International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(1s) 2026 pp. 1-17

https://doi.org/10.70917/ijcisim-2026-1996
Article

An Integrated Framework for Webpage Designing: A Comprehensive Study on Literature Review and Hybrid VAE-GAN

Rohit Yadav[1], Reena Hooda[2], Manish Gupta[3]

[1,2]Department of Computer Science and Engineering, IGU, Meerpur, Rewari
[3]School of Computer Science, Faculty of Engg. & IT, University of Technology, Sydney, Australia.
Emails: [1]rk6yadav@gmail.com; [2]reena.cse@igu.ac.in;[3]manish.gupta@uts.edu.ac;

Abstract: The digital world is rapidly expanding with over 1.1 billion active websites. It creates a critical bottleneck for web development and remains intensive for non-technical users. The existing Artificial Intelligence (AI) and Machine Learning (ML) solutions offer automation but they frequently suffer from code hallucination and structural inconsistency. This research proposes an integrated hybrid framework design to bridge this gap by a visual perception with linguistic generation. The framework uses a multi-phase pipeline where a Variational Autoencoder (VAE) extracts UI features into a 512-dimensional probabilistic latent space (). This latent space is shared with a Generative Adversarial Network (GAN) Discriminator for visual realism. A transformer-based decoder performs the synthesis to fine-tune using GPT-2 tokenizer. This translates the latent design structure into sequential HTML code. The transformer can generate scattered tokens, causal making and adaptive repetition penalty will ensure the syntactic integrity for token generation for optimisation layer. The model is validated over a subset of 50,000 image-code pair entries sliced from the Web2Code dataset on the NVIDIA 4070 GPU over 8 GB VRAM. The 50-epoch model training experimental results demonstrate a significant and notable achievement of 100% syntax validity initialised on epoch 20 and remains consistent till the end with 0.2412 code generation loss and 0.2839 VAE structural loss. A comparative analysis shows that the framework achieves a superior design quality score of 9.2/10 and code quality of 9.4/10. The performance metrics further reveal a reduced load time of 1.4 seconds. Ultimately, the research provides a robust and scalable solution for automated webpage generation and moving toward an intelligent ecosystem that can combine human creativity with computational precision.
Keywords: - Artificial Intelligence, Machine Learning, Web Design, Automated Design Generation, User Interface, Deep Learning, Generative Models

1. Introduction

The internet is expanding every day. Nowadays, more than 1.1 billion websites are currently active and 252000 are launching each day. The traditional approaches are reliable but very time-consuming in designing, personalising, scaling and achieving an efficient user experience. Creating a website without technical expertise is more challenging [1]. That is why an automated system is needed for it. AI & ML offer automation for the complex design process. The various design challenges, like design to code generation, layout preparation and personalised user experiences can be solved by automation [2]. The generative models such as Generative Adversarial Networks (GANs) and Transformer models can understand repetitive complex design structures. These are also able to generate website code based on requirements [3]. Generative AI and transformer models are able to enhance technical and academic work. But there are a lot of challenges related to creative approach, accuracy and reliability of AI-generated content [4]. Well-structured and knowledge-based methods can be integrated with Large Language Models (LLMs) to address these challenges [5]. In this research paper, the author covers two key objectives: One is to review existing literature on webpage design models, analyse traditional approaches, AI-based solutions & hybrid frameworks [6]. Another is to integrate web designing with AI/ML techniques, examine current methodologies and tools to automate the webpage creation [7]. An empirical evaluation compares various web design approaches such

as traditional, template-based, AI-generated & proposed framework approaches. Introducing a pseudocode for key algorithms in the proposed framework [8].

This paper is structured in a flow to represent the section 2 provides a literature review & identifies a specific research gap. Section 3 introduced the research methodology and datasets. Section 4 presents the proposed hybrid architecture & algorithmic implementation. Section 5 analyses the results and concludes the study in Section 6 with outlines for future work.

2. Literature Review

This section covers the traditional approach of web design and the integration of AI & ML in web design. The revolution of web design approach varies at different times, it started from static HTML-based and moved towards dynamic and responsive frameworks [9]. In the early stage of web development relied on manual coding using HTML, CSS, and JavaScript. Later on template-based web development started [10]. According to studies traditional manual design approaches are also limited to produce high-quality code on 8.0/10. It also required an average of 40 hours for development and struggled to personalise [11]. The evolution of responsive design frameworks such as Bootstrap, manages cross-device compatibility. But it also needs technical expertise [12]. To implement a robust framework that is able to generate complex, multi-modal data from initial inputs such as predicting a customer's future behaviour from the current historical data of user [13]. While templates help to increase the speed of designing but lacks for intelligence to adapt the user behavior to lead the integration of ML.

2.1. AI in Web Development

The integration of web development flow with AI & ML is able to enhance the compatibility of work from simple automation to sophisticated design of websites [14]. The initial AI application focused on automating repetitive tasks such as coding and syntax checking. There are several key areas in modern AI web development such as code generation, automating the layout design and content generation. Various tools using LLM models to generate code by a prompt written in natural language. The LLM models are also able to generate the content of a website based on Natural Language Processing (NLP) and ML algorithms. These models are trained on a huge amount of code repositories to understand patterns and context of code in programming [15]. Along with the code generation AI-based tools like Wix ADI and Relume are able to generate layouts according to given natural language text prompts. These tools use ML algorithms to analyse user preferences to generate personalised designs [16]. The basic ML improves the UX but the generation of new and unique layouts requires advanced generative architectures such as GANs.

2.2. Machine Learning (ML) Applications in UI/UX

User Interface and User Experience (UI/UX) design is one of the focusable area for the research to automate with ML [17]. There are various key factors to automate using ML such as personalised UI generation, predictive user behaviour and automated testing. The current research elevates the automation in UI design using machine learning and deep learning techniques. Some models such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are able to create personalised and adaptive UI layouts to enhance the UX. Various ML models have the capability to analyse the patterns of user interaction and also able to make predictions for user behaviour based on the analysis [18]. Along with the personalised UI generation and predicting the user behaviour, testing and optimisation can also be automated using ML models. Tools such as Applitools & automated UI testing frameworks can test the user interface automatically. These tools use machine learning and computer vision to identify components and inconsistencies to optimise the user interface [19]. GAN models are powerful but lack the sequential logic of Transformers. That is why an approach is required.

2.3. Automated Design Generation

The automated design generation can be enhanced with the help of various machine learning & deep learning techniques. Some models such as Generative Adversarial Networks (GANs), Transformer models and computer vision are able to lead the automation workflow for design generation. GAN models are emerging for analysing and generating the UI layouts because of their analytical learning capability from large datasets. These models can learn and identify existing patterns in the datasets. According to recent studies GANs are able to capture relationships among UI elements and generate designs [20]. Transformer models can be used to generate code and an interface. These models can make a hierarchical structure to represent complete UI pages [21]. Various deep learning models can analyse the screenshots and able to generate the code by integrating with computer vision [22]. Ultimately, the

objective of the enhancements is to create an easy-to-use system that manages complex design workflow and provides a natural language-based interaction [23]. It is necessary to refine the diffusion models to enhance stability and Generative Adversarial Networks (GANs) to interpret the internal working of other neural networks (NN) [24], [25].

2.4 Comparative Study of Existing Literature

Table 1 provides a comparative analysis of existing research works in the field of AI-driven website & UI design. It highlights their methodology, focus areas & identified limitations.

Table 1: Comparative Analysis of research work

Reference	Author(s) & Year	Methodology / Techniques	Focus Area	Key Contributions / Findings
[1]	Stige et al. (2023)	Systematic Literature Review	AI & UX Design	Identified a future research agenda focusing on human-AI collaboration in design.
[13]	Mancisidor et al. (2022)	Deep Generative Models	User Behavior Prediction	Successfully generated synthetic customer behavior data for predictive modeling.
[25]	Guna et al. (2024)	Conditional GANs	CNN Prediction Interpretation	Enhanced the interpretability of neural network outputs in visual tasks.
[15]	Wan et al. (2025)	Divide-and-Conquer / Deep Learning	UI Code from Screenshots	Automatically generates UI code from visual inputs with higher accuracy than previous models.
[26]	Zhan et al. (2024)	Deep Learning	Personalized UI Layout	Proposed an adaptive interface design approach to enhance user experience through DL.
[27]	Duan et al. (2024)	Tree Algorithm / Deep Learning	Aesthetic UI Design	Balanced efficiency and aesthetics in interface generation using hierarchical algorithms.
[28]	Svyatkovskiy et al. (2020)	Transformers (IntelliCode)	Code Generation	Demonstrated high-efficiency code completion and generation using Transformer architectures.
[29]	Delitzas et al. (2024)	Deep Learning (Calista)	Aesthetic Evaluation	Developed a system for understanding and quantitatively evaluating website aesthetics.

[30]	Costa et al. (2024)	AI-Driven Design	Web Application UI	Proposed a workflow for moving towards fully AI-driven UI design for modern web apps.

In various literature works, authors have tried to bridge the various weaknesses in merging web development with AI & ML. The integration of AI tools and frameworks is still emerging with various challenges such as limited comprehensive frameworks, scalability issues, user experience gaps and research fragmentation. The current solutions address some specific part of web design but not the integration of multiple AI & ML techniques. The existing system is still facing challenges in multi-page layouts and advanced workflows of real-world resources integration [31]. Most of the automated systems are facing challenges during high-quality code generation and also in maintaining it to professional standards. The integration of web design and AI/ML still lacks a systematic literature review to consolidate findings [32]. Current research on AI-based web design focuses on specific features rather than comprehensive and integrated frameworks. Resultantly existing systems struggle with complex and multi-page applications. Many AI-based tools can able to generate code but still lack optimisation & consistent evaluation benchmarking. These systems still need proper technical expertise to implement & maintain [33]. This research fills the gap on the Tags Looping problem, Visual-Code Disconnect and Syntactic Integrity in existing models often produce hallucinated or broken tags and contributes a systematic analysis of research papers and industry reports on AI & ML integration with web design [34]. Here author highlights a hybrid approach to combine GANs, Transformers & deep learning based automated web design [35].

3. Methodology

3.1 Research Design, Data Acquisition And Preprocessing

This research follows a mixed-method approach for a systematic literature review and an academic empirical analysis. Here, the initial phases involved a comprehensive literature review of AI & ML in web design by reviewing research papers and industry reports published from 2020 to 2025 [36]. Later on moving toward the core development phase built upon the Web2Code dataset. It is a large-scale repository of webpage screenshots & their respective HTML code.

3.1.1 Dataset Slicing & Sampling

To ensure the computational efficiency & diverse layout representation a sub-dataset of 50,000 image-code pairs was extracted using random sampling techniques from the primary dataset. This slicing strategy was implemented to maintain a manageable training timeframe on the NVIDIA 4070 GPU with 8 GB VRAM. During the data set slighting it is also ensuring that the model encounters a wide variety of UI components are included into subset of the primary dataset.

3.1.2 Data Preprocessing Pipeline

A pipeline is prepared to process the data and make it usable for model training within the following steps:

1. **Image Transformation:** All web screenshots are resized into 256 * 256 pixels and used with a normalised mean & 0.5 standard deviation to ensure the uniform input.
2. **Code Tokenisation:** The corresponding HTML code is processed using GPT-2 tokenizer to enhance the complex learning of syntactic relationships between design elements & code structure.

Evaluation of an integrated framework combined with different AI & ML techniques for automated webpage generation [37]. All these phases of the research are done on the data collected from the various sources & methodologies such as IEEE Xplore, ACM Digital Library, ScienceDirect for literature sources [38]. After literature studies conducting a comparative analysis of existing approaches such as traditional design, template-based systems & AI-powered solutions [39]. Quantitative data on design quality, development time, user satisfaction, responsiveness and code quality across different approaches [40] and evaluation of data collected from 200 participants across different demographics to analyse user satisfaction & interface usability.

3.2 Framework

The proposed model is an integrated framework that integrates the different AI & ML techniques to overcome the weaknesses of existing tools. The components of the framework are a generative design engine for visual layout generation using GAN models, a code generation module for automated code creation by transformer models, a personalisation engine for user-specific customisation by using deep learning and also applies reinforcement learning for continuous improvement in the optimisation layer [26]. The proposed model uses a modular architecture. It enhances the independent development & testing of components. It also preserves seamless integration & overall framework integrity [41]. The architecture consists of three primary technical layers in a pipeline:

1. **Generative Design Engine (VAE-GAN):** This model is a combination of Variational Autoencoder to prepare the UI features for a structured latent space & Generative Adversarial Network discriminator used to generate layouts with visual fidelity & realism.
2. **Transformer Code Module:** It is a translation layer to generate the sequential HTML code by taking the latent representation from the design engine.
3. **Optimiser Layer:** In this layer reinforcement learning & adaptive repetition penalty is applied to prevent common generative errors [26], [41].

3.3 Algorithm Development

The proposed framework is prepared upon several algorithms. Each algorithm is developed to address the specific stage of the automated web design process. The various frameworks are included such as the Hybrid VAE-GAN algorithm used for personalised UI layout generation including L1 Reconstruction Loss, KL Divergence, and Adversarial Loss to make a balance between design accuracy and creative variety. Transformer-based Code generation for automated HTML, CSS and JavaScript generation [27]. Beam Search Decoding with 3 to 5 width and causal masking is used to predict code tokens sequentially to maintain syntactic integrity. Multi-objective Optimisation for balancing design quality, performance and user preferences [42]. It is a post-processing step to check unclosed tags & structural inconsistency. An Adaptive Learning Algorithm is crucial for enabling continuous improvement by systematically processing & learning from user feedback.

3.4 Evaluation Metrics

A dual-metric approach is applied to measure the technical accuracy & human satisfaction to validate the framework.

Quantitative Metrics: The technical validation is quantified by measuring the BLEU score, syntax validation percentage & Task Completion Rate. The system performance is also measured by load time and development time.

Qualitative Metrics: This framework is prepared to involve distinct performance levels & design quality. These are grouped into two metrics such as quantitative & qualitative. Distinct dimensions of performance for quantitative are the Design Quality Score, User Satisfaction Score, Responsiveness Score, Code Quality Score from 0 to 10 scale and Development Time in hours. Other than these the qualitative metrics performance dimensions are user experience assessment, accessibility compliance, cross-browser compatibility and SEO performance.

