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

Textual Prompt Interpretation for Image Synthesis Using Generative AI Techniques

Dr. Anita Harsoor[1], Rahul Magaji[2], Rakesh Rathod[3], Prajwal[4]

[1]Professor, Computer Science and Engineering Dept, Poojya Doddappa Appa College of Engineering, Kalaburagi, Karnataka, India
[2,3,4]Student, Computer Science and Engineering Dept, Poojya Doddappa Appa College of Engineering, Kalaburagi, Karnataka, India

ABSTRACT:
This paper describes how the design and development of a text-to-image creation application for mobile phones with advanced ways of generating artificial intelligence. Users have access to the application to insert text prompts and get relevant responses and AI-generated images. It relies on a form of diffusion model with Replicate as its host platform. visual results that are accurate and well-constructed. ExpressJS is used on the backend, and React Native (Expo) is used on the frontend. Strapi CMS is used together with PostgreSQL for backend development. The media storage and retrieval are done through Neon.tech delivered through Cloudinary to make the user experience smooth and efficient and have the ability to deal with media smoothly. Emphasis is given to the architecture, modular design, the ability to grow and cloud-powered features to keep users performing in real time. The paper shows the way the system was set up and how it was built. This permits more people to access user-friendly creative AI tools to boost the ways of content generation.

KEYWORDS: Generative AI, Text-to-Image Generation, Diffusion Models, React Native, Strapi CMS, Cloudinary, Replicate AI.

INTRODUCTION

Because of generative AI, machines now generate content that looks and sounds like something written or spoken by people. Text-to-image generation has become very popular, letting users turn their ideas into pictures by describing them. People use this in art, design, education and marketing settings. A cross-platform mobile app was created that uses an AI model based on diffusion to turn language descriptions into pictures. Using a mobile-first development approach allows customers to easily manage their books from anywhere. People are becoming more interested in AI-based platforms for creativity, which is clear from the popularity of DALL·E, Midjourney and Stable Diffusion. Even so, using these tools may need piracy tools or web access. To cover this gap, we made our app supportive of both hands-on mobile use and access to advanced tools managed by the backend server. The main hurdles resolved in this study are bringing in third-party AI models, handling multimedia files and efficient communication with mobile app users. We use Replicate.com for inference, Strapi CMS for content behind the scenes, PostgreSQL for storing data and Cloudinary for managing and scaling media files.

PROPOSED SYSTEM

The system is created to be scalable, use cloud architecture and include a modular approach that allows users to create images from text prompts on their phones. It relies on the React Native frontend, a backend built with Strapi CMS and PostgreSQL, the AI inference layer on Replicate.com and Cloudinary for providing storage and delivery for the visuals. The frontend uses React Native with Expo, meaning the app can be made and run on Android and iOS quickly and easily. Users write the business prompts using the interface, which is then safely transmitted to the backend through RESTful API calls.

Strapi CMS, PostgreSQL and Neon manage the backend, which responds to requests, logs the original prompt and shows the metadata for each image that is created. As soon as the prompt arrives, the backend talks with Replicate.com's diffusion model using special keys. The model goes through the input text and gives back an image URL. Cloudinary, a cloud platform for managing media, saves, processes and serves the image as quickly and safely as possible. The image with the associated data, such as prompt and time of generation, is kept in the database for easy retrieval and display later.

As a result, users quickly interact with the system since image-related tasks and media storage are moved to cloud platforms. Breaking apart the responsibilities of the frontend, backend, AI model and media storage makes it simple to maintain and expand the app. It also helps create images on the spot and forms a base for future changes such as user account systems, galleries and advanced ways to personalise the platform.

SYSTEM DESIGN

The application uses system architecture that keeps it easy for the management and efficient by dividing the frontend, backend and media services. With the React Native and Expo framework, the frontend is built and acts as the main place for the users to enter text and get back the image results. It is connected with the backend using safe API methods.

Managed by Strapi CMS on the backend, it serves API endpoints for handling the data, prompts and user sessions. It mediates data transactions, keeps the flow secure and handles calls to external services such as the Replicate API for AI image generation. Neon.tech PostgreSQL is a reliable, scalable and cloud-native relational database that is where all Neon Structured Data is stored. Cloudinary is also included to handle how images are both stored and sent to the users. After the Replicate API generates an image from the prompt given to the AI, the system puts these images on Cloudinary. As a result, the correct image version is fetched by users, making downloads efficient and boosting their experience.

Clerk enables the application to authenticate users so that people can register and handle their accounts with security. Thanks to this process, just authorised people can access and make use of the application. Because of all these components, the AI behind image processing can effectively manage user requests, process information and deliver top-quality AI pictures using many devices.

