International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Domagineers: A Human–AI Collaborative Platform for Modular Website and Application Development

Thushi Suhana S[1], Charishma P[2], Bhavish Nagarajan M[3], MS. Nidhiben Shailesh Soni[4]

[1,2,3]Department of Computer Science and Engineering, M. S. Ramaiah University of Applied Sciences, Bangalore, Karnataka, India – 560058
[4]Faculty & Staff, Department of Computer Science and Engineering, M. S. Ramaiah University of Applied Sciences, Bangalore, Karnataka, India

Abstract
In recent years, the demand for rapid and flexible website development tools has increased significantly, particularly among startups and non-technical users. Existing solutions either require extensive technical expertise or rely on restrictive automation that limits creative freedom. This paper presents DOMAGINEERS, a modular AI-assisted website builder designed to bridge the gap between technical complexity and creative control. The system adopts a human–AI collaborative approach where users interactively generate, modify, and approve website components with AI assistance. The platform integrates component-based architecture, real-time preview, and automated deployment to enable section-by-section website construction. Experimental validation and user testing demonstrate that the proposed approach improves usability, personalization, and development efficiency compared to fully automated builders. The results indicate that guided human–AI collaboration produces more adaptable and educational outcomes in web development workflows.

Keywords: Human–AI Collaboration, AI-Assisted Web Development, Modular Architecture, Website Builder, Real-Time Preview.

1. Introduction

The growing reliance on digital platforms has increased the need for efficient and accessible website and application development solutions. Traditional development methods often require significant technical expertise, creating barriers for non-technical users and increasing development effort.

Recent advances in Artificial Intelligence, particularly Large Language Models, have enabled automatic code generation from natural language inputs. Although such tools improve development speed, many existing solutions rely on full automation, which limits transparency, customization, and user control.

Research indicates that human–AI collaboration is more effective than fully automated approaches for complex and creative development tasks. In parallel, modern web development has adopted modular and component-based architectures to improve scalability and maintainability.

Motivated by these observations, this research introduces Domagineers, a human–AI collaborative platform for modular website and application development. The platform integrates AI-assisted code generation, real-time preview, and automated deployment to balance efficiency with transparency and control.

2. Literature Review

Prior studies have consistently shown that the adoption of modular and component-based design approaches enhances scalability, simplifies maintenance, and supports long-term system evolution in web applications[8] [11]. Framework-level research further indicates that component-oriented architectures reduce redundancy and accelerate development, particularly in large-scale systems, while modern frontend frameworks enable consistent and efficient interface iteration[17] [18].

Artificial Intelligence has increasingly been integrated into web development workflows to reduce manual effort through automated layout generation, styling assistance, and early-stage code creation[10] [24]. However, existing evaluations of low-code and no-code platforms reveal notable limitations related to customization flexibility, transparency, and extensibility, highlighting the need for systems that balance automation with meaningful user control[13] [25].

Recent empirical research suggests that collaborative human–AI approaches are more effective than fully automated systems, particularly for creative and technically complex tasks. Multiple studies report improved output quality, higher user satisfaction, and increased trust when AI operates as an assistive partner rather than an autonomous agent[1] [2] [16].

Advances in Large Language Models have significantly strengthened AI-assisted code generation capabilities. Benchmark-based evaluations demonstrate that models such as GPT-4 can reliably produce syntactically valid and executable code across a wide range of programming tasks[5]. At the same time, survey-driven studies emphasize that structured prompting strategies and human validation remain essential for ensuring contextual accuracy and correctness[6] [21].

Real-time interaction features and deployment automation further strengthen modern development workflows. Research on live preview systems shows that immediate visual feedback reduces errors and improves developer understanding during iterative development[9]. Similarly, studies on automated deployment mechanisms demonstrate that one-click deployment reduces operational complexity and simplifies production release processes[15].

Overall, the reviewed literature identifies a clear gap for platforms that integrate modular architecture, AI-assisted code generation, structured prompting, real-time preview, deployment automation, and human-in-the-loop control within a unified system. The Domagineers platform is designed to address this gap by combining these principles into a collaborative and transparent AI-assisted web development environment[1] [8].

3. Methodology

The methodology adopted in this research follows a human–AI collaborative development approach, where Artificial Intelligence assists in code generation while human users retain control over design decisions, validation, and refinement. The development process is structured around clearly defined objectives, corresponding methods, and supporting technologies.

Table 3.1 Objective-wise Methodology and Resources Utilised

Objective	Method	Description
OB1	Full-stack development	The platform is developed using modern web technologies including Next.js, React, and Tailwind CSS to support modular, scalable, and responsive user interfaces.
OB2	AI Integration	The Gemini API is integrated to enable clean and structured code generation based on user-provided natural language prompts.
OB3	Real-time Preview	A live preview mechanism is implemented to reflect frontend updates instantly as users modify prompts or generated code.
OB4	Database Management	Supabase is used for modular data storage, session management, and persistence of user interaction history.
OB5	Deployment	One-click deployment is enabled through integration with cloud hosting platforms such as Vercel and Netlify.