4. Proposed Model Architecture

4.1. Integrated System Overview

The proposed architecture is designed as a unified pipeline. It follows a multi-phase hybrid approach. In this approach visual perception and code generation are linked with a mathematical space [43], [28]. Figure 1 illustrates the continuous end-to-end pipeline workflow of the framework. The process initiates with a ResNet-based CNN model for visual feature extraction. It is mapped into a 512-dimensional probabilistic latent space. The latent vector is commonly used in GAN for visual refinement and the transformer for sequential code generation. Finally, the output is refined by the optimiser to ensure syntactical integration and professional performance.

The detailed workflow of each phase of architecture is described as follows:

1. **VAE Encoder:** The process begins with the Variational Autoencoder its primary work is feature extraction. It takes the preprocessed 256 * 256 webpage screenshots and compress it into low dimension latent space

(). The vector represents the core DNA of the webpage. It captures the layout, colour schemes and element placement. Here the system ensures that the code generation is based on abstract design patterns instead of exact pixel matching with the help of vector.

2. **GAN discriminator:** VAE phase tackle with the structure of webpage whereas the Generative Adversarial Network discriminator ensures the quality of it. This phase evaluates the generated design against the ground-truth sample space taken from the Web2Code dataset. It gives a Adversary Loss signal when the generated layout looks unrealistic or unstructured. In this situation the discriminator penalises the model. This penalisation technique ensures the output remains aesthetically professional and well structured.
3. **Transformer decoder:** the final page is the synthesis phase or transformer-based code generation modules. At this position the system acts as a translator. It receives a vector and treat it as prompt content. It predicts HTML code tokens sequentially. Every time written by Transformers is mathematically tied to the visual features identified in the perception phase. It ensures that the generated code properly renders according to the visual layout.
4. **Optimiser layer:** The last phase of the pipeline is optimization layer that monitors the output of the transformer and verifies the efficiency of the output. Reinforcement learning is applied for penalty and syntax validation. It also prevents the looping bugs generated by Transformers and ensures the HTML document is ready for the browser without manual debugging.

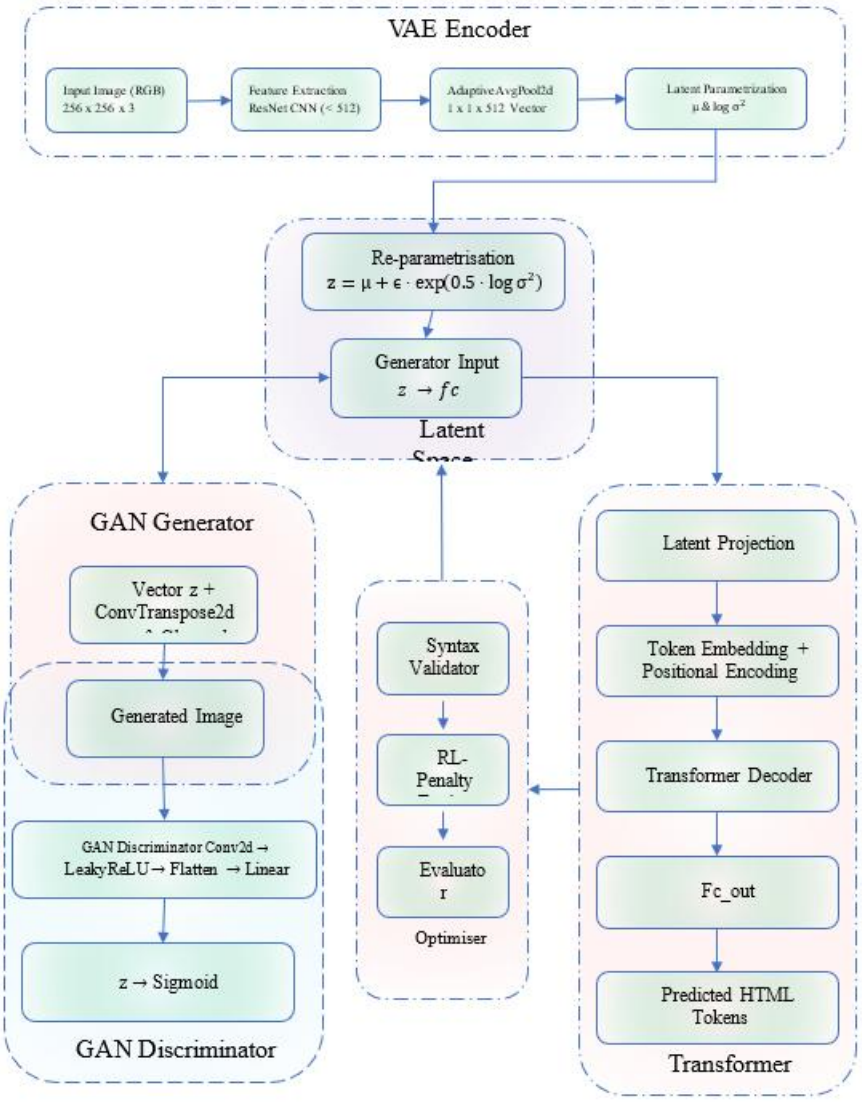


Figure 1: Hybrid System Architecture Diagram

4.2. The Hybrid VAE-GAN design Engine

This integration provides a robust and holistic solution for automated web design [44]. Here, a transition from high-level architecture to specific deep learning models gives the ability to handle visual layout generation. The VAE & GAN models are the core of the framework and work together to create a robust latent representation of web design [29]. The Variational Autoencoder is used as a perceptual component for structural mapping. It maps the input UI image to a probabilistic latent space. It processes the webpage image metadata to produce two vectors: a mean (μ) and log-variance ($\log \sigma^2$). It allows the back propagation in model training with a reparameterization technique and sampling a latent vector $z = \mu + \epsilon . \sigma$.

Algorithm 1: Hybrid VAE-GAN Training Logic

This algorithm works on the visual perception layer in the framework. It optimised the model to compress the UI layout in the standard latent space and ensured the output.

Require: Pre-processed Dataset D, Latent Dimension d weights $\lambda_1, \lambda_2, \lambda_3$

Ensure: Optimised Encoder E, Decoder D & Discriminator Disc

1. **Initialise** Encoder E, Decoder D, and Discriminator Disc

2. **For** each epoch e $=$ 1 to max_epochs **do**:
3. **For** each batch B in D **do**:
4. $\mu, \log \sigma^2 \leftarrow E(B)$ // Generate latent distribution parameters
5. $z \leftarrow \mu + \epsilon \odot \exp(0.5 \cdot \log \sigma^2)$ where$\epsilon \sim \mathcal{N}(0, I)$ // Reparameterization trick
6. $Rec \leftarrow D(z)$ // Reconstruct layout from latent vector
7. **Compute Individual Losses:**
8. $\mathcal{L}_{rec} = |B - Rec|_1$ // Pixel-level reconstruction accuracy
9. $\mathcal{L}_{kl} = -\frac{1}{2}\sum\left(1 + \log \sigma^2 - \mu^2 - e^{\log \sigma^2}\right)$ // Latent space regularisation
10. $\mathcal{L}_{adv} = \log\left(Disc(B)\right) + \log\left(1 - Disc(Rec)\right)$ // Adversarial quality check
11. **Calculate Weighted Total Loss:**
12. $\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{kl} + \lambda_3 \mathcal{L}_{adv}$
13. **Backpropagate** $\mathcal{L}_{total}$ to update E and D.
14. **Update** Disc by minimising binary cross-entropy between real and fake samples.
15. **End For**
16. **End For**

The VAE handles the structural mapping and the GAN is responsible for virtual realism and professional refinement. A role of the discriminator is to observe the implementation and evaluate the reconstructed layout or real-world training sample data from the Web2Code dataset. It gives an adversarial loss signal (L_{adv}) to force the model to move beyond simple pixel reconstruction. The GAN ensures layout boundaries, element alignments and aesthetic violence meet high-fidelity requirements for modern web standards.

