured into a two-phase processing model:

- **Phase 1: Direct Mapping (Deterministic Processing)**

In this phase, structured user data is directly mapped to predefined template placeholders without invoking the AI model. This includes deterministic fields such as user name, skills, project details, and work experience entries. Since this process does not involve any model inference, it is computationally efficient, highly reliable, and ensures accurate data representation.

- **Phase 2: AI-Based Content Generation**

In the second phase, only selected fields that require contextual or creative content are generated using the LLM. These fields are explicitly defined in the template's placeholder_schema and may include professional summaries, headlines, taglines, and descriptive content.

The template selection process is driven by a scoring function:

$$\text{Score}(T) = \Sigma\ (w_i \times f_i(\text{user\_data}))$$

where each template is evaluated based on weighted user attributes such as skill count, experience level, and number of projects.

By restricting AI generation to only essential fields, the system significantly reduces computational cost, minimizes latency, and improves overall efficiency. The final output is a structured placeholder_map, which serves as the input for frontend rendering.

4. Database Management

Portiva utilizes PostgreSQL as the primary relational database for storing user information, portfolio data, templates, and subscription details. A key design feature is the use of JSONB fields to store flexible and semi-structured user input data. This approach enables dynamic schema evolution without requiring frequent database migrations, thereby improving system adaptability.

In addition to PostgreSQL, Redis is employed as an in-memory data store to enhance system performance and scalability. Redis is used for multiple purposes, including:

- Caching AI-generated results with a defined time-to-live (TTL) to reduce repeated computations
- Storing frequently accessed portfolio data for faster retrieval
- Managing authentication tokens and implementing rate limiting mechanisms

This hybrid data storage strategy combines the reliability of relational databases with the speed of in-memory caching, ensuring both data consistency and high performance under varying workloads.

5. External Service Integration

The Portiva system integrates external services to extend functionality and support a complete SaaS ecosystem. Payment processing is handled through Razorpay, enabling secure subscription management, billing, and monetization of the platform.

The AI engine is designed as a loosely coupled service, allowing flexibility in deployment and future enhancements. This design enables the system to switch between locally hosted models and cloud-based AI services without impacting the core application architecture.

The modular nature of the system ensures that additional external services—such as analytics

tools, email services, or third-party integrations—can be incorporated seamlessly. This extensibility supports future scalability and continuous feature expansion while maintaining system stability.

6. System Architecture

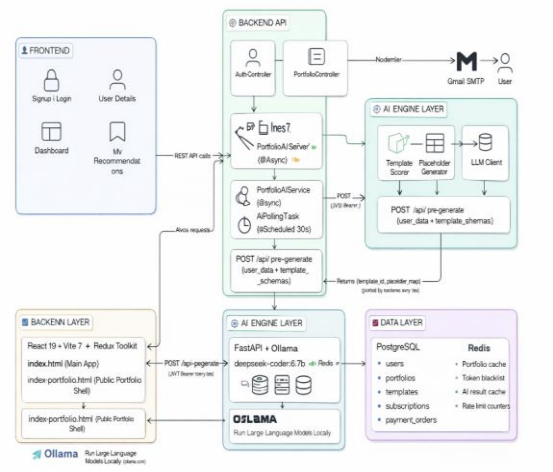

Figure1: System Architecture of Portiva – A Scalable SaaS Platform for AI-Driven Portfolio Generation

7. A seamless flow from input to intelligent output

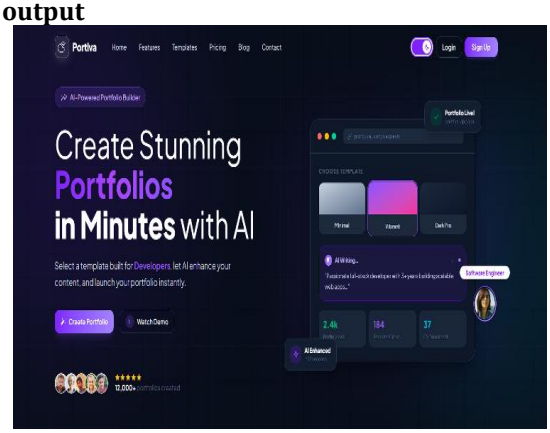

Figure 2: The Home page serves as the main interface for interacting with the system.

