Archives available at journals.mriindia.com

International Journal on Advanced Computer Engineering and Communication Technology

ISSN: 2278-5140

Volume 15 Issue 01, 2026

An AI-Driven SaaS Platform for Automated Portfolio Generation, Live Publishing, and Schema-Guided Template Rendering

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

[1]Assistant Professor- Dept. of Computer Engineering, SITRC Nashik

[2,3,4,5] B.E – Dept of Computer Engineering, SITRC Nashik

Email: [1].abhay.gaidhani@sitrc.org, [2]yashraj07thube@gmail.com, [3]kalpeshghodekar2004@gmail.com, [4]rishikeshmarathe04@gmail.com, [5]Shreyashagare437@gmail.com

Peer Review Information

Submission: 28 Jan 2026

Revision: 22 Feb 2026

Acceptance: 13 March 2026

Keywords

Large Language Model; 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 that automates the creation, live publishing, and management of professional portfolio websites. The system accepts structured user data through a guided multi-step form, delegates content synthesis to a locally-hosted Large Language Model (LLM) via Ollama, and renders a fully live portfolio on a unique public URL — all within minutes of signup. The core contribution of this work is a novel schema-driven placeholder mapping architecture in which each template declaratively specifies its data requirements through a structured placeholder_schema stored in the database. The AI engine reads this schema to generate precisely the content each template needs, eliminating hardcoded field mappings and enabling templates to be added without modifying application code. The platform is built on a modern stack — React 19, Spring Boot 3.2, FastAPI, PostgreSQL, and Redis — and follows production-grade practices including asynchronous AI generation, Redis-based result caching, soft deletes, and Razorpay-integrated subscription management. Experimental evaluation demonstrates sub-200ms API response for portfolio generation, successful dynamic template selection through a scoring algorithm, and complete end-to-end portfolio delivery in under three minutes on commodity hardware.

Introduction

The rapid growth of the digital economy has significantly increased the importance of personal branding, particularly for software engineers, designers, and other digital professionals. A well-designed portfolio website serves as a critical medium for showcasing skills, projects, and professional achievements. However, creating and maintaining such a portfolio remains a complex task requiring technical expertise in web development, design principles, hosting, and content creation.

Traditional approaches to portfolio creation rely on manual website builders or custom development, which are often time-consuming and require considerable effort. As a result, many individuals—especially students and early-career professionals—either delay building portfolios or create suboptimal ones that fail to effectively represent their capabilities.

With the emergence of Artificial Intelligence (AI), there is an opportunity to automate and personalize portfolio generation. However, existing AI-based website builders still lack structured data collection, dynamic adaptability, and a formal integration between generated content and visual templates. This paper introduces **Portiva**, a schema-driven SaaS

platform that automates portfolio creation, content generation, and live deployment through a scalable and intelligent architecture.

1. Personalized Portfolio Generation using AI

Modern AI technologies, particularly Machine Learning (ML) and Large Language Models (LLMs), enable systems to generate high-quality, context-aware content based on structured user input. In the context of portfolio generation, these models can analyze user-provided data such as skills, experience, projects, and career goals to produce professional summaries, project descriptions, and personalized branding content. Unlike traditional systems that rely on manual content writing, Portiva leverages LLMs to automate this process. The system uses structured onboarding data as input and generates only the required content dynamically. This significantly reduces user effort while improving content quality and consistency. Additionally, by constraining AI generation through predefined schemas, the system ensures relevance and avoids unnecessary or generic outputs.

2. System Architecture and Data Integration

Portiva follows a modern microservices-based architecture integrating multiple technologies for scalability and performance. The system consists of a React-based frontend, a Spring Boot backend, and a FastAPI-powered AI engine. Data is stored in PostgreSQL, while Redis is used for caching and asynchronous processing.

The platform processes both structured and semi-structured data efficiently. User inputs collected through a multi-step form are stored as flexible JSON objects, allowing schema evolution without database migrations. The backend aggregates user data and template schemas, which are then processed by the AI engine to generate content.