The hybrid nature of the engine gives a structural and visual aesthetic to the output. It solves a common problem where layout is mathematically accurate but visually unrealistic in automated design. Visually oriented design established a latent space through VAE-GAN integration model and uses a transformer-based module to translate these visual features into functional HTML code.

4.3 Transformer-based Code Generation

Once the Generative Design Engine establishes a visual layout in latent space, the framework is ready to move towards the code generation module. This module is responsible for the translation of visual features into functional code and makes it browser-ready HTML code.

In this phase, a transformer-based decoder architecture is used. It is fine-tuned on GPT-2 tokenizer. The input of this page is a latent vector generated by VAE-GAN models. The transformer treats the features as a contextual prompt. Later on the causal masking comes into the picture for syntactical validation of the generated code. It corresponds to the generate square subsequence mask function in the implementation to prevent the model from looking ahead and ensures a logical & top-down generation of the webpage structure. Once the code is generated, a module Beam Search Decoding is used to refine the output. The system maintains multiple candidate code strings simultaneously as a next word instead of a single word. An adaptive penalty is applied to discourage the model from falling into a repetitive loop in code generation.

Algorithm 2: Transformer-based Code Generation

This algorithm acts as a linguistic translator in the framework. It decodes latent features into sequential HTML tokens. It uses a self-attention mechanism and a heuristic search strategy to ensure visual alignment and syntactic correction

Require: Latent vector z, vocabulary V, beam width k, repetition penalty α, Max length T

Ensure: Syntactical validation of HTML code string C

1. **Initialise** sequence $S = [START\_TOKEN]$
2. **Generate** Causal Mask M of size (T, T) to ensure autoregressive consistency.
3. **For** step $t = 1$ to T **do**:
4. **Compute Attention:**
5. $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$
6. **Forward Pass:**
7. $\text{Logits}_t \leftarrow \text{Transformer}(S, z, M)$ // Conditioned on latent vector z
8. **Apply Repetition Penalty:**
9. For each token $i \in, \text{Logits}_t[i] = \text{Logits}_t[i]/\alpha$ // Where $\alpha = 1.2$
10. **Beam Search Selection:**
11. $\text{Candidates} \leftarrow$ Select top k sequences with highest cumulative probability.
12. $S \leftarrow$ Update best sequence from Candidates.
13. **If** last_token == [END_TOKEN] **break**.
14. **End For**
15. **Detokenize** S using GPT-2 tokenizer into raw string C_{raw}.
16. **Execute** Syntax Validation (check for unclosed tags and hierarchy).
17. **Return** C (Optimised functional code).

4.4 Design Optimisation Layer

Algorithm 3: Multi-objective Design Optimisation

This algorithm works as a post-generation refinement layer. It evaluates the web pages against competing objectives such as accessibility and performance to validate the final output performance.

Require: Initial Design Candidates P (from Transformer), Objective function $F = \{f_{acc}, f_{seo}, f_{perf}\}$, Weights W, Max generators G

Ensure: Optimised Design D_{opt}

1. **Initialise** population P_0 using variations of the Transformer output.
2. **For** generation $g = 1$ to G **do**:
3. **For** each individual $p \in P_g$ **do**:
4. **Calculate Fitness Score:**
5. $\Phi(p) = w_1 \cdot f_{acc}(p) + w_2 \cdot f_{seo}(p) + w_3 \cdot f_{perf}(p)$
6. **End For**
7. **Selection:** $\text{Parents} \leftarrow$ Select top N individuals based on Φ.
8. **Crossover:** $\text{Offspring} \leftarrow$ Recombine structural components of Parents.
9. **Mutation:** Apply random perturbations to CSS properties.
10. **Constraint Check:** Verify HTML syntax integrity (from Alg 2).
11. P_{g+1} Update population with Parents and Offspring.
12. **If** $\Delta\Phi <$ threshold **break** (Convergence met).
13. **End For**
14. $D_{opt} \leftarrow \text{argmax}_{p \in P}\Phi(p)$
15. **Return** D_{opt}

4.5. User Interface Generation

The UI generation process follows various approaches such as requirements analysis is needed to analyse the inputs given by the user in various forms. Based on user input data layout generation layer generates multiple layouts through the hybrid VAE-GAN model. Content population is used to generate the content like text, images, etc. for the layouts. After that the conversion of designs into functional web code will be done by a transformer-based code generation module. Once the code is ready optimisation is required to enhance the performance and user experience of the algorithms. The last phase of UI generation is to validate the cross-browser compatibility and optimise performance.

5. Results & Analysis

5.1. Experiment Setup & Training Convergence

The framework was turned on 50 folks with a batch size of 6 and a learning rate of 0.0001. The training process demonstrated a strong convergence in Figure 2. The model achieved a final main loss of 3.5582. Whereas remain at 0.2412 and the VAE reconstruction loss remains at 0.2839.

Convergence behavior shown in the training loss graph. The total main loss experienced a sharp decline before the 10th epoch. It shows that the hybrid architecture quickly identifies the core structural parameters of the Web2Code dataset.

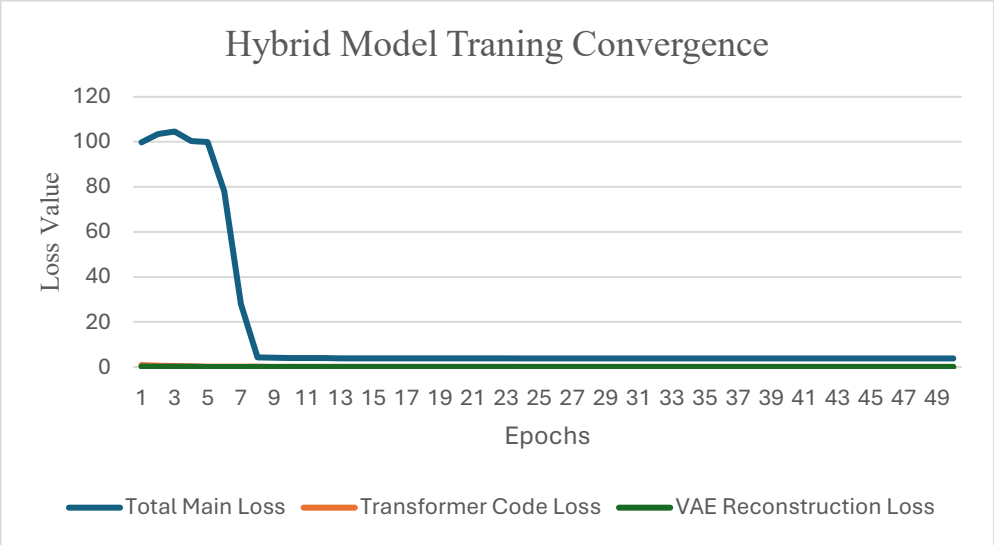


Figure 2: Training Losses

The experimental evaluation is conducting through a comprehensive dataset comprising real-world web requirements & user interactions [45]. The evaluation framework compared four distinct approaches such as traditional manual design based on human designer using conventional tools, template-based systems platforms like WordPress, AI-generated Approaches from existing web design tools and the proposed framework for the integrated hybrid approach. Each approach was evaluated on various websites with different complexity levels. The evaluation involved 200 participants from diverse backgrounds for testing.

