IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

A Prompt-Driven Web Architecture for AI-Generated Graphic Design: The AI Designer Pro Platform

PG Ruchikar Narasimha S A[1], Prof. Mahalakshmi M[2] and PG Shashank B V[1]
[*1,2]Student, Department of Computer Applications
[*3]Faculty, Department of Computer Applications
Maharaja Institute of Technology, Mysore, Mandya, Karnataka, India.

Abstract: *Producing a usable graphic design — a logo, a poster, a promotional banner, an identity card, or a visiting card — conventionally requires either design software skill or a template that constrains the result to a fixed layout. This paper presents AI Designer Pro, a web application that instead treats design production as a prompt-driven task: a user describes the design they want in natural language, and the system forwards that description to a commercial AI image-generation API, storing and organizing the result rather than requiring any manual composition. The platform is built as a conventional PHP and MySQL web application layered around session-based authentication, five dedicated category-specific generator pages, a single AI-generation endpoint that enforces a free-tier usage quota, a searchable gallery with a lightweight in-browser editor, and a Razorpay-based payment flow for plan upgrades. Beyond describing the architecture and implementation, this paper reports the outcome of a structured functional and security testing phase carried out across every module, including a defect log documenting specific weaknesses — among them a path-traversal vulnerability and an unescaped-output cross-site-scripting gap — identified and resolved during development. The result is a working, end-to-end prototype demonstrating that a small, modular web application can package general-purpose text-to-image generation into a focused, everyday design tool with a viable free-to-paid usage model, while surfacing the concrete secure-coding discipline such a system requires.*

Keywords: *Artificial intelligence, text-to-image generation, web application security, graphic design automation, PHP, prompt engineering, usage-based monetization*

I. INTRODUCTION

Everyday graphic design needs — a logo for a new venture, a poster for a college event, a banner for a sale, an identity card for a small organization — are common, but the tools available to meet them sit at two extremes. Professional design software offers complete creative control but demands skill most occasional users do not have. Template-based platforms lower that skill barrier but confine every output to a pre-built layout, so results are rarely original and rarely feel tailored to what the user actually had in mind.

Recent advances in text-conditioned image generation have made it practical to turn a short natural-language description directly into a coherent image without any manual drawing. This paper is motivated by the observation that this capability, rather than being left as a general-purpose chat or image tool, can be narrowed deliberately to a small set of everyday design categories and wrapped in the organizational and monetization scaffolding a real product needs — accounts, a gallery, basic editing, and a usage-based payment model. AI Designer Pro is presented as exactly this: a focused, prompt-driven design generator rather than a general template library or an unconstrained AI chat interface.

This paper describes the system's architecture, its implementation across five design categories, and — distinct from a purely descriptive systems paper — reports the outcome of the structured testing phase applied to it, including specific

DOI: 10.48175/IJARSCT-37909

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

defects identified and corrected, since the security posture of a system that lets user input reach the file system and a database is as material to its viability as the AI integration itself.

II. RELATED WORK

Text-to-image generation has progressed from early transformer-based approaches to diffusion-based models that now underlie most commercial image-generation APIs. Zero-shot text-to-image generation was demonstrated by a large-scale transformer trained on image-text pairs, showing that a single model could generalize to prompts unseen during training [1]. Subsequent work introduced latent diffusion models, performing the generative process in a compressed latent space rather than directly on pixels, substantially reducing the computational cost of high-resolution synthesis and underlying many of the commercial APIs available today, including the class of service this project depends on [2]. The underlying denoising diffusion process itself was formalized separately, establishing the iterative noise-removal procedure that diffusion-based generation builds on [3].

Two further strands of work underlie the practical behaviour of prompt-conditioned generation. The Transformer architecture's self-attention mechanism, originally proposed for language modelling, is a core component of the text encoders that condition image models on a user's prompt [4], and joint image-text embedding models have shown that aligning visual and textual representations at scale improves how faithfully a generated image follows its input description [5, 6]. Separately, generative adversarial networks represent the generation paradigm that dominated the field prior to diffusion-based approaches and remain useful context for understanding why the latter has become the standard for commercial APIs [7].

On the systems side, template-based design platforms represent the dominant existing approach to accessible graphic design, trading originality for ease of use — the opposite trade-off this project explores [8]. On the security side, the OWASP Top 10 catalogues the categories of web-application weakness most relevant to a system that accepts user-controlled filenames and renders user-supplied metadata, including broken access control (which covers path traversal) and injection [9], and standard software-engineering references inform the structured, module-by-module testing methodology applied in this work [10].