By combining structured data handling with AI-driven processing, the system minimizes manual intervention, improves performance, and ensures seamless communication between different components of the architecture.

3. Automated Template Rendering and Content Generation

Portiva is designed to generate complete, production-ready portfolio websites by combining structured data with intelligent template rendering. The system collects detailed user information, including skills, experience, projects, education, and achievements, and uses this data to populate dynamic templates.

A key innovation is the use of a **schema-driven placeholder mapping system**, where each template defines the exact content it requires. The AI engine reads this schema and generates only the necessary fields, ensuring a precise match between data and design.

Additionally, the platform supports dynamic template switching without requiring users to re-enter data. This flexibility allows users to experiment with different designs while maintaining consistent content. The result is a highly personalized, visually appealing portfolio that can be published instantly via a public URL.

4. Research Gaps and System Challenges

Despite advancements in AI-driven website generation, several challenges remain. Existing systems often lack a structured connection between content generation and template rendering, leading to inconsistencies and reduced adaptability. Many platforms also depend heavily on manual input or generic prompts, limiting personalization and scalability. Another key limitation is the absence of efficient architectures that minimize computational overhead while maintaining high-quality output. Systems that rely on multiple AI calls or unoptimized pipelines often face performance bottlenecks and increased costs.

Furthermore, important considerations such as data privacy, system transparency, and user control over generated content must be addressed. Ensuring secure data handling, explainable AI outputs, and reliable system behavior is essential for building trust in AI-powered platforms.

Portiva addresses these gaps through a schema-driven architecture, a two-phase AI generation pipeline, and an asynchronous processing model, enabling efficient, scalable, and reliable portfolio generation.

Literature Survey

Over the past decade, website development and portfolio generation systems have evolved from manual coding approaches to automated and AI-assisted platforms. Early solutions relied heavily on static templates and user-driven content creation, requiring significant technical expertise and design effort. With the advancement of machine learning and generative AI, modern platforms can now assist in content creation and layout design by analyzing minimal user input.

Despite these advancements, many existing systems still lack structured data integration, adaptability, and a formal connection between generated content and template rendering. Most AI-based website builders depend on unstructured prompts and produce static outputs that are difficult to modify or reuse. Furthermore, they often do not support dynamic

updates or continuous personalization based on user data. These limitations highlight the need for more intelligent, schema-driven systems, leading to the development of platforms such as **Portiva**, which aim to provide automated, scalable, and structured portfolio generation through AI-driven architecture.

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 follow a generalized approach and do not adapt dynamically to individual user profiles. As a result, users without design or content-writing expertise often struggle to create professional-quality portfolios. These limitations emphasize the need for intelligent systems that can automate both content generation and design selection based on user-specific data.

2. AI-Based Website Generation Platforms

Recent advancements have introduced AI-powered website builders such as Framer AI and Durable.co, which generate websites based on short textual prompts. These platforms significantly reduce the effort required to create content; however, they still face several challenges.

One major limitation is the lack of structured data collection. Since these systems rely on prompts rather than detailed user profiles, the generated content may lack depth, accuracy, and personalization. Additionally, there is no formal mapping between AI-generated content and template placeholders, making it difficult to modify or reuse the generated output across different designs.

These shortcomings highlight the need for systems that combine structured data input with AI-driven generation, ensuring both accuracy and adaptability.

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, including professional summaries, project descriptions, and branding content. In the context of portfolio generation, LLMs can significantly reduce the burden of content creation by transforming structured user data into well-written, context-aware outputs.

However, many existing implementations use LLMs in an unstructured manner, generating entire websites or content blocks without constraints. This often leads to inconsistent outputs and inefficiencies, including multiple redundant AI calls.

A more effective approach is to integrate LLMs within a structured pipeline, where only specific content fields are generated based on predefined requirements. This improves efficiency, reduces computational cost, and ensures consistency across different templates.