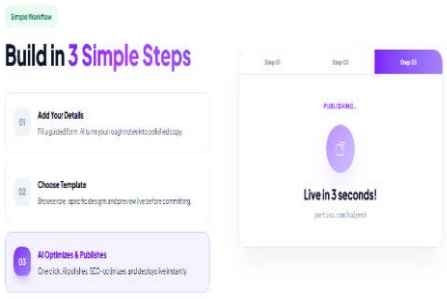

Figure 3: Turn your ideas into reality in seconds with intelligent automation

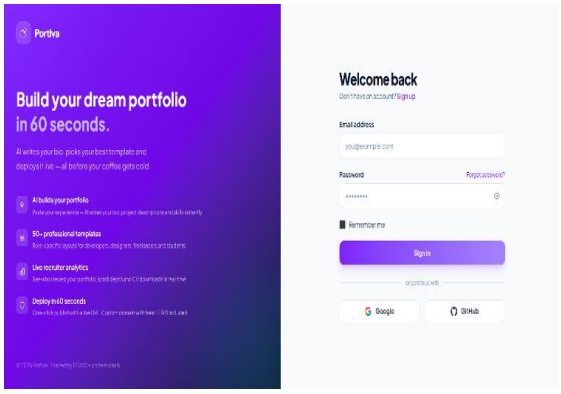

Figure 4: The Authentication module verifies user identity and protects system data

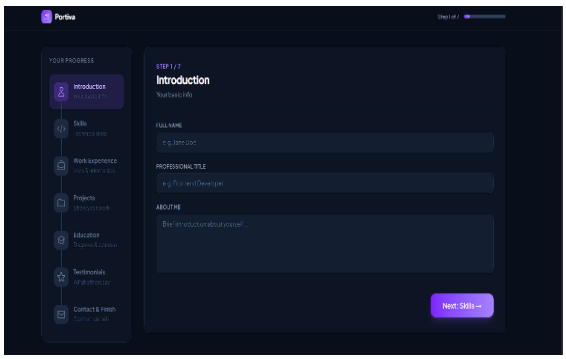

Figure 5: Start your journey by adding your details — simple, guided, and seamless.

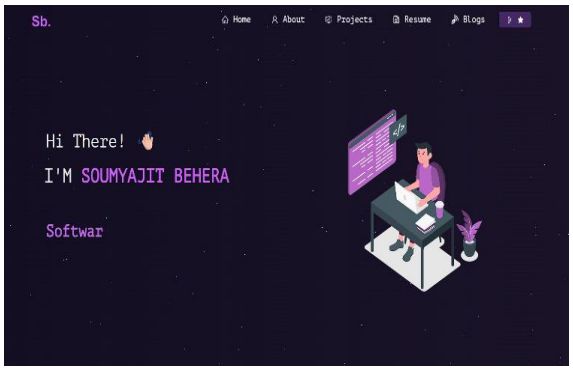

Figure 6: The portfolio is automatically deployed to a live public URL, making it instantly accessible and shareable.

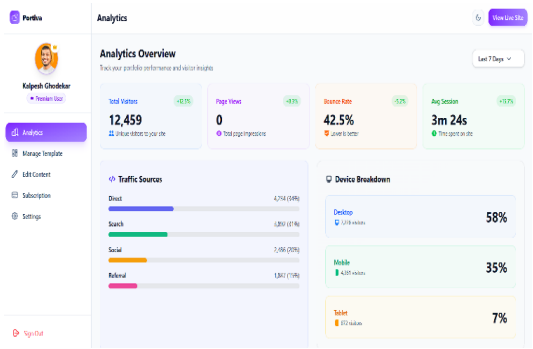

Figure 7: Track your growth with real-time insights and performance analytics.

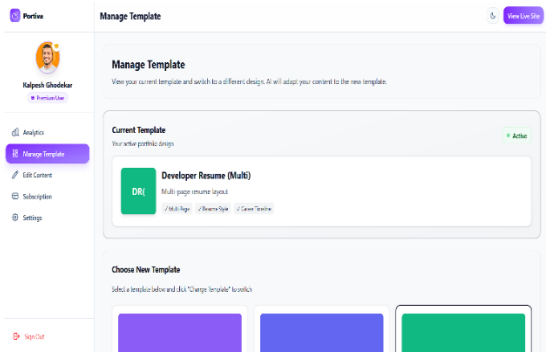

Figure 8: Customize your portfolio by choosing and switching templates effortlessly.

Conclusion

This paper presented **Portiva**, a cloud-based Software-as-a-Service (SaaS) platform designed to automate the creation, deployment, and management of professional portfolio websites using a schema-driven AI architecture. By integrating structured data collection, intelligent template selection, and Large Language Model (LLM)-based content generation, the system enables the rapid development of highly personalized and production-ready portfolios within minutes.

Unlike traditional website builders and existing AI-driven tools, Portiva introduces a formal contract between templates and AI generation through a database-defined *placeholder schema*. This approach improves consistency, scalability, and flexibility, allowing new templates to be added without requiring changes to application logic. In addition, the two-phase generation pipeline, which combines deterministic data mapping with selective AI-based content generation, optimizes performance by reducing computational overhead while maintaining high-quality outputs.