4. System Architecture

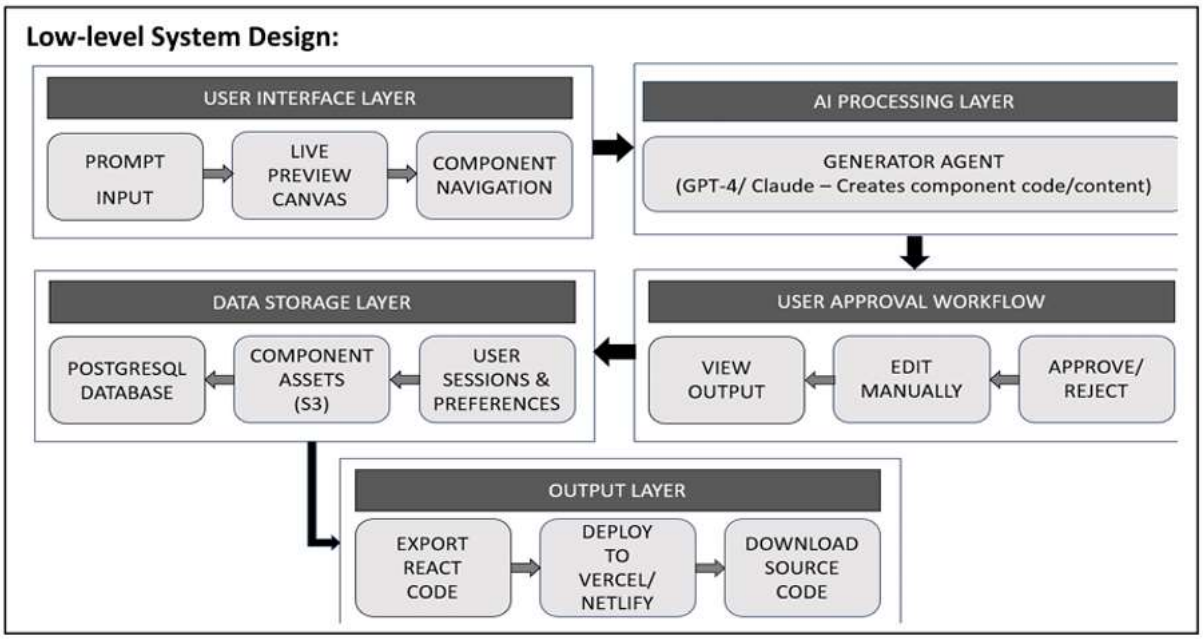


Figure 4.1 illustrates the high-level block diagram of the Domagineers system architecture, highlighting interactions between these components.

The Domagineers platform follows a layered and modular system architecture designed to support scalability, flexibility, and real-time interaction. The architecture separates responsibilities across multiple functional layers to simplify maintenance and enable incremental development.

The system consists of the following core components:

- **User Interaction Layer:** Provides interfaces for prompt input, code editing, and navigation across modules.
- **AI Processing Layer:** Processes user prompts and generates modular frontend and backend code using structured prompt templates.
- **Application Logic Layer:** Manages component integration, routing logic, and state consistency across generated modules.
- **Real-Time Preview Layer:** Renders application updates dynamically to provide immediate visual feedback during development.
- **Deployment Layer:** Handles build processes, environment configuration, and automated deployment to cloud hosting platforms.

5. Challenges of Each Module

5.1 Prompt Interpretation Module

Accurately interpreting user prompts expressed in natural language remains a significant challenge. Prompts may be incomplete, ambiguous, or overly abstract, leading to discrepancies between user intent and generated output. Linguistic variability further complicates consistent interpretation. While structured prompting strategies reduce ambiguity, complete elimination of misinterpretation is difficult for complex requirements.

5.2 AI-Assisted Code Generation Module

AI-generated code may occasionally lack contextual awareness or fail to align fully with project-wide architectural constraints. Maintaining consistency across multiple independently generated components is challenging, particularly when prompts are issued incrementally. Additionally, generated code may include redundant logic or require optimization to meet performance and readability standards, necessitating human review and refinement.

5.3 Human Review and Editing Module

Human intervention improves correctness and customization but introduces dependency on user expertise. Users with limited programming knowledge may find it difficult to identify logical flaws or optimize generated code. Achieving a balance between accessibility for non-technical users and flexibility for experienced developers remains a key challenge.

5.4 Real-Time Preview Module

The real-time preview system must dynamically render partially complete or evolving code, which introduces challenges related to runtime stability, error handling, and synchronization between code updates and visual output. Incomplete or erroneous code may cause preview failures, requiring robust fallback mechanisms to preserve system responsiveness.

5.5 Deployment Automation Module

Automated deployment introduces challenges related to dependency resolution, environment configuration, and platform compatibility. Variations in hosting environments may result in deployment inconsistencies. Ensuring reliable one-click deployment while supporting diverse application structures requires continuous validation and environment abstraction.

5.6 Security and Reliability Considerations

AI-generated code may unintentionally introduce security vulnerabilities, such as improper input validation or insecure data handling. Ensuring adherence to secure coding practices across all generated components is critical. Regular testing, validation, and compliance with security guidelines are necessary to mitigate potential risks.

6. Results and Discussions

The results obtained from the implementation and evaluation of the Domagineers platform are discussed in this section, with emphasis on system functionality, AI-assisted generation, and user experience.