III. PROBLEM STATEMENT

Existing options for producing a quick graphic design fall into two categories, neither of which serves a user who wants an original result with minimal effort. Professional design software offers full creative control at the cost of a real skill investment. Template-based platforms lower that skill requirement but restrict the user to a fixed set of layouts, fonts, and stock imagery. There is a gap for a tool that generates an original design directly from a short text description, tailored to a specific, common design category, while still giving the user a way to organize, revisit, lightly edit, and download what has been generated — and that does so with the secure-coding discipline appropriate to a system handling file uploads, generated media, and payment data.

IV. OBJECTIVES AND SCOPE

4.1 Objectives

- Build a secure web application supporting registration, login, and session-based access control.
- Provide dedicated, category-specific generator pages (logo, poster, banner, ID card, visiting card), each translating a text prompt into a generation request.
- Integrate a commercial AI image-generation API behind a single, quota-enforcing endpoint.
- Persist each generated design together with descriptive metadata in a lightweight, file-based format.
- Provide a searchable gallery, an in-browser editor for basic transforms and metadata edits, and a Razorpay-based payment flow for plan upgrades.
- Apply and verify secure coding practices — input validation, output escaping, and parameterized queries — through structured testing rather than by assumption.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

4.2 Scope

The platform is domain-narrow by design: it supports five common design categories rather than an open-ended canvas, since the value of the approach lies in pairing a focused workflow with AI generation rather than in generality. The current implementation targets a single-server deployment, storing user and payment records relationally in MySQL while storing generated-design metadata as per-image JSON files on the file system; the architecture is structured so this storage approach, the AI provider, and the payment gateway can each be substituted without disturbing the remaining layers.

V. SYSTEM ARCHITECTURE

The platform is organized into four layers — presentation, application, AI integration and payments, and data — structured so each can be modified independently.

5.1 Presentation Layer

Server-rendered PHP pages provide the landing site, dashboard, the five generator pages, the gallery, the editor, and an administrative view, with vanilla JavaScript used only for client-side interactivity such as gallery search and the editor's transform controls.

5.2 Application Layer

PHP request handlers implement session-based authentication and the business logic coordinating the remaining layers: validating input, checking quota state, and dispatching to the AI, payment, and data layers as appropriate.

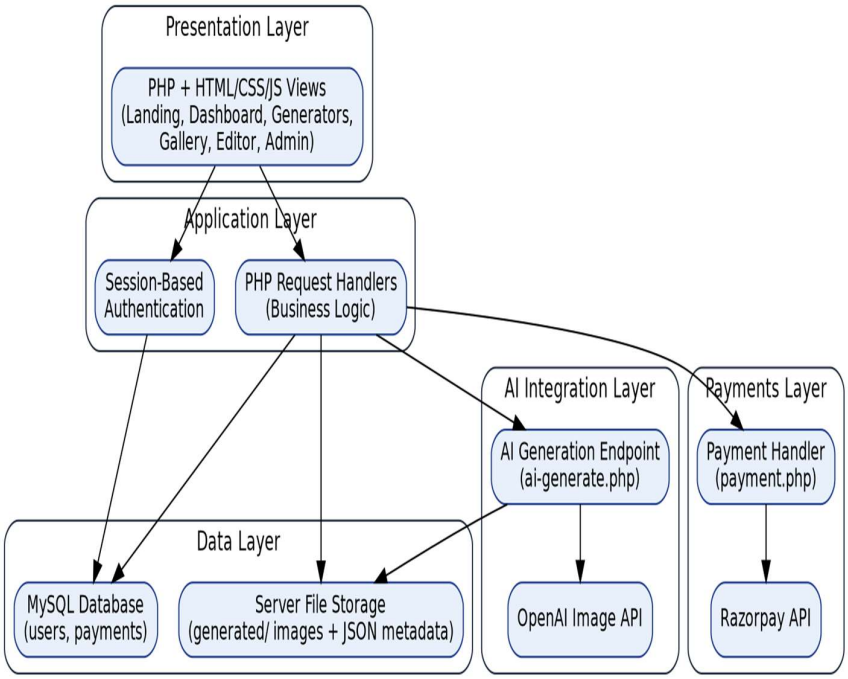

Fig. 1: System Architecture of AI Designer Pro

5.3 AI Integration and Payments Layer

A single AI-generation endpoint wraps all calls to the external image-generation API, centralizing quota enforcement so it cannot be bypassed by calling a generator page directly. A parallel payment layer wraps calls to the Razorpay order-creation and confirmation APIs, resolving plan prices from a server-side table rather than trusting any client-supplied amount.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

5.4 Data Layer

MySQL stores user accounts and payment records, which benefit from relational constraints such as a unique email address; generated designs are stored directly on the file system with a companion JSON metadata document per image, avoiding the overhead of a media database table for a single-server deployment.