The system demonstrates strong performance through asynchronous processing, Redis-based caching, and efficient API design, achieving low latency and high scalability. By automating both content creation and deployment, Portiva reduces the technical barrier for users and significantly accelerates the portfolio development process.

Despite these advantages, certain challenges remain. The quality of AI-generated content may vary depending on the underlying model, and local inference can introduce latency on resource-constrained systems. Furthermore, ensuring content accuracy and maintaining user control over generated outputs are important considerations for real-world deployment.

Future work will focus on enhancing the system through GPU-accelerated inference to reduce generation time, supporting multiple models including both local and cloud-based LLMs for improved output quality, enabling real-time analytics and personalization based on user behavior, integrating mobile applications for broader accessibility, and developing collaborative template ecosystems for community-driven design. These improvements will further strengthen Portiva's contribution to automated web development and AI-driven personalization, making it a scalable and practical solution for modern digital professionals.

References

Wix.com, "Wix Website Builder," [Online]. Available: https://www.wix.com. [Accessed: 2025].

Squarespace Inc., "Squarespace Website Builder," [Online]. Available: https://www.squarespace.com. [Accessed: 2025].

Framer B.V., "Framer AI — Design and Publish," [Online]. Available: https://www.framer.com/ai. [Accessed: 2025].

Pivotal Software Inc., "Spring Boot Reference Documentation v3.2," 2024. [Online]. Available: https://docs.spring.io/spring-boot/docs/3.2.x/reference/html.

S. Weerasinghe and I. Perera, "Taxonomical Classification and Systematic Review on Microservices," International Journal of Engineering Trends and Technology (IJETT), vol. 70, no. 3, pp. 222–232, Mar. 2022. doi: 10.14445/22490183/IJETT-V70I3P225.

P. Zhang, Y. Yang, Z. Song, and L. Xiang, "Adaptive Load Balancing and Fault-Tolerant Microservices Architecture for High-Availability Web Systems Using Docker and Spring Cloud," Discover Applied Sciences, Springer Nature, Jul. 2025. doi: 10.1007/s42452-025-07320-7.

The PostgreSQL Global Development Group, "PostgreSQL 15 Documentation — Chapter 8: JSON Types," 2023. [Online]. Available: https://www.postgresql.org/docs/15/datatype-json.html.

E. Jones, "PostgreSQL Large JSON Value Query Performance," Feb. 2022. [Online]. Available: https://www.evanjones.ca/postgres-large-json-performance.html.

L. Shi, H. Qiao, C. Yang, Y. Jiang, K. Yu, and C. Chen, "Research and Application of Distributed Cache

Based on Redis," Journal of Software, vol. 19, no. 1, pp. 493–510, Jan. 2024.

Kaptosv, "Using Redis for Caching Optimization in High-Traffic Web Applications," International Journal of Advanced Multidisciplinary Research and Studies, vol. 5, no. 4, pp. 1714–1722, 2025.

D. Seth, A. Singh, and S. Panyam, "Distributed Caching Challenges and Strategies in Enterprise Applications," International Research Journal of Modern Engineering Technology and Science, vol. 6, pp. 115–126, 2023. doi: 10.56726/IRJMETS58564.

Redis Ltd., "Redis Documentation — Data Types and Commands," 2024. [Online]. Available: https://redis.io/docs.

Meta Platforms Inc., "React 19 — The Library for Web and Native User Interfaces," 2024. [Online]. Available: https://react.dev.

Vite Contributors, "Vite 7 — Next Generation Frontend Tooling," 2024. [Online]. Available: https://vitejs.dev.

S. Tiangolo, "FastAPI — Modern, Fast Web Framework for Building APIs with Python," 2024. [Online]. Available: https://fastapi.tiangolo.com.

IETF, "RFC 7159 — The JavaScript Object Notation (JSON) Data Interchange Format," T. Bray, Ed., Mar. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7159.

OpenAPI Initiative, "OpenAPI Specification v3.1.0," 2021. [Online]. Available: https://spec.openapis.org/oas/v3.1.0.

Razorpay Software Pvt. Ltd., "Razorpay Payment Gateway — Developer Documentation," 2024. [Online]. Available: https://razorpay.com/docs.

Redux Toolkit Contributors, "Redux Toolkit 2.x — Official Documentation," 2024. [Online]. Available: https://redux-toolkit.js.org.

M. Aleksendrić, Full Stack FastAPI, React, and MongoDB. Birmingham, UK: Packt Publishing, 2022. [Online]. Available: https://www.oreilly.com/library/view/full-stack-fastapi/9781803231822/.