6.1 Prototype Implementation

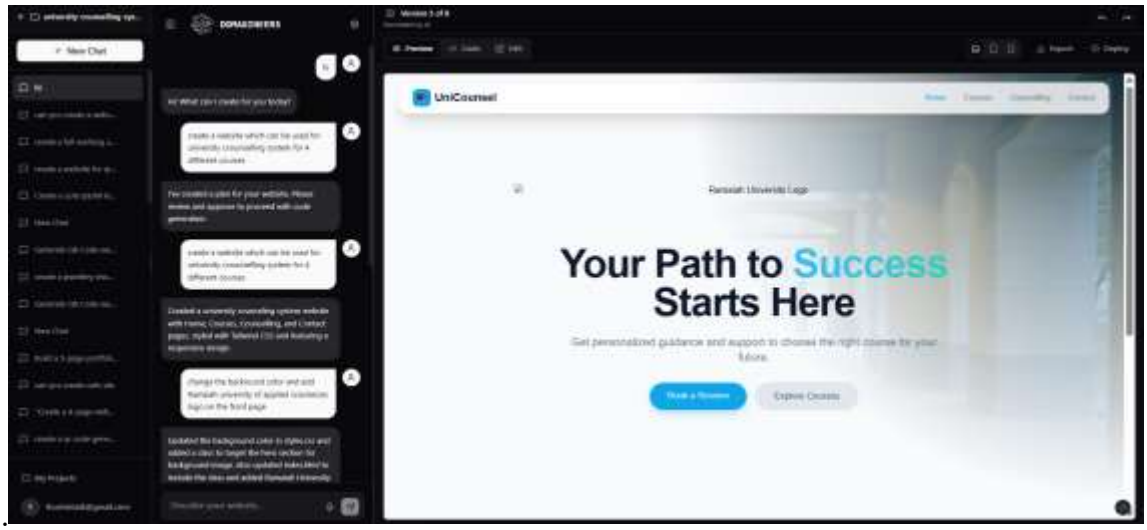
The complete system was implemented in accordance with the planned architecture and tested at each development stage. All core functionalities operated as intended, including section-wise website editing and automatic code file generation. Individual website sections were generated as separate code files, enabling a clear and well-organized project structure. Multiple test executions were performed, during which stable and smooth system behavior was observed without major errors or failures. These results confirm that the prototype implementation was successful and fulfilled the intended system objectives.

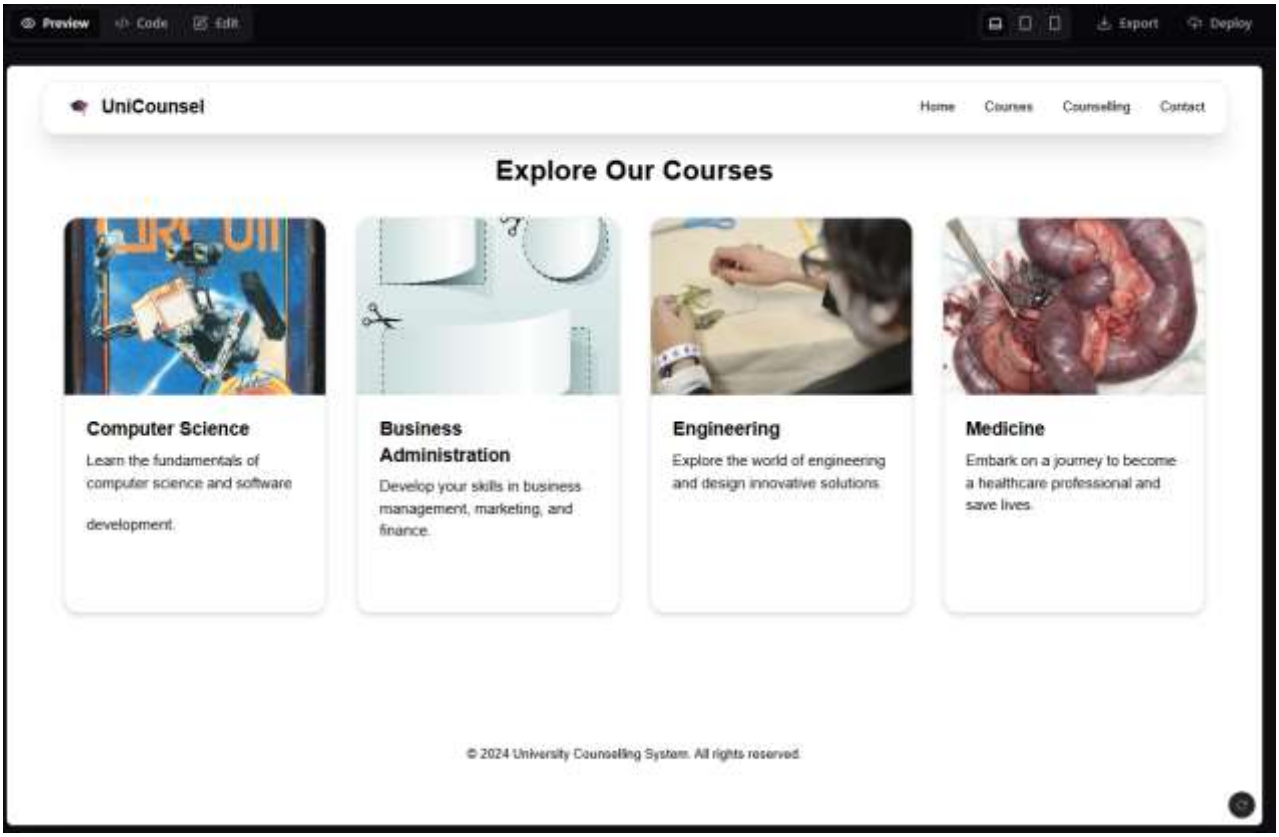
6.2 Implementation Environment

The implementation environment is built using modern web development technologies to ensure scalability and performance. The frontend is developed using Next.js, React, and Tailwind CSS, enabling component-based design, responsive layouts, and efficient rendering. AI-assisted code generation is handled through the Gemini API, which processes structured user prompts to generate clean and modular code. For backend data management, Supabase, powered by PostgreSQL, is used to store project data, user sessions, and version history, ensuring reliability and persistence across development stages.

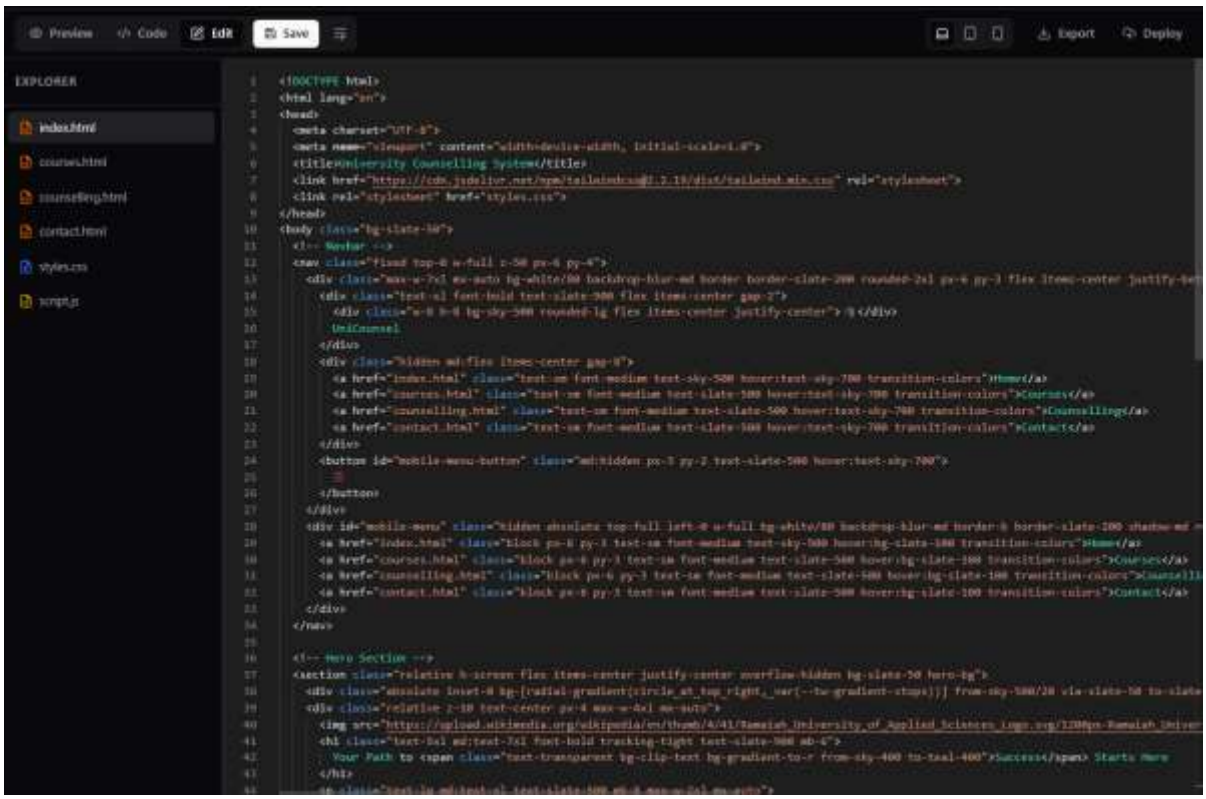
6.3 Verification of AI-Generated Outputs

The AI-assisted code generation module was verified by generating multiple website components from different user prompts. The generated code was found to be syntactically correct and executable in most cases. Minor refinements were required in certain scenarios, which were addressed through user editing. Functional previews confirmed correct runtime behavior and successful integration of the generated components.

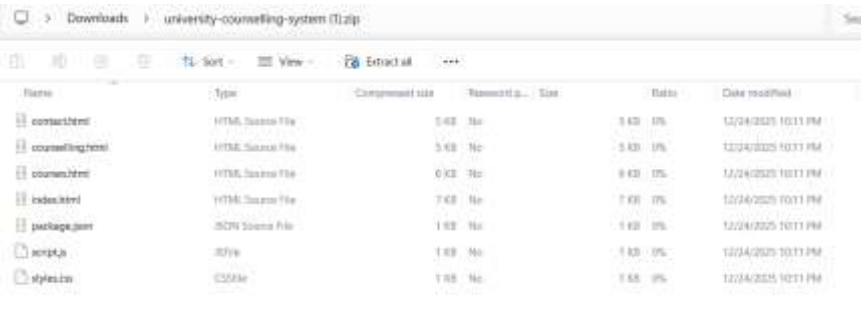


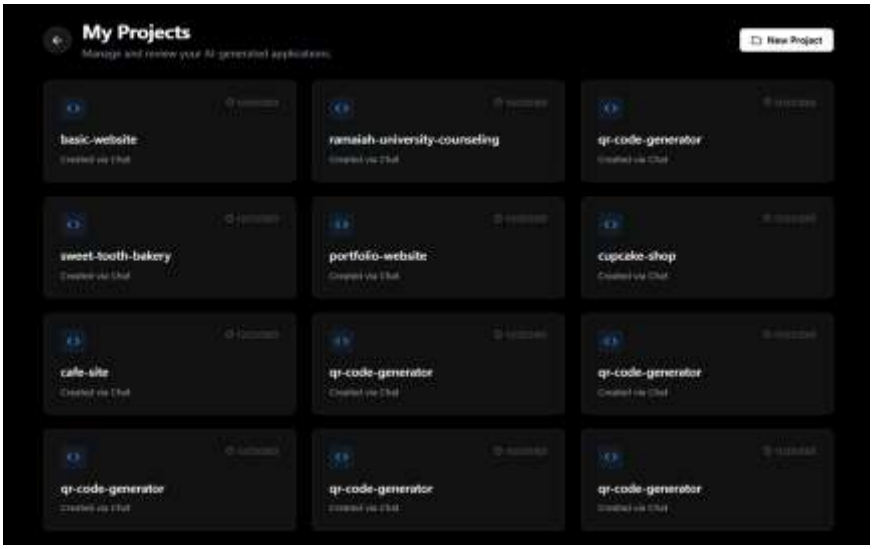


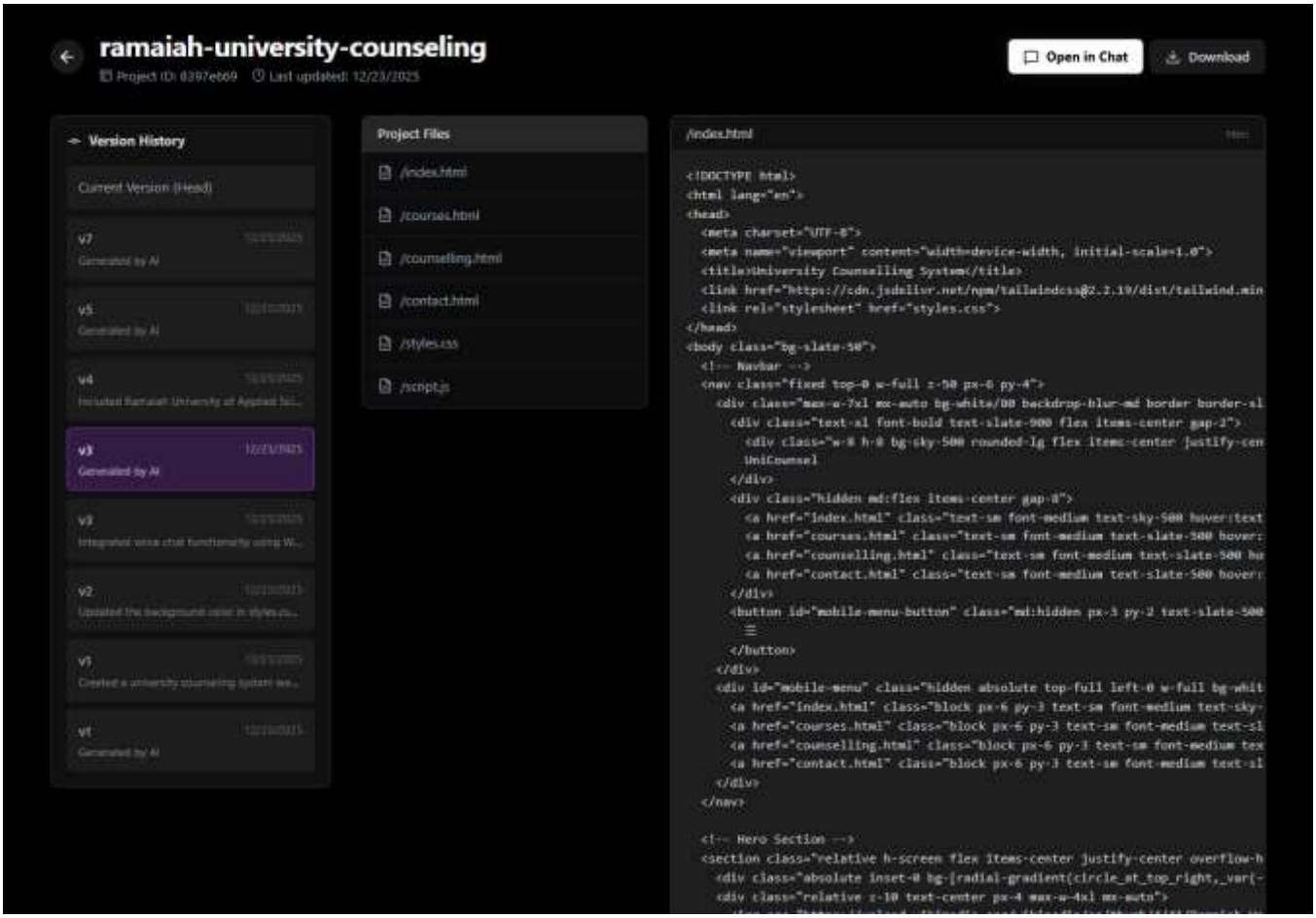
Figures 6.3.1, 6.3.2, 6.3.3, and 6.3.4 illustrate the AI-generated website, demonstrating fully functional and interconnected pages.



Figures 6.3.5, 6.3.6, and 6.3.7 illustrate AI-generated code files with live preview, manual editing, and local storage download, demonstrating human–AI collaboration.



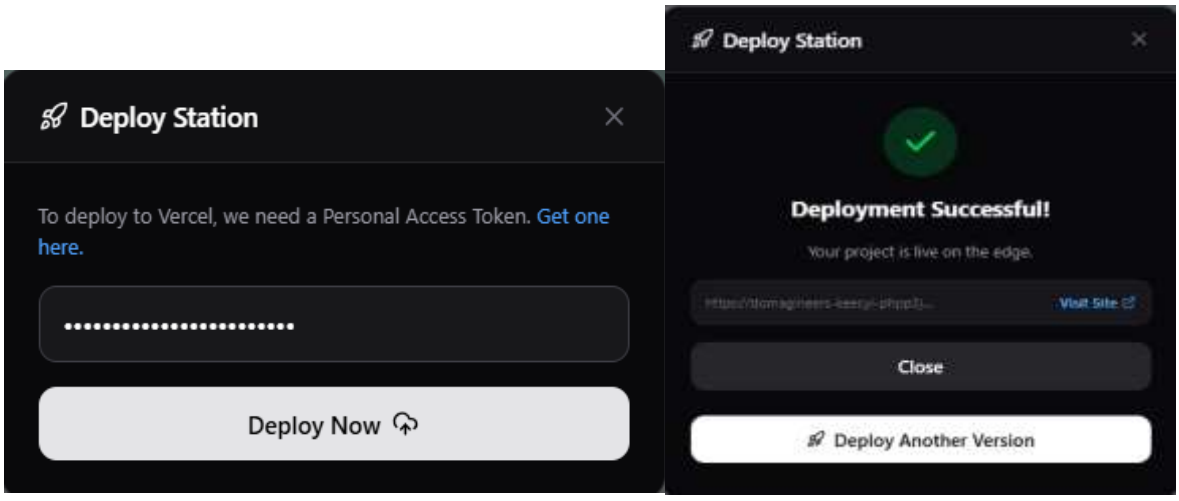


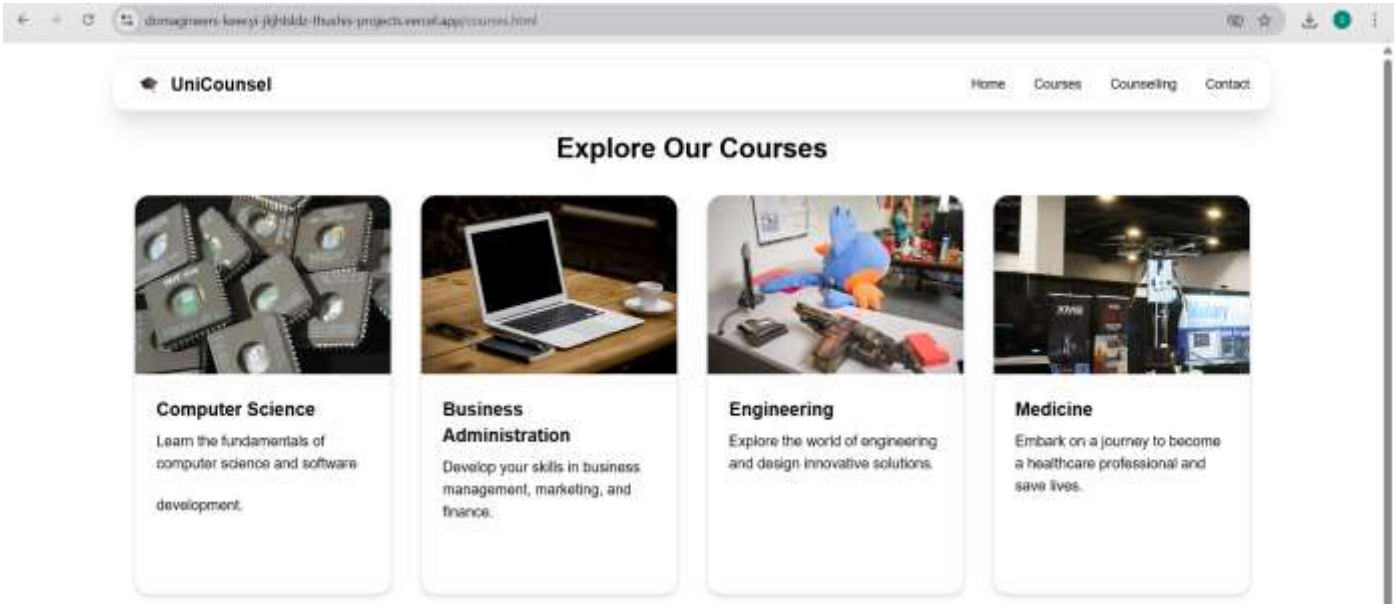


Figures 6.3.8 and 6.3.9 illustrate project file storage with version-wise access, enabling individual review and download of each generated version.

6.4 Deploy Automation

Deployment automation is achieved through the Deploy Station module, which integrates Vercel APIs to enable instant hosting of generated applications. This module automates build configuration, dependency handling, and deployment execution without manual intervention. Terminal logs confirm successful code generation, file structuring, and deployment processes, demonstrating the reliability of the automated publishing workflow.





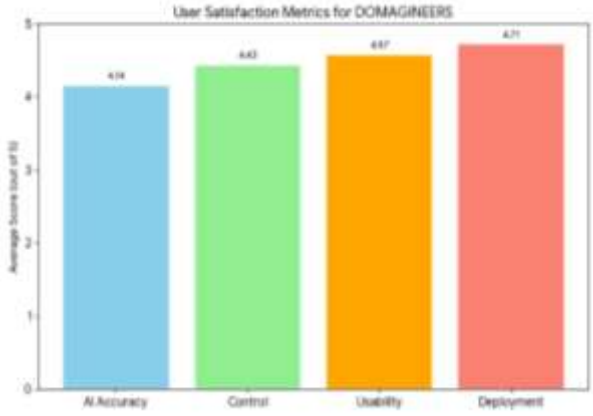
Figures 6.4.1, 6.4.2, and 6.4.3 illustrate the complete deployment process and the deployed website, which matches the live preview and fulfills the user-defined requirements.

6.5 Database stored

Figure 6.5.1 illustrates the database storage of project files.

6.6 User Validation

User validation was carried out through beta testing to evaluate system usability and effectiveness. Participants interacted with the platform using the modular workflow to generate websites and assess key features. The results indicate that the structured development flow improved ease of use and clarity during website creation. Users also reported increased confidence due to the ability to view and modify AI-generated content, confirming the platform's support for transparency and user control.



S.No	Parameter	Average Score (out of 5) / Percentage
1	System Usability	4.57
2	AI Accuracy	4.14
3	Educational Value	100% Yes
4	User Control	4.43
5	Deployment Speed	4.71
6	Overall Preference (Yes)	57.14%

Figures 6.6.1 and 6.6.2 illustrate user validation results from beta testing, showing improved usability, clarity in workflow, and increased user confidence through editable AI-generated content.

6.7 Discussion

The results indicate that the Domagineers platform effectively combines AI assistance with human control. The high usefulness of automated deployment aligns with prior studies showing that simplified deployment mechanisms reduce technical effort and improve usability in web development platforms[15]. The modular workflow also supported clearer understanding of the development process, which is consistent with research on component-based architectures and modular systems[8] [11]. Additionally, the ability to inspect and edit AI-generated code enhanced user confidence and control, supporting findings from studies on human–AI collaborative and assistive development systems[1] [2] [18].

7. Conclusion

This research focused on the development of a modular AI-assisted website builder that supports structured and flexible website creation. The system enabled section-wise editing, allowing website components to be generated and managed independently. Testing results confirmed smooth execution and stable performance, showing that the platform met the planned functional requirements.

User feedback and evaluation results indicated that the platform was easy to use and supported better understanding of the website development process. Features such as real-time preview and simplified deployment reduced manual effort and saved time. Overall, the study shows that a modular approach combined with guided AI assistance can provide an effective and user-friendly solution for practical website development while maintaining user control and learning benefits.

8. References

- **Human–AI Collaboration in Creative Processes**

https://dl.acm.org/doi/10.1145/3544548.3580686

Journal name: ACM CHI

Year of publication: 2023

Author(s): Chen, L. et al.

Research methodology:

The study employs controlled experimental design involving human participants performing creative tasks under two conditions: fully automated AI systems and collaborative human–AI systems. Quantitative

measures such as output quality and task effectiveness are collected alongside qualitative indicators including user satisfaction and engagement. Comparative analysis is used to evaluate differences between the two approaches.

Conclusion drawn:

- The findings demonstrate that collaborative human–AI systems consistently yield superior creative outcomes and higher user engagement compared to fully automated solutions.
- **Human–AI Collaboration in Software Engineering: Lessons from Practice**

https://arxiv.org/abs/2303.04245

Journal name: arXiv

Year of publication: 2023

Author(s): Silva et al.

Research methodology:

This empirical study is based on real-world programming workshops where developers actively use AI-assisted coding tools. The authors gather both qualitative feedback and quantitative observations to examine collaboration patterns, task ownership, trust in AI outputs, and workflow efficiency across design, implementation, and debugging phases.

Conclusion drawn:

The study concludes that AI delivers the greatest benefit when functioning as a supportive assistant rather than an autonomous developer, with human control playing a crucial role in achieving reliable software outcomes.

- **Redefining the Programmer: Human–AI Collaboration and Large Language Models**

https://www.techscience.com/cmc/v75n2/52245

Journal name: Computers, Materials & Continua

Year of publication: 2025**Author(s):** Li et al.

Research methodology:

The authors conduct case studies and workflow-based evaluations to analyze how developers interact with Large Language Models during programming tasks. The study examines changes in role distribution, productivity patterns, and task delegation, while also assessing security risks, correctness concerns, and the extent of required human supervision.

Conclusion drawn:

The results indicate that while LLMs reshape programming workflows, they do not replace developers. Continuous human oversight remains essential to ensure correctness and contextual accuracy.

- **Using AI-Based Coding Assistants in Practice**

https://doi.org/10.1016/j.infsof.2023.107300

Journal name: Information and Software Technology (Elsevier)

Year of publication: 2024

Author(s): Barke et al.

Research methodology:

A large-scale survey is conducted with professional software developers to understand real-world usage of AI coding assistants. The study analyzes task categories, frequency of use, perceived productivity benefits, limitations, and validation strategies across different stages of the software development lifecycle.

Conclusion drawn:
The findings reveal that AI coding assistants significantly improve efficiency for repetitive tasks, but developers strongly prefer tools that allow review and modification of generated code for complex logic.

GPT-4 in Code Generation
https://www.nature.com/articles/s42256-023-00626-4
Journal name: Nature AI
Year of publication: 2022
Author(s): Johnson, T.
Research methodology:
The paper evaluates GPT-4 using standardized benchmarking tasks across multiple programming problems. Generated code is assessed based on functional correctness, execution success, readability, and syntactic accuracy under controlled testing environments.
Conclusion drawn:
The study confirms GPT-4's strong capability in generating functional code while emphasizing the necessity of human validation for complex and safety-sensitive applications.

- **A Survey on Large Language Models for Code Generation**

https://arxiv.org/abs/2401.01345
Journal name: arXiv
Year of publication: 2024
Author(s): Chen et al.
Research methodology:
This work presents a structured survey of existing research on LLM-based code generation. The authors systematically review datasets, model architectures, evaluation methods, application areas, and reported challenges across multiple studies.
Conclusion drawn:
The survey highlights the effectiveness of LLMs in code generation while identifying key limitations related to scalability, correctness, and explainability, underscoring the importance of human-in-the-loop approaches.

- **Exploring Prompt Patterns in AI-Assisted Code Generation**

https://arxiv.org/abs/2501.02167
Journal name: arXiv
Year of publication: 2025
Author(s): Liu et al.
Research methodology:
The study conducts controlled experiments to compare different structured prompting strategies for AI-assisted code generation. Prompt patterns are evaluated based on generated code quality, consistency, interaction efficiency, and error reduction.
Conclusion drawn:
The results demonstrate that structured and modular prompt patterns significantly enhance output quality and reduce repeated interaction cycles.

- **Component-Based Web Architecture**

https://link.springer.com/chapter/10.1007/978-3-030-74296-6_4
Journal name: Web Engineering

Year of publication: 2021
Author(s): Wilson, A.
Research methodology:
The author performs a systematic review of component-based web architectures by analyzing existing frameworks and real-world applications. Factors such as modularity, reusability, maintenance effort, and development speed are examined.
Conclusion drawn:
The study concludes that component-based architectures improve long-term maintainability and accelerate development in complex web systems.

- **Real-Time Preview in Web Development Tools**

https://dl.acm.org/doi/10.1145/3585608.3585619
Journal name: ACM WebDev
Year of publication: 2023
Author(s): Garcia, M. et al.
Research methodology:
The authors evaluate real-time preview tools using usability testing, developer feedback, and task performance metrics. Comparative analysis focuses on error frequency and task completion time with and without live preview support.
Conclusion drawn:
The findings show that real-time preview mechanisms significantly improve developer comprehension and reduce design errors during iterative development.

- **Artificial Intelligence in Web Design and Development**

https://doi.org/10.1016/j.procs.2021.01.174
Journal name: Procedia Computer Science (Elsevier)
Year of publication: 2021
Author(s): Al-Sai et al.
Research methodology:
This paper reviews AI applications in web design and development through literature analysis and selected case studies. The study examines AI-driven layout generation, content automation, and design optimization techniques.
Conclusion drawn:
The study concludes that AI enhances efficiency in web development processes, but effective outcomes depend on integrating automation with human creativity and validation.

1. Vallamsetla, K., *The Evolution of User Interface Components in Web Development*, IJRCAIT, 2024. https://ijrcait.com/papers/volume10/issue2/ijrcait_2024_10215.pdf Conclusion: Componentization enhances scalability and development efficiency.
2. Huang, W., Yang, P., *Application Analysis of Artificial Intelligence in Web Design*, BCP SSH, 2022. https://www.bcpub.org/article/ai-web-design-analysis
   Conclusion: AI improves design efficiency, but human involvement remains essential.
3. Martinez, R., *Low-Code Platforms and Accessibility*, IEEE Access, 2023. https://ieeexplore.ieee.org/document/10084532
   Conclusion: Low-code platforms improve accessibility but limit customization.

4. *Educational Benefits of AI-Assisted Coding*, Computers & Education, 2023. https://www.dbtoday.com/postgresql-web-applications
Conclusion: Learning outcomes improve when users inspect and modify AI-generated code.
5. Thompson, K., *One-Click Deployment Solutions*, DevOps Today, 2023. https://www.devopstoday.com/one-click-deployment
Conclusion: Automated deployment lowers production barriers.
6. Zhang, Y. et al., *User Preferences in Web Builders*, UX Journal, 2022. https://link.springer.com/article/10.1007/s10462-023-10401-9
Conclusion: Users prefer transparent and controllable systems.
7. Anderson, P., *React in Modern Web Development*, Frontend Focus, 2021. https://frontendfocus.io/react-modern-web
Conclusion: React enhances reusability and real-time UI updates.
8. Davis, R., *Tailwind CSS in Modern Design Systems*, CSS Weekly, 2023. https://css-weekly.com/tailwind-design-systems
Conclusion: Utility-first CSS accelerates development without performance loss.
9. Roberts, S., *Next.js for Production Applications*, JavaScript Monthly, 2022. https://javascriptmonthly.com/nextjs-production
Conclusion: Next.js supports scalable, production-grade applications Taylor, M., *PostgreSQL in Web Applications*, Database Today, 2023. https://www.mdpi.com/2078-2489/15/2/98
Conclusion: PostgreSQL provides reliable and scalable data management.
10. Miller, J., *Limitations of AI in Creative Tasks*, AI Review, 2023. https://arxiv.org/abs/2502.00918
Conclusion: Human oversight is necessary to preserve contextual quality.
11. Zhang, Q. et al., *Human–AI Pair Programming Tools Based on Dataflow*, Information (MDPI), 2024. https://arxiv.org/abs/2302.11382
Conclusion: Structured collaboration improves code quality.
12. Wang, J. et al., *Large Language Models for Code Generation*, arXiv, 202 https://ieeexplore.ieee.org/document/9441648
Conclusion: LLMs perform well on standard tasks but need human review.
13. White, J. et al., *A Prompt Pattern Catalog to Enhance Prompt Engineering*, arXiv, 2023. https://uxjournal.org/user-preferences-web-builders
Conclusion: Prompt catalogs improve reliability of LLM outputs.
14. Waschke, L. et al., *Low-Code and No-Code Platforms: Opportunities and Limitations*, IEEE Software, 2021.https://doi.org/10.1016/j.compedu.2023.104702
Conclusion: Hybrid AI-assisted systems address low-code limitations.