4. Benefits of Personalization in Portfolio Systems

Research indicates that personalized digital content significantly improves user engagement and perceived value. In portfolio systems, personalization ensures that the content, design, and presentation align with the user's professional identity, skills, and career goals.

When portfolios are tailored to individual profiles, users are more likely to effectively communicate their strengths and stand out in competitive environments. Personalized systems also improve usability by reducing the effort required to create and maintain professional websites.

This demonstrates the importance of combining structured user data with intelligent systems to deliver highly customized and impactful portfolio experiences.

5. Contribution of the Portiva System

Building upon the limitations of existing systems, Portiva introduces a novel approach that integrates structured data collection, schema-driven template design, and AI-based content generation within a unified framework.

Unlike traditional and AI-based website builders, Portiva establishes a **formal contract between templates and AI generation** through a database-stored placeholder_schema. This enables the system to generate only the required content dynamically, ensuring consistency and eliminating hardcoded mappings.

Additionally, Portiva employs a **two-phase generation pipeline**, where deterministic data is directly mapped and only selective fields are generated using AI. This significantly improves efficiency and reduces computational overhead.

By combining structured data, intelligent template selection, and scalable architecture, Portiva enhances the accuracy, flexibility, and usability of automated portfolio generation systems, addressing key gaps identified in existing literature.

Methodology

The architecture of the **Portiva system** integrates modern web technologies, distributed system design, and AI-driven content generation to automate the creation and deployment of professional portfolio websites. The system is designed to efficiently collect structured user data, process it through an intelligent pipeline, and generate a fully functional portfolio aligned with user-specific requirements.

By combining a schema-driven template system, a two-phase AI generation pipeline, and a scalable microservices architecture, Portiva ensures efficient content generation, minimal latency, and high adaptability. The methodology is divided into multiple core components, including frontend interaction, backend orchestration, AI processing, data management, and external service integration, all working together to deliver a seamless and reliable user experience.

1. Frontend Design

The frontend of the Portiva system is developed using **React 19** with **Vite** for fast build performance and **Tailwind CSS** for responsive and consistent UI design. The interface is structured to provide a smooth and guided user experience through a multi-step onboarding process.

Users input structured professional data such as skills, experience, projects, education, and personal details. This guided approach ensures completeness and consistency of data while reducing user effort. The frontend also includes a dashboard for managing portfolios, switching templates, and viewing live previews.

A separate lightweight rendering layer is implemented for public portfolio pages, ensuring minimal load time by isolating it from application state management. This separation improves performance and scalability while maintaining a clean user experience.

2. Backend Framework

The backend is implemented using **Spring Boot 3.2**, providing a robust and scalable architecture for handling business logic, API communication, and data processing. It exposes RESTful endpoints for authentication, portfolio management, template handling, and subscription services.

The backend acts as an orchestrator between the frontend and the AI engine. Upon receiving user data, it stores the information in the database, triggers asynchronous AI generation, and manages status updates through polling mechanisms.

To ensure performance and scalability, the backend incorporates:

- **Asynchronous processing (@Async)** for non-blocking operations
- **Scheduled polling (@Scheduled)** to retrieve AI results
- **Redis caching** for fast data access and reduced latency

This design enables the system to handle multiple concurrent users efficiently while maintaining low response times.

3. AI Generation Engine

The AI engine is implemented using **FastAPI** and powered by a locally hosted LLM via **Ollama**. It is responsible for template selection and content generation using a structured, two-phase pipeline.

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

User-provided data is directly mapped to template placeholders without invoking the AI model. This includes fields such as name, skills, projects, and experience entries. This phase ensures fast and accurate population of structured data.

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

Only specific fields defined in the template's placeholder_schema are generated using the LLM. These include creative elements such as summaries, headlines, and taglines.

The template selection process is based on a scoring function:

$$Score(T) = \sum_{i=1}^{n} w_i \cdot f_i(user\_data)$$

where each template is evaluated based on user attributes such as skills, experience, and projects. By restricting AI generation to required fields, the system reduces computational cost and improves efficiency. The final output is a structured **placeholder_map** that is used for rendering the portfolio.

4. Database Management

Portiva uses **PostgreSQL** as the primary database to store users, portfolios, templates, and subscription data. A key feature is the use of **JSONB fields** to store flexible user input data, enabling schema evolution without requiring database migrations.

Redis is used as an in-memory data store for:

- AI result caching (24-hour TTL)
- Portfolio caching (fast retrieval)
- Token management and rate limiting

This hybrid data management approach ensures both consistency and high performance while supporting scalable system operations.

5. External Service Integration

The system integrates external services to enhance functionality and provide a complete SaaS experience. Payment processing is handled through **Razorpay**, enabling subscription-based access and monetization.

Additionally, the AI engine operates independently as a service, allowing flexibility in switching between local and cloud-based models in future extensions. The modular architecture ensures that external services can be integrated or replaced without affecting the core system.

6. System Architecture

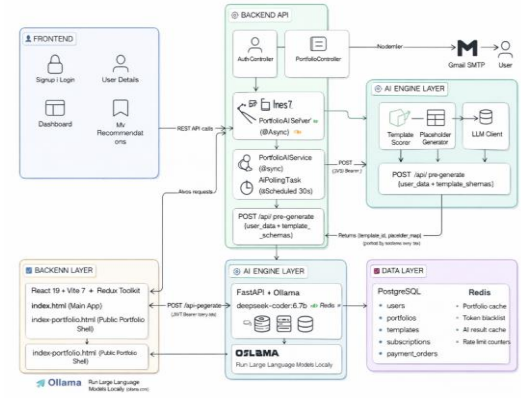
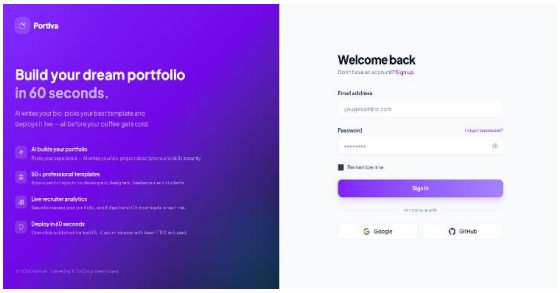
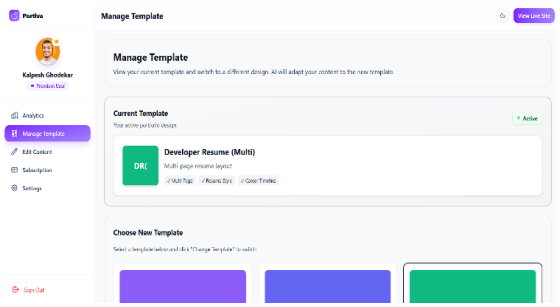
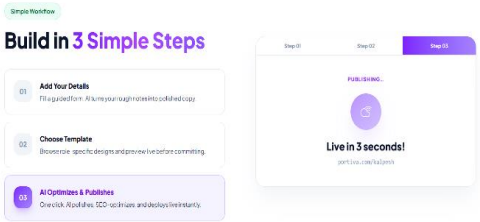
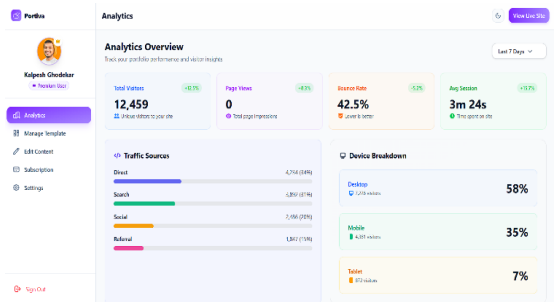


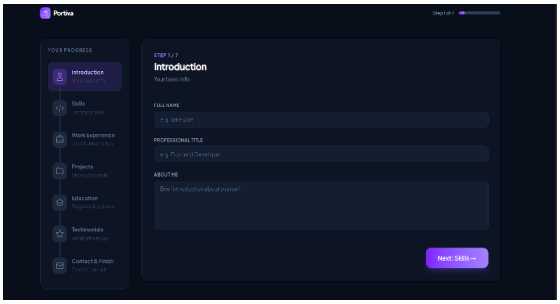
Figure 1

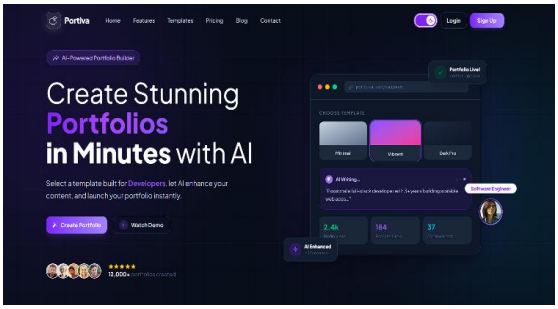












Conclusion

This paper presented **Portiva**, a cloud-based Software-as-a-Service (SaaS) platform that automates 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 delivers highly personalized and production-ready portfolios within minutes.

Unlike traditional website builders and existing AI-based tools, Portiva introduces a **formal contract between templates and AI generation** through a database-defined placeholder_schema. This approach ensures consistency, scalability, and flexibility, allowing new templates to be added without modifying application logic. Additionally, the **two-phase generation pipeline**, combining deterministic data mapping with selective AI generation, significantly reduces computational overhead while maintaining high-quality outputs.

The system also 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 enables faster portfolio development.

Despite these advancements, certain challenges remain. AI-generated content quality 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 system capabilities through:

- **GPU-accelerated inference** to reduce generation time
- **Multi-model support** (local + cloud LLMs) for improved quality
- **Real-time analytics and personalization** based on visitor behavior
- **Mobile application integration** for broader accessibility
- **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

OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, updated insights 2024–2025.

Anthropic, "Claude 3 Model Card and Capabilities," 2025.

Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," 2024–2025.

Meta AI, "LLaMA 3: Open and Efficient Foundation Language Models," 2025.

Ollama, "Run Large Language Models Locally," [Online]. Available: https://ollama.com. [Accessed: 2026].

FastAPI, "FastAPI Documentation (Production APIs)," [Online]. Available: https://fastapi.tiangolo.com. [Accessed: 2026].

Pivotal Software, "Spring Boot 3.2+ Reference Guide," [Online]. Available: https://docs.spring.io/spring-boot. [Accessed: 2026].

Meta Platforms Inc., "React 19 Documentation and Concurrent Features," [Online]. Available: https://react.dev. [Accessed: 2026].

PostgreSQL Global Development Group, "PostgreSQL 15+ Documentation," [Online]. Available: https://www.postgresql.org/docs. [Accessed: 2026].

Redis Ltd., "Redis 7 Documentation: Caching and Distributed Systems," [Online]. Available: https://redis.io/docs. [Accessed: 2026].

Microsoft, "Designing Microservices Architectures for Cloud-Native Applications," 2024–2025.

Amazon Web Services (AWS), "Building Scalable SaaS Applications: Architecture Best Practices," 2025.

Framer B.V., "Framer AI Website Builder," [Online]. Available: https://www.framer.com/ai. [Accessed: 2026].

Durable AI, "AI Website Generation Platform," [Online]. Available: https://durable.co. [Accessed: 2026].

Wix.com, "Wix Website Builder Platform," [Online]. Available: https://www.wix.com. [Accessed: 2026].

Squarespace Inc., "Squarespace Website Builder," [Online]. Available: https://www.squarespace.com. [Accessed: 2026].

Stripe Inc., "Subscription-Based SaaS Billing Systems," 2025.

Razorpay, "Payment Gateway Integration for SaaS Platforms," [Online]. Available: https://razorpay.com. [Accessed: 2026].

M. Fowler, "Microservices Architecture Patterns," 2024 Edition.

T. Grance et al., "NIST Cloud Computing Reference Architecture," Updated Guidelines 2025.