International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN (Online): 2347-5552, Volume-13, Issue-3, May 2025
https:/doi.org/10.55524/ijircst.2025.13.3.12
Article ID IRP-1636, Pages 73-75
www.ijircst.org

Text-to-Image Generator Web Application Using React.js and Node.js for Dynamic Image Creation and Storage

Pervez Rauf[1], Md Wasim Khan[2], Md Ambar Shafi[3], Md Sahil[4], and Md Shahbaz Shamim[5]

[1]Assistant Professor, Department of Computer Science & Engineering, Integral University, Lucknow, India
[2, 3, 4, 5]B.Tech Scholar, Department of Computer Science & Engineering, Integral University, Lucknow, India

Correspondence should be addressed to Md Shahbaz Shamim mdshahbazshamim9@gmail.com

Received 12 April 2025; Revised 26 April 2025; Accepted 11 May 2025

ABSTRACT- The field of generative artificial intelligence has experienced tremendous growth with the advent of deep learning-based text-to-image models, revolutionizing how machines interpret and visualize human language. This paper presents a comprehensive overview of a full-stack web-based application designed to harness this technology for practical and creative use. The application tries facilitating the generation of images from text descriptions by integrating frontend and backend technologies offering a smooth user experience and efficient performance. The system features a robust frontend built with React.js, enabling a dynamic and responsive user interface that supports real-time interactions. Tailwind CSS is well-used to ensure a consistent, mobile-first design framework that adapts onto various screen sizes and devices. On the backend, the application utilizes Node.js with the Express.js framework to look on server-side logic, route handling, and communication with external services. RESTful APIs bridge the frontend and backend, allowing clean and scalable request handling between the client and the server. For media management, the application incorporates Multer, a middleware for handling multipart/form-data, which is primarily used for uploading files. This enables users not only to generate new images from text prompts but also to upload existing images for display or further analysis. A gallery interface is provided, allowing users to browse previously generated content, encouraging exploration, creativity, and reuse of past results. Central to the system is the integration of a pre-trained deep learning-based image generation model, capable of translating natural language prompts into high-quality, photorealistic, or stylized images. This model leverages state-of-the-art transformer architectures and diffusion techniques, ensuring accuracy and fidelity in the visual output. The system supports a variety of prompt types, including descriptive, abstract, and conceptual input, expanding its applicability across domains such as art, education, entertainment, and marketing. Extensive testing and evaluation of the platform confirm its effectiveness in delivering real-time image generation with low latency and minimal resource overhead. The application also demonstrates strong user interactivity features, such as prompt history, loading indicators, and error handling for invalid input. Backend optimizations and asynchronous data handling ensure that large image files are processed efficiently without degrading the user experience.

KEYWORDS- Text-to-Image Generation, Web Application, React.JS, Node.Js, AI Image Synthesis, Rest API, Image Gallery.

I. INTRODUCTION

Text-to-image generation is a good area of artificial intelligence that's all about turning written words into pictures [1] [2]. It lets you type in a description—and an image based on that description gets created. This kind of technology has opened new doors in areas like graphic design, education, entertainment, and digital media. With the latest advances in deep learning—especially with models like diffusion and transformers [1][2]AI has become really better at creating realistic or artistic image from just a few lines of text. But even though the tech is impressive, there's still a problem: most platforms that use these models are either hard to use or require powerful hardware and tech know-how that regular users don't have. That's where this project comes in. It's a web-based project that makes text-to-image generation easy and accessible to everyone. You don't need any special equipment but just a browser. The front-end is done with React.js [5], so it's smooth, modern, and works great on phones and computers alike. Behind the scenes, Node.js and Express take care of the server tasks like handling user requests and running the app's logic [6]. Using the tool is simple. You type in what you want to see, and the app shows you an image created by AI. You can also have a look through a gallery of past images and even upload your own. The image generation itself is powered by advanced APIs like OpenAI's DALL·E [3] or Stability AI's Stable Diffusion [4], depending on what the user needs. All in all, this project offers a creative and practical way for people to explore what AI can do. It breaks down barriers to entry, inspires imagination, and brings powerful tools to everyone—whether you're a student, a designer, or just curious.

II. RELATED WORKS

A growing number of platforms now offer text-to-image generation capabilities by leveraging powerful APIs such as OpenAI's DALL·E [3], MidJourney, and Stability AI's

Stable Diffusion [4]. These services have made it possible to integrate advanced image synthesis into a variety of applications, leading to increased experimentation and adoption in both commercial and research settings [1], [2]. Previous studies and implementations have primarily focused on backend development, utilizing frameworks like Flask and Django, or deploying directly through cloud-based services to process user inputs and return generated images. Although these backend-focused solutions have proven the feasibility and performance of AI-based image generation, relatively few academic or open-source projects have extended their focus to include fully-featured, user-centric interfaces or robust media management systems [2]. In many cases, the front-end functionality is minimal, and file handling, including uploads and persistent image storage, is either absent or inadequately addressed. This project advances beyond existing efforts by implementing a complete full-stack architecture that emphasizes not only backend integration and API connectivity but also an engaging and scalable user interface [2]. The frontend is carefully structured to support a responsive user experience, offering smooth navigation and real-time feedback [5], [7]. One of the core additions is a custom-built image gallery system that allows users to view, manage, and revisit their generated content efficiently. Additionally, the application supports streamlined image uploads, enabling users to contribute or reuse visual content beyond what is generated by the model [8]. What sets this system apart is its holistic approach. Rather than functioning solely as a demonstration of model capability or backend performance, it delivers an end-to-end solution that combines technical depth with an accessible and polished user experience. This comprehensive design makes the platform not only practical for developers and researchers but also inviting for general users, educators, and content creators [2].