5.2. Quantitative performance matrix

The evaluation of the model was done using the BLEU-4 score and the syntax validation percentage. The BLEU-4 score is 0.0351 at the 10th epoch and raise towards 0.0558 on 50th epoch as shown in figure 3. The syntax validation percentage is used to track the learning progress of the Transformer-based code generator at the 10th epoch interval as shown in figure 4. Here, the data reveals a critical breakthrough on Syntax Validity reached at 100% and remains stable throughout the training. This confirms that the Casual Masking and Beam Search Algorithm effectively eliminates the hallucinated and unclosed HTML tags.

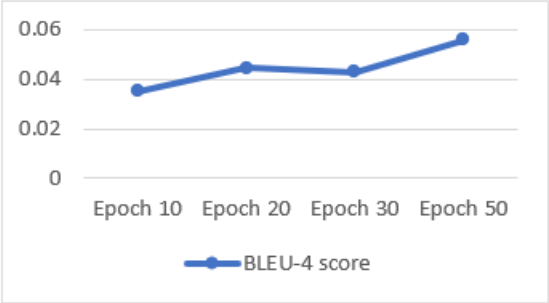


Figure 3: BLEU-4 score

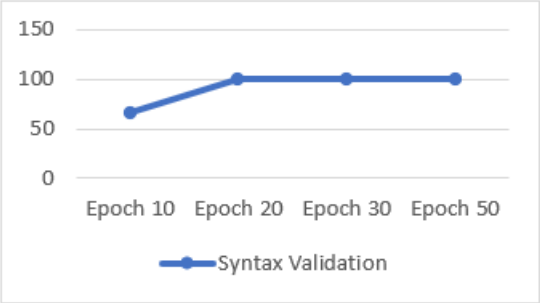


Figure 4: Syntax Validation Percentage

5.3 Structural Accuracy Analysis

The ability of the model is identified with a confusion matrix generated for the specific HTML structural tags. The is high precision element with up to 363 correct predictions and <button> tags prediction remains 118. The strong diagonal presents in the matrix validates the VAE-GAN design engine accurately translate visual UI components into corresponding call tokens. A minor confusion observed between and <p> tags. Figure 5 shows the confusion matrix of primary tags by comparing the actual tags in code with predicted tags.

Confusion Matrix: Core HTML Structural Tags

Actual (Ground Truth) Tag \ Predicted		<button>	<img>	<p>	<a>	<input>
	363	7	4	10	12	3
<button>	2	118	2	1	3	4
<img>	7	4	134	6	4	5
<p>	3	2	4	73	6	1
<a>	4	3	2	0	106	4
<input>	0	4	2	2	2	93

Predicted Tag

Figure 5: HTML Confusion Matrix

5.4. Performance Evaluation

The performance evaluation revealed the improvements across all measured metrics in Table 2 [46]:

Table 2: Performance Comparison of Web Design Approaches

Metric	Traditional	Template	AI-Generated	Proposed
Design Quality (/10)	7.2	6.8	8.1-8.5	9.2
User Satisfaction (/10)	7.5	6.9	8.2-8.6	9.1
Code Quality (/10)	8.0	6.5	7.8-8.3	9.4
Task Completion (%)	85	78	92	96
Development Time(hrs)	40	8	6-12	4

Compare the performance of various approaches with charts:

The approaches such as Design Quality, User Satisfaction, Code Quality is out of 10 in Figure 6. The proposed framework achieved a score on design quality 9.2, traditional manual design scored 7.2, template-based systems scored 6.8 and existing AI-generated approaches scored in the range from 8.1 to 8.5. Similarly, the proposed framework scored 9.1 & 9.4 for user satisfaction and code quality, human-written code scored 7.5 & 8.0 for user satisfaction & code quality, template-based system got a score of 6.9 & 6.5 for user satisfaction and code quality respectively [47].

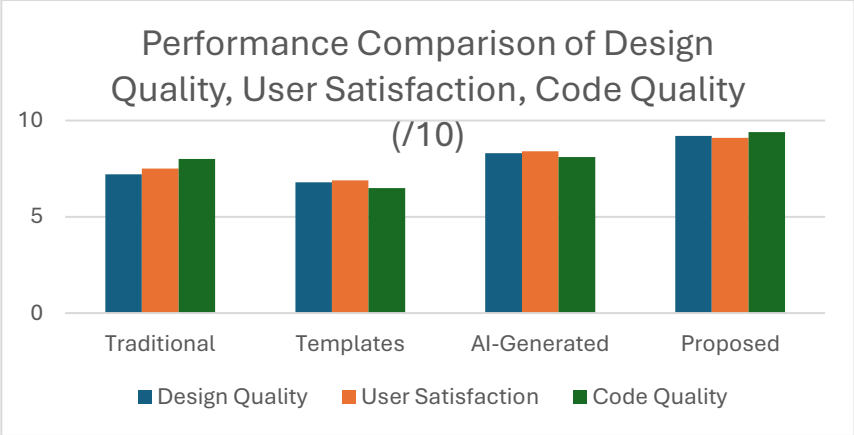


Figure 6: Comparison of Design Quality

Figure 7 shows the improvement in development time with the proposed framework needing 4 hours over the traditional which takes 40 hours in development time [48]. Figure 8 shows the comparison of the task completion rate of the proposed framework, traditional coding, AI-generated and template-based coding.

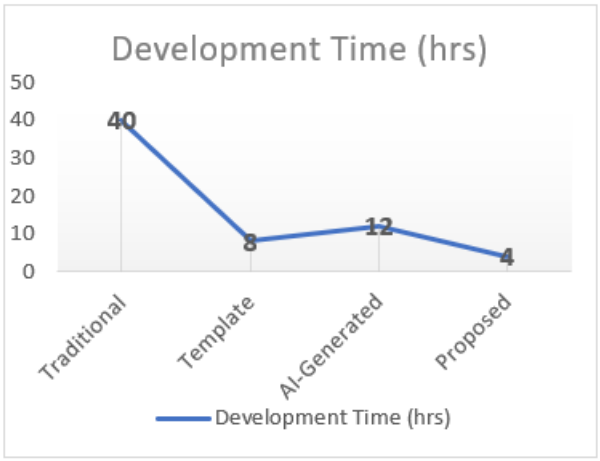


Figure 7: Development time of various approaches

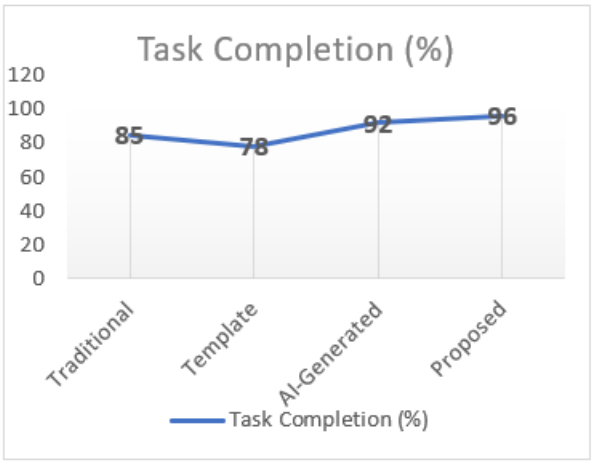


Figure 8: Comparison of Task Completion

Comparison with State-of-the-Art:

A comparison of existing state-of-the-art approaches demonstrates the framework's qualities:

Table 3: State-of-the-Art Performance Metrics

Metric	Traditional	AI-Generated	Proposed
Load Time (s)	3.2	2.4	1.4
Time to First Byte (ms)	450	320	220
Accessibility Score (/10)	7.2	8.1	9.2
SEO Score (/10)	6.8	7.9	9.1
User Engagement (%)	62	78	85

Performance Metrics: Load times improved over 3.2 seconds to 1.4 in the traditional approach to the proposed framework and the time to first byte reduced from 450ms to 220ms for the proposed framework as compared to the traditional approach shown in Figure 9 and Figure 10 [49].