VI. IMPLEMENTATION DETAILS

6.1 Technology Stack

Frontend: HTML5, CSS3, vanilla JavaScript
Backend: PHP
Database: MySQL
AI Integration: OpenAI Image-Generation API
Payment Gateway: Razorpay (order creation and confirmation APIs)
Deployment target: Apache-based LAMP-compatible hosting

6.2 Generation and Quota Enforcement

A generation request carries the user's prompt, category-specific fields, and a source identifier to the shared AI endpoint. The endpoint resolves the requesting identity — a logged-in user by session, or a guest by a session-scoped counter — and checks remaining free-tier quota before making any external call; only if quota remains does it forward the prompt to the image-generation API, persist the returned image alongside a JSON metadata document, and increment the used-generation counter.

6.3 Data Flow

Figure 2 traces the system as a single process exchanging data with the user, the AI image-generation API, the Razorpay gateway, the MySQL database, and the server's file storage, showing the boundaries across which validated data must cross in either direction.

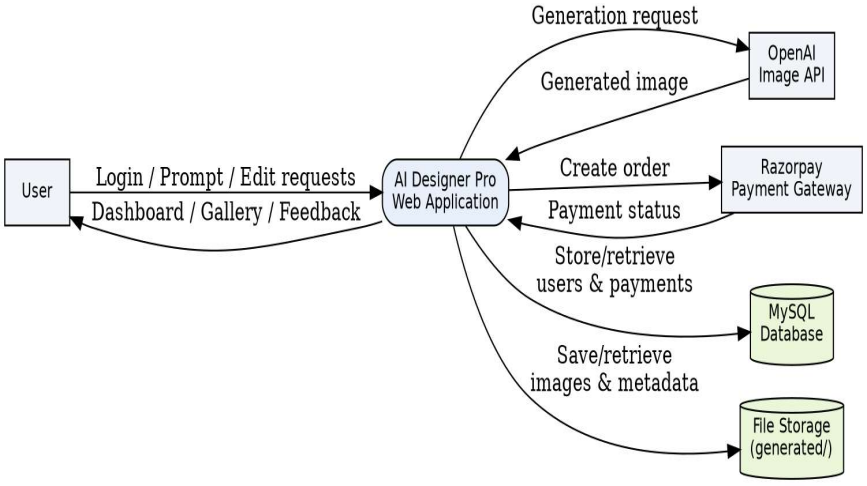

Fig. 2: Data Flow Diagram — Level 0

6.4 Security Measures

Three measures were applied consistently across every module handling user input. First, any filename accepted from a request is reduced to a bare filename and checked against the list of files actually present on the server before being used in a file operation, preventing path traversal. Second, every dynamic value rendered into HTML — filenames, titles, tags, descriptions — is escaped, preventing cross-site scripting. Third, every database query that incorporates user-supplied

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

values uses a parameterized (prepared) statement rather than string concatenation, preventing SQL injection. Destructive actions, such as deleting a generated design, are additionally restricted to POST requests guarded by a session-bound token.

VII. HARDWARE AND SOFTWARE REQUIREMENTS

The platform requires no specialized hardware: a standard development or hosting machine with at least 8 GB of RAM and a stable internet connection is sufficient, since all AI processing and payment handling occur through external APIs rather than local computation. The software stack — PHP, MySQL, and a standard Apache web server — is available as free, open-source tooling on Windows, Linux, or macOS, keeping the platform accessible on consumer-grade infrastructure.

VIII. RESULTS AND DISCUSSION

Rather than an informal usage pilot, this work's evaluation consisted of a structured functional and security testing phase applied module by module as the system was built, followed by end-to-end testing of complete user journeys. Thirty test cases were executed across authentication, the generator and quota-enforcement logic, the gallery, the editor, cross-cutting security checks, the payment flow, and the admin dashboard, all of which passed in the final test run, summarized in Fig. 3.

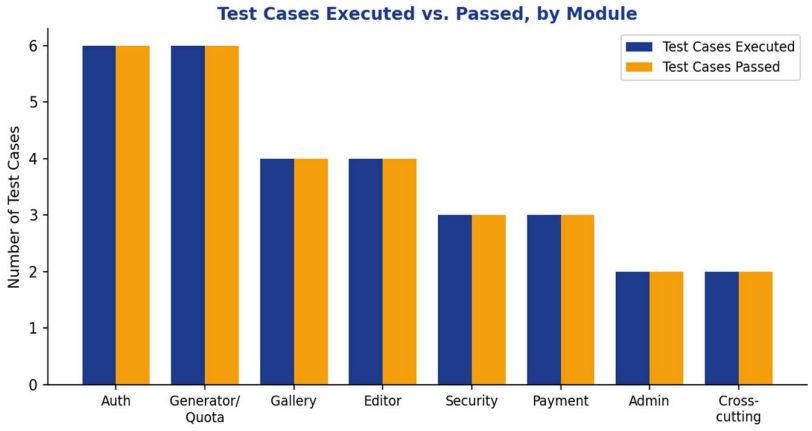

Fig. 3: Test Cases Executed vs. Passed, by Module (final test pass)

That 100% final pass rate reflects defects resolved earlier in testing rather than an absence of issues found. Eight defects were identified and corrected during development, concentrated most heavily in the editor module, which is also the module with the largest attack surface since it accepts a filename directly from a request parameter. Figure 4 breaks these down by module.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

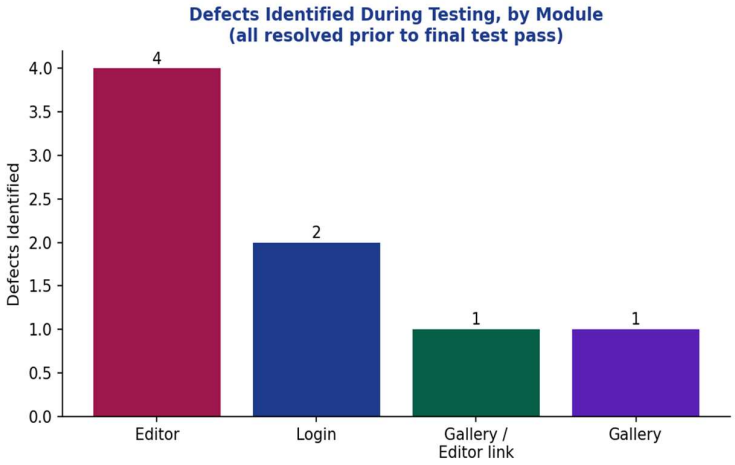

Fig. 4: Defects Identified During Testing, by Module

Two of the eight defects are worth discussing specifically because they represent distinct classes of weakness rather than a single repeated mistake. The first was a path-traversal weakness in the editor, where a filename supplied via the request was used to build a file path without being checked against the set of files actually present on the server; this was resolved by extracting the bare filename and validating it against that whitelist before any file operation. The second was an authentication logic defect in which a session key checked at the top of the editor page did not match the key structure actually set at login, so the page attempted (and, when the browser had already received partial output, silently failed) a redirect on every request regardless of login state, producing a blank page rather than the intended behaviour; this was traced to a mismatch between two independently written modules and resolved by aligning the check to the session structure the login module actually sets.

Both findings support a broader point about small, modular web applications built incrementally: module-by-module testing surfaces integration-level defects — like the session-key mismatch — that unit-level reasoning about a single file would not reveal, since the defect only manifests when two modules' assumptions about shared state diverge. This reinforces the value of testing each module both in isolation and against the modules it depends on, rather than treating a passing individual-file review as sufficient evidence of correctness.

IX. CONCLUSION AND FUTURE SCOPE

This paper presented AI Designer Pro, a prompt-driven web platform that narrows general-purpose AI image generation to five everyday design categories, wrapped in session-based authentication, a searchable gallery with lightweight editing, and a free-to-paid usage model backed by a real payment gateway. Rather than reporting an informal usage pilot, this work reported the outcome of a structured testing phase, including a defect log documenting the specific path-traversal, output-escaping, and session-handling issues identified and resolved during development — evidence that the security discipline required of such a system was applied and verified rather than assumed.

The work remains a single-developer academic project verified through manual functional and security testing rather than automated test suites or a live multi-user pilot. Planned future work includes migrating generated-design metadata from per-image JSON files into a dedicated database table to support richer search as usage scales, adding automated regression testing (for example, with PHPUnit) to complement the manual test suite reported here, introducing automated content-moderation checks on AI-generated output before it is shown to a user, and extending the platform to additional design categories using the same generator-page pattern.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 4, July 2026
Impact Factor: 8.2

REFERENCES

[1] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-Shot Text-to-Image Generation," Proc. 38th Int. Conf. Machine Learning (ICML), 2021.
[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), 2022.
[3] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," Proc. 38th Int. Conf. Machine Learning (ICML), 2021.
[6] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," arXiv preprint, 2022.
[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," Advances in Neural Information Processing Systems (NeurIPS), vol. 27, 2014.
[8] Canva Pty Ltd, "Canva — Design Platform Overview," Available: https://www.canva.com/. Accessed: July 2026.
[9] OWASP Foundation, "OWASP Top 10:2021," Available: https://owasp.org/Top10/. Accessed: July 2026.
[10] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed., McGraw-Hill Education, 2019.
[11] OpenAI, "Images API Documentation," OpenAI Platform Documentation. Available: https://platform.openai.com/docs/guides/images. Accessed: July 2026.
[12] Razorpay, "Orders API Documentation," Razorpay Developer Documentation. Available: https://razorpay.com/docs/api/orders/. Accessed: July 2026.
[13] OWASP Foundation, "Cross-Site Request Forgery Prevention Cheat Sheet," Available: https://cheatsheetseries.owasp.org/. Accessed: July 2026