III. METHODOLOGY

The development process involved full-stack architecture planning and integration of various technologies. The system consists of multiple interconnected modules as outlined below:

A. Frontend Development

React.js is utilized to develop a responsive and interactive user interface, enabling seamless user interactions and dynamic content updates. Tailwind CSS was integrated to apply utility-first styling, ensuring that the application maintains a consistent and adaptive layout across both mobile and desktop devices. The frontend includes several key features that enhance usability and user experience. These use a text input form where users can enter prompts to generate images, real-time loading indicators that provide feedback during the image generation process, a visual gallery that displays previously generated images with modal previews for detailed viewing, and an upload interface that allows users to submit and manage their own image content.

B. Backend and API Integration

Node.js, in combination with the Express framework, was employed to manage HTTP requests and handle server-side operations efficiently. RESTful APIs were developed to facilitate key backend functionalities, including accepting text prompts from the frontend and forwarding them to the AI image generation model, returning the URLs of generated images back to the client, and managing file uploads while storing relevant metadata in either a local or cloud-based database. To support file handling, Multer middleware was integrated, enabling secure and efficient processing and storage of uploaded image files.

C. Database Integration

The system uses MongoDB as the primary database for storing metadata related to user prompts, generated images, and upload history. MongoDB's flexible schema design is well-suited for JSON-like data structures and enables efficient querying and indexing. Mongoose, an ODM (Object Data Modeling) library for MongoDB, is utilized to simplify schema definition and database interactions. The database stores entries that include timestamps, prompt text, image URLs, user session information, and tags for organization. For scalability, the architecture allows optional integration with cloud databases such as MongoDB Atlas. This enables real-time synchronization, remote access, and enhanced data security features, making it ideal for production deployment.

D. Image Generation

The core image generation functionality of the system is powered by third-party APIs, including OpenAI's DALL·E and locally hosted instances of Stable Diffusion. These advanced AI models are capable of transforming natural language descriptions into detailed visual outputs. When a user submits a text prompt through the frontend interface, the input is transmitted to the backend, which then forwards the request to the selected image generation API. The API processes the prompt using deep learning techniques—such as diffusion processes or transformer-based architectures—and generates a corresponding image. Once the image is created, the API responds using the help of a URL or binary data representing the image, which is then returned to the frontend for display in the user interface. This architecture enables the application to offer high-quality, AI-generated visuals in real time while maintaining flexibility in choosing between cloud-based or locally hosted generation services.

E. Security and Optimization

To maintain system integrity and provide a secure environment for all users, basic input validation mechanisms were implemented to sanitize and verify user-submitted data before it is processed. This helps prevent common issues such as malformed inputs, injection attacks, and unintended API behaviour. Additionally, rate limiting was introduced to restrict the number of requests a user can make within a specific time frame, thereby mitigating the risk of abuse, such as spamming the API or overloading the server with huge traffic. To further enhance performance and responsiveness, the system has strategies to temporarily store frequently accessed data which reduces the need to continuously fetch identical content from the API or database. This tends to significantly faster load times and a smoother user

experience. Also, asynchronous operations were employed to ensure that time-consuming tasks, such as image generation or file uploads, do not block the main application thread. By processing these operations in the background, the application remains responsive and capable of handling multiple user interactions concurrently. Together, these measures contribute to a more secure, efficient, and user-friendly platform.

IV. EVALUATION AND RESULTS

The system has been thoroughly tested across a variety of scenarios, including both mobile and desktop web browsers, to ensure broad compatibility and consistent performance. Several key evaluation metrics were used to assess the system's effectiveness. The "generation latency", which measures the time from prompt submission to the display of the generated image, averaged under five seconds, indicating efficient backend processing and API response time. The "gallery loading time" was also optimized, with a good view successfully loading several images in less than few seconds. Additionally, the system demonstrated robust performance in handling uploads, with upload throughput tests confirming that multiple 5 MB files could be uploaded simultaneously without triggering timeouts or errors. A usability study conducted with a group of 20 participants revealed high levels of user satisfaction. Participants praised the platform for its ease of use, visually appealing design, and quality of the images produced. These findings support the effectiveness of the application in delivering a responsive and engaging user experience. In conclusion, this paper introduces a full-stack web application for text-to-image generation, built using React.js on the frontend and Node.js on the backend.

V. CONCLUSION

The integration of better AI-based image generation models with a responsive and user-friendly interface allows users to generate, upload, and manage visual content efficiently. The system's design balances technical performance with intuitive interaction, making it accessible for a wide range of users. Looking ahead, future enhancements could include implementing user authentication, maintaining prompt history, supporting multiple generation models, and offering built-in image editing tools. This project highlights the potential of combining modern web development frameworks with artificial intelligence to create powerful and interactive digital applications.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

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," arXiv preprint arXiv:2102.12092, Feb. 2021. Available from: https://arxiv.org/abs/2102.12092

[2] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. Ghasemipour, B. Ayan, S. Mahdavi, R. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," Advances in Neural Information Processing Systems (NeurIPS), 2022. Available from: https://arxiv.org/abs/2205.11487

[3] OpenAI, "DALL·E API documentation.". Available from: https://platform.openai.com/docs/guides/images

[4] Stability AI, "Stable Diffusion Documentation.". Available from: https://stability.ai

[5] React, "React Documentation.". Available from: https://reactjs.org

[6] Express.js, "Express.js Documentation.". Available from: https://expressjs.com

[7] Tailwind CSS, "Tailwind CSS Documentation.". Available from: https://tailwindcss.com

[8] Express.js, "Multer Middleware.". Available from: https://github.com/expressjs/multer