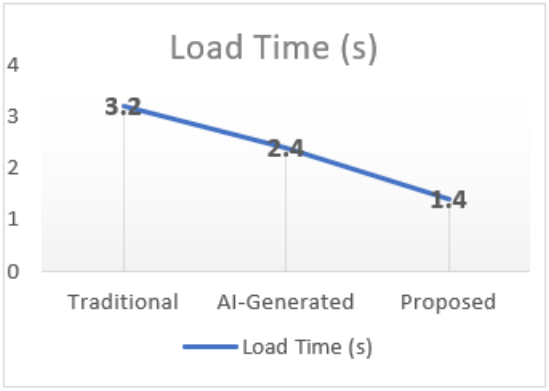


Figure 9: Load Time in seconds

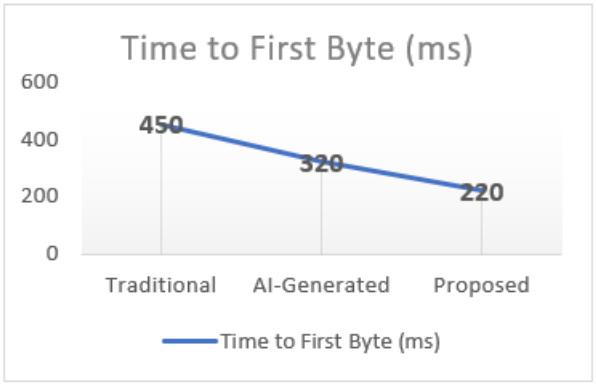


Figure 10: Comparison of Time to First Byte in milliseconds

Accessibility & SEO: The score of accessibility & SEO is out of 10. The framework scores for accessibility are 9.2 & SEO scores of 9.1 for proposed models and traditional method scores 7.2 and 6.8, respectively. Comparison shown by Figure 11 and Figure 12 [30].

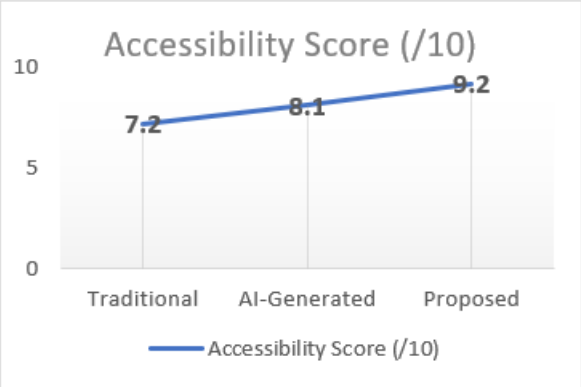


Figure 11: Comparison of Accessibility & SEO

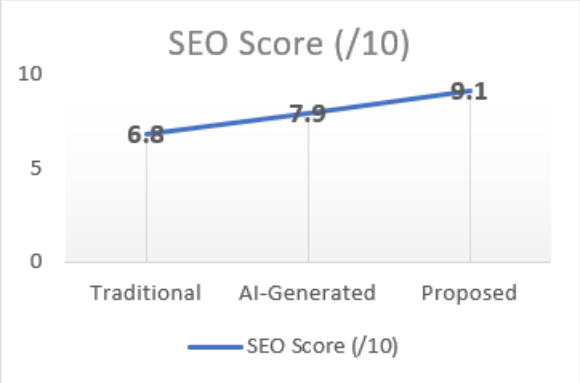


Figure 12: Comparison of Accessibility & SEO

User Engagement: User engagement rates increased from 62% to 85% in the proposed model over the traditional approaches and 78% for existing AI-generated solutions. Comparison shown in Figure 13 [50].

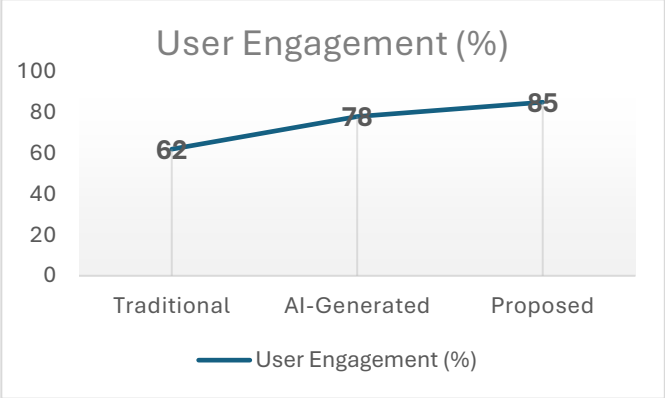


Figure 13: Comparison of User Engagement

5.5. Discussion

The experimental results of research validate the efficiency of the hybrid architecture to bridge the gap between visual intent & functional code. The key points of discussion are the VAE-GAN engine, Transformer-based Code Generation and the Optimiser Layer. The visual perception has a dual-model design in the framework. A ResNet VAE model allows to compare the UI features into a 512-dimensional latent vector is shown in the system architecture. The reconstruction loss (0.2839) confirms the success of the latent space in capturing the structure of the webpage. Furthermore, the GAN Discriminator force the model to achieve professional-grade aesthetic fidelity and reflects the high design quality in human evaluation. The causal masking and repetition penalty used in Algorithm 2 help to achieve 100% syntax validity by epoch 20 and remain consistent later in the experiment instead of getting stuck in code hallucination or infinite loop. On the other hand, the transformer code loss converges at 0.2412 to demonstrate the rigorous top-down hierarchical learning model. Furthermore, the confusion matrix shows high precision for structural tags. These results indicate that the framework has moved beyond the template filling to semantic understanding of UI components. Finally, the optimisation layer is used to apply a post-generation syntax checking and reinforcement learning based on the penalty engine to eliminate the looping bugs generated by standard transformers.

6. Conclusion & Future Work

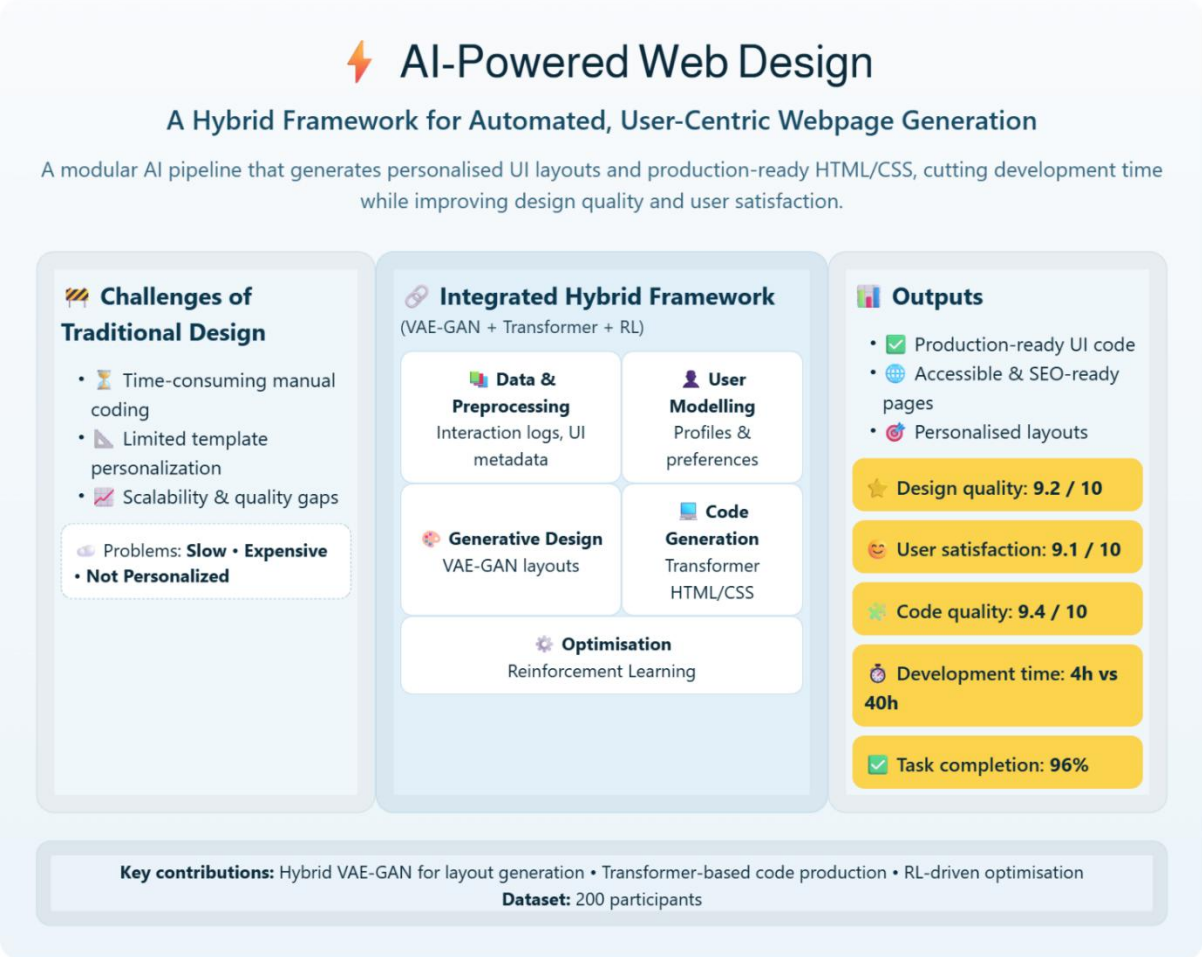
This research successfully addresses the limitations of traditional web design by integrating Artificial Intelligence & Machine Learning into a hybrid framework. The identified gap in the literature review can be filled with high precision by demonstrating a hybrid VAE-GAN-Transformer model. The key contribution of the research includes an architectural upgradation by introducing a hybrid pipeline where the Variational Autoencoder establishes a 512-dimensional latent design space which is refined by the GAN discriminator for visual realism. The

transformer-based decoder helped to achieve the 100% syntax validation by epoch 20 eliminates the structural inconsistencies. The framework reached at 0.2412 for code generation loss and 0.2839 for design reconstruction loss in the 50-epoch training. This proves the ability to learn complex UI hierarchies from the Web2Code dataset.

The current framework provides a robust foundation for automated webpage design. It also shows a path for future research areas such as Human-AI collaboration with active-learning models where human designers provide real-time feedback to optimisation layer to enhance the creative variation for output. Integration of multi-model inputs to expand the system to recognise voice commands and gestures as inputs. The model enhancement to increase complexity & scalability for designing large-scale and multi-page web applications. The automated models always have a chance to give biased results also manageable in future work. Ultimately, the future of web design is moving towards an intelligent ecosystem where human creativity can be boosted by computational precision using hybrid AI architectures.

7. Acknowledgement: The authors declare that they have no conflict of interest regarding the publication of this manuscript. No funding was received from any agency, institution or organisation for the preparation of this work.

Graphical Abstract



References:

1. Å. Stige, P. Mikalef, E.D. Zamani, Y. Zhu, Artificial Intelligence (AI) and User Experience (UX) Design: A Systematic Literature Review and Future Research Agenda, Information Technology & People 37 (2023) 2324–2352. https://doi.org/10.1108/ITP-07-2022-0519.

2. 10 AI Tools Transforming Web Development in 2025 | DigitalOcean, (n.d.). https://www.digitalocean.com/resources/articles/ai-tools-web-development (accessed July 14, 2025).
3. N. Guler, S. Kirshner, R. Vidgen, A Literature Review of Artificial Intelligence Research in Business and Management Using Machine Learning and ChatGPT, Data & Information Management 8 (2024) 100076. https://doi.org/10.1016/j.dim.2024.100076.
4. E. Mohammadi, M. Thelwall, Y. Cai, T. Collier, I. Tahamtan, A. Eftekhar, Is generative AI reshaping academic practices worldwide? A survey of adoption, benefits, and concerns, Information Processing & Management 63 (2025) 104350. https://doi.org/10.1016/j.ipm.2025.104350.
5. L. Some, W. Yang, M. Bain, B.H. Kang, A Comprehensive Survey on Integrating Large Language Models with Knowledge-Based Methods, (2025). https://doi.org/10.2139/ssrn.5111497.
6. AI Website Builder - Design & Create A Website In Minutes, Elementor (n.d.). https://elementor.com/products/ai/ (accessed July 14, 2025).
7. Y. Lu, Y. Yang, Q. Zhao, C. Zhang, T.J.-J. Li, AI Assistance for UX: A Literature Review Through Human-Centered AI, (2024). https://doi.org/10.48550/arXiv.2402.06089.
8. How to Learn Machine Learning in 2024 - GeeksforGeeks, (n.d.). https://www.geeksforgeeks.org/machine-learning/how-to-start-learning-machine-learning/ (accessed July 14, 2025).
9. Relume — Websites designed & built faster with AI | AI website builder, (n.d.). https://www.relume.io/ (accessed July 14, 2025).
10. G. Wagner, R. Lukyanenko, G. Paré, Artificial Intelligence and the Conduct of Literature Reviews, Journal of Information Technology 37 (2022) 209–226. https://doi.org/10.1177/02683962211048201.
11. AI Website Builder - Create a website in 3 minutes, Webwave.Me (n.d.). https://webwave.me/ai (accessed July 14, 2025).
12. Y. Shi, T. Gao, X. Jiao, N. Cao, Understanding Design Collaboration Between Designers and Artificial Intelligence: A Systematic Literature Review, Proc. ACM Hum.-Comput. Interact. 7 (2023) 368:1-368:35. https://doi.org/10.1145/3610217.
13. R.A. Mancisidor, M. Kampffmeyer, K. Aas, R. Jenssen, Generating customer's credit behavior with deep generative models, Knowledge-Based Systems 245 (2022) 108568. https://doi.org/10.1016/j.knosys.2022.108568.
14. AI Website Builder: Create Websites From Text | Replit, (n.d.). https://replit.com/usecases/ai-website-builder (accessed July 14, 2025).
15. Y. Wan, C. Wang, Y. Dong, W. Wang, S. Li, Y. Huo, M.R. Lyu, Automatically Generating UI Code from Screenshot: A Divide-and-Conquer-Based Approach, (2025). https://doi.org/10.1145/3729364.
16. Automate HTML Development Using AI Tools, (n.d.). https://www.dhiwise.com/post/automate-html-development-using-ai-driven-tools (accessed July 14, 2025).
17. SEO Automated Content Generation:AI Tools to Rank Higher, (n.d.). https://www.poweredbysearch.com/learn/seo-automated-content-generation/ (accessed July 14, 2025).
18. AI Image Generator - Create Art, Images & Video | Leonardo AI, Https://Leonardo.Ai/ (n.d.). https://leonardo.ai/ (accessed July 14, 2025).
19. AI Website Builder: Create a Custom Website in Minutes with AI, Squarespace (n.d.). https://www.squarespace.com/websites/ai-website-builder (accessed July 14, 2025).
20. V. Parada, Automatic Generation Of Algorithms, CRC Press, 2025.
21. Improve performance and UX for client-side AI | web.dev, (n.d.). https://web.dev/articles/client-side-ai-performance (accessed July 15, 2025).
22. AI Code Generation Explained: A Developer's Guide, About.Gitlab.Com (n.d.). https://about.gitlab.com/topics/devops/ai-code-generation-guide/ (accessed July 15, 2025).
23. S. Ojuri, T.A. Han, R. Chiong, A. Di Stefano, Optimizing text-to-SQL conversion techniques through the integration of intelligent agents and large language models, Information Processing & Management 62 (2025) 104136. https://doi.org/10.1016/j.ipm.2025.104136.
24. A. Wibisono, Denny, P. Mursanto, S. See, Natural generative noise diffusion model imputation, Knowledge-Based Systems 301 (2024) 112310. https://doi.org/10.1016/j.knosys.2024.112310.
25. R.T.A. Guna, O.K. Sikha, R. Benitez, Interpreting CNN predictions using conditional Generative Adversarial Networks, Knowledge-Based Systems 302 (2024) 112340. https://doi.org/10.1016/j.knosys.2024.112340.
26. X. Zhan, Y. Xu, Y. Liu, Personalized UI Layout Generation Using Deep Learning: An Adaptive Interface Design Approach for Enhanced User Experience, International Journal of Innovative Research in Engineering and Management 11 (2024) 68–79. https://doi.org/10.55524/ijirem.2024.11.6.7.
27. S. Duan, R. Zhang, M. Chen, Z. Wang, S. Wang, Efficient and Aesthetic UI Design with a Deep Learning-Based Interface Generation Tree Algorithm, (2024). https://doi.org/10.48550/arXiv.2410.17586.
28. A. Svyatkovskiy, S.K. Deng, S. Fu, N. Sundaresan, IntelliCode Compose: Code Generation Using Transformer, (2020). https://doi.org/10.48550/arXiv.2005.08025.
29. A. Delitzas, K.C. Clis, M.S. Kosslyn, Calista: A Deep Learning-Based System for Understanding and Evaluating Website Aesthetics, Int. J. Hum.-Comput. Stud. 180 (2024) 103019. https://doi.org/10.1016/j.ijhcs.2023.103019.
30. A. Costa, F. Silva, J.J. Moreira, Towards an AI-Driven User Interface Design for Web Applications, Procedia Computer Science 237 (2024) 179–186. https://doi.org/10.1016/j.procs.2024.05.094.

31. AI Website Builder - Create A Website In Minutes | Wix, (n.d.). https://www.wix.com/ai-website-builder (accessed July 14, 2025).
32. Y. Luo, Designing With AI: A Systematic Literature Review on the Use, Development, and Perception of AI-Enabled UX Design Tools, Advances in Human-Computer Interaction 2025 (2025) Article ID 3869207. https://doi.org/10.1155/ahci/3869207.
33. I.H. Sarker, Machine Learning: Algorithms, Real-World Applications and Research Directions, Journal of Big Data 8 (2021) 27. https://doi.org/10.1186/s40537-021-00426-2.
34. Machine Learning Crash Course, Google for Developers (n.d.). https://developers.google.com/machine-learning/crash-course (accessed July 14, 2025).
35. AI UI Design | AI-Powered UI Design Is Here! | Uizard, (n.d.). https://uizard.io/ai-design/ (accessed July 14, 2025).
36. Machine Learning Algorithms - GeeksforGeeks, (n.d.). https://www.geeksforgeeks.org/machine-learning/machine-learning-algorithms/ (accessed July 15, 2025).
37. UI Testing - The Definitive Guide for 2025, (n.d.). https://www.globalapptesting.com/blog/ui-testing/ (accessed July 15, 2025).
38. 10 Machine Learning Algorithms to Know in 2025 | Coursera, (n.d.). https://www.coursera.org/in/articles/machine-learning-algorithms (accessed July 15, 2025).
39. B. Martinović, R. Rozić, Perceived Impact of AI-Based Tooling on Software Development Code Quality, SN COMPUT. SCI. 6 (2025) 63. https://doi.org/10.1007/s42979-024-03608-4.
40. UI Automation Testing: What it is, Tools, Steps & Best Practices | BrowserStack, (n.d.). https://www.browserstack.com/guide/ui-automation-guide (accessed July 15, 2025).
41. A. Deolekar, K. Dhanawade, R. Chavan, S. Bhambure, UI Code Generation using Deep learning, 08 (2021).
42. What is Attention and Why Do LLMs and Transformers Need It? | DataCamp, (n.d.). https://www.datacamp.com/blog/attention-mechanism-in-llms-intuition (accessed July 15, 2025).
43. Deep Learning Tutorial - GeeksforGeeks, (n.d.). https://www.geeksforgeeks.org/deep-learning/deep-learning-tutorial/ (accessed July 15, 2025).
44. N. Das, R. Panta, N. Karki, R. Manandhar, D.B. Kshatri, A Comparative Study on Code Generation with Transformers, (2024). https://doi.org/10.48550/arXiv.2412.05749.
45. 7 Steps to Benchmark Your Product's UX, Nielsen Norman Group (n.d.). https://www.nngroup.com/articles/product-ux-benchmarks/ (accessed July 15, 2025).
46. AI Web Design Tools for Teams | Ramotion Agency, (n.d.). https://www.ramotion.com/blog/ai-web-design-tools/ (accessed July 15, 2025).
47. AI in Web Development - Transforming Trends, Tools, and Future Insights, (2025). https://tutedude.com/blogs/ai-in-web-development-trends-tools-insights/ (accessed July 15, 2025).
48. M. Veggi, State of the Art on Artificial Intelligence Resources for Interaction Media Design in Digital Cultural Heritage, (2025). https://doi.org/10.48550/arXiv.2504.13894.
49. K. Ofosu-Ampong, Artificial intelligence research: A review on dominant themes, methods, frameworks and future research directions, (2024).
50. N.S. Hidayat, M.D.A. Kautsar, A.F. Wicaksono, F. Koto, Simulating Training Data Leakage in Multiple-Choice Benchmarks for LLM Evaluation, (2025). https://doi.org/10.48550/arXiv.2505.24263.