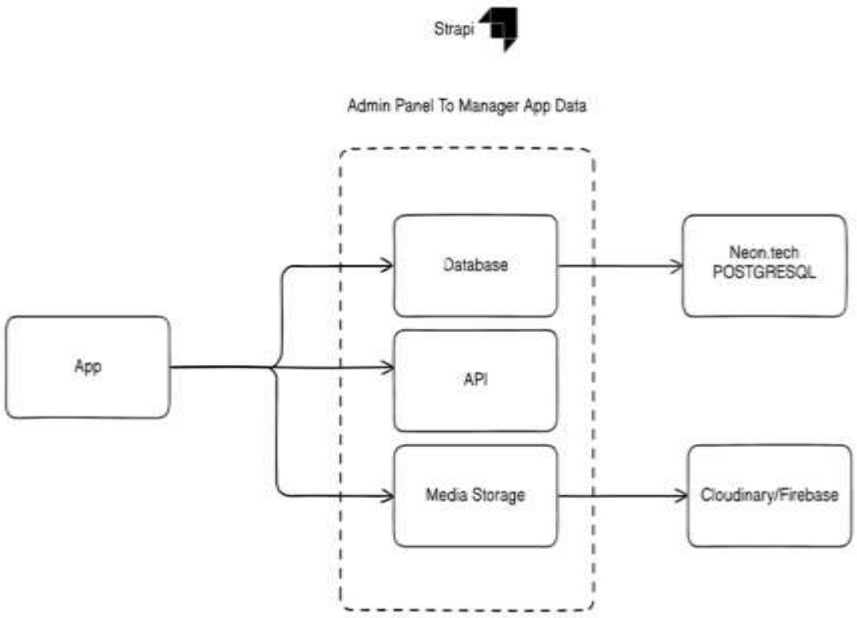

Fig. 1 System Architecture

Description of used technologies:

React Native: Meta created React Native, an open-source framework, to help developers make applications for both Android and iOS. It allows programmers to code with JavaScript and see their applications work smoothly on all devices, whether Android or iOS. Because of its rich ecosystem and reusable components, changes and additions can be made rapidly, so APIs and services from other services integrate without trouble.

Strapi CMS: Strapi CMS is a open-source option that developers use to generate custom APIs quickly. The admin area is easy to operate, and it has broad plugin capabilities. Strapi takes care of all the API requests, looks after the data, and provides secure endpoints for the frontend.

Expo: Expo relies on React Native and helps developers by providing helpful tools and services. It makes it simple for developers to develop, test, and release software fast since they do not need to deal with the details of native code, and it offers features such as over-the-air updates and easy builds with EAS.

Cloudinary: Cloudinary is a cloud service that handles storing, optimising and sending images and videos. Cloudinary makes the uploading of media, its transformation and its transportation through a safe CDN fast and efficient.

Neon.tech PostgreSQL: Using Neon.tech PostgreSQL, modern applications can rely on serverless PostgreSQL platforms. The system can grow, is always accessible and manages relational data efficiently, so it can be used in the project for handling structured data.

Android Studio SDK: Android Studio SDK is used as the official integrated development environment for making Android apps. You can use emulators, debuggers and build tools from this package to guarantee Android apps are properly compatible and stable.

Replicate API: Developers use Replicate API to add AI models to the apps they build. In this case, it allows users to interact with Stable Diffusion to produce images following the user's prompts.

Clerk: Clerk helps the different applications to securely handle user registration, login, and session controls using email, acting as a user management and authentication system.

Visual Studio Code: With Visual Studio Code, you can work on multiple programming languages and

frameworks in a powerful, yet light editor. Using it, you can develop, handle, and test the project code.

IMPLEMENTATION

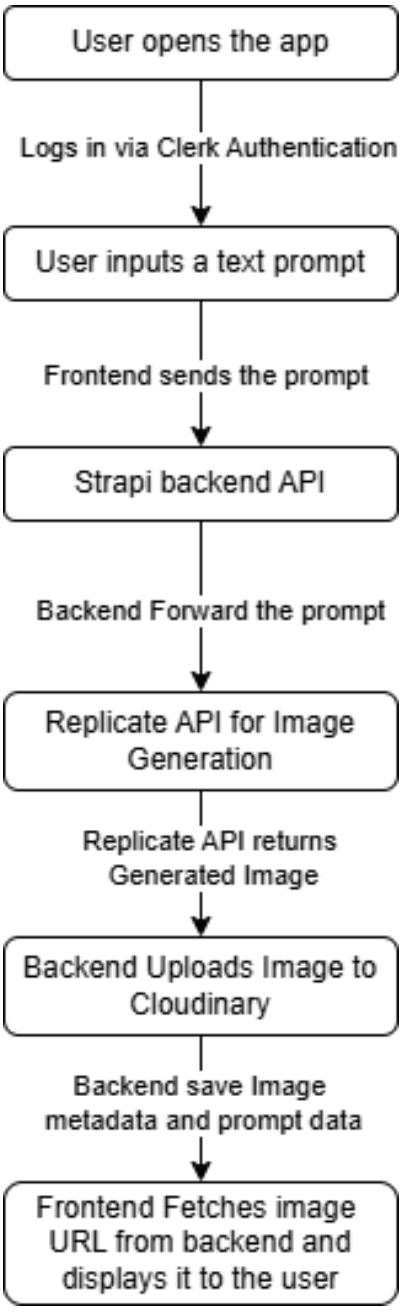

Fig. 2 Flow Diagram

1. **User opens the app:**

The user opens the mobile application on their Android or iOS device, built using React Native with Expo.

2. **Logs in via Clerk authentication:**

The user is authenticated through Clerk, a user authentication service, for making secure access and enabling features like user-specific image galleries in the future.

3. **The user inputs a text prompt:**

The user enters a descriptive text prompt (e.g., “a futuristic city at sunset”) using a simple and interactive text input field in the app.

4. The frontend sends the prompt:
Once the prompt is submitted, the frontend app sends a secure API request having the prompt text to the backend system for processing.

5. Strapi backend API:
The backend, built using Strapi CMS, receives the request. Strapi manages the backend logic, including validation, user association, and routing the prompt to the image generation service.

6. The backend forwards the prompt:
Strapi relays the prompt text to the Replicate API, which hosts diffusion-based image generation models such as Stable Diffusion or SDXL.

7. Replicate API for image generation:
The Replicate AI model receives the prompt from the backend, processes it using advanced algorithms and generates a realistic image. This step can take some 10–15 seconds.

8. Replicate API sends the generated image as a URL:
The generated image from the AI model is sent to the Strapi backend as a URL.

9. The backend uploads the image to Cloudinary:
The backend uploads the image to Cloudinary, which ensures fast content delivery and proper media handling. These types of services provide CDN support, caching, and transformation features.

10. The backend saves image metadata and prompt data:
Along with storing the image, the backend logs the prompt and metadata (e.g., image URL, user ID, timestamp) are stored in the PostgreSQL database, enabling future retrieval and analytics.

11. Frontend fetches image URL from the backend and displays it on the screen:
The final step is to fetch the image URL and metadata from the backend and display the generated image in the mobile app's UI.

RESULTS AND DISCUSSION

It proves that the AI system can produce recommended images within a mobile application used by many types of devices. Adding Replicate API, Cloudinary, Firebase and Clerk to the app was easy and made the system ready for use without placing much burden on the end device.

1. Login screen UI:

Fig. 3 login screen UI

The screen introduces users and invites them to use a reliable method to log into their account. Because the interface is minimalistic, anyone who is new to the app can use it easily.

2. User Authentication using Clerk:

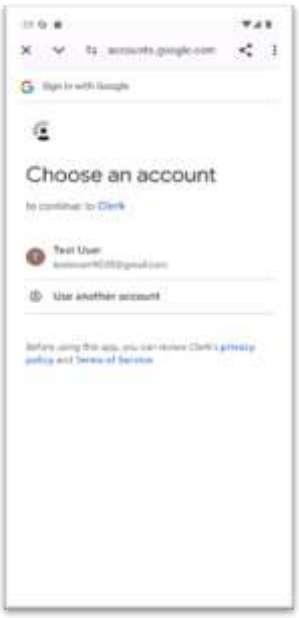

Fig. 4 user authentication using Clerk

This image presents how Clerk offers authentication services to the application. People can access the app securely using their email or a third-party account to enjoy personalised features.

3. Home Section UI:

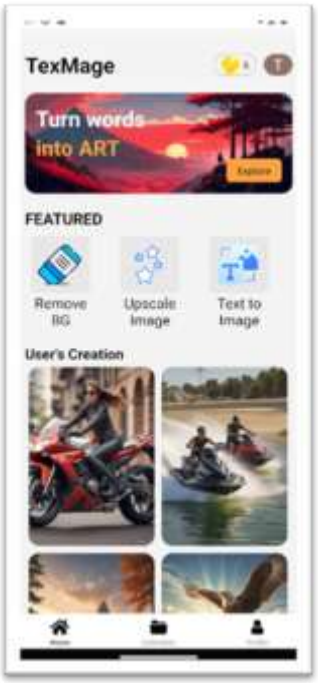

Fig. 5 Home section UI

Users can access all the important parts of the application from the home screen. Users can access the prompt, look through pictures and change their settings with this feature.

4. Text-to-Image Generating Input UI:

Fig. 6 Text-to-Image Generating Input UI

People can use this interface to write text that explains the problem to the virtual assistant. An input field that is empty and an action button make it convenient for users to launch image generation.

5. Displaying the Generated Image:

Fig. 7 Displaying the Generated Image

As soon as the prompt is processed, the resulting image appears to the user. Once an image is generated, you might have choices to save it, share it or look at the full version.

6. Collection Section UI:

Fig. 8 Collection Section UI

All the images made by the user are displayed in this section. The gallery is created by saving entries with their metadata, such as what the user was shown, when they were shown it, and the option to download them.

7. Profile Section UI:

Fig. 9 Profile Section UI

When you open the profile page, it shows you and your settings. It gives you the ability to look at account settings and log out of the app.

Our project stands out by offering a fully mobile-based text-to-image generation experience, unlike most web or desktop-dependent tools. It integrates Replicate's diffusion models with a React Native frontend and a Strapi-powered backend, making it lightweight and accessible. The use of cloud services ensures real-time performance, while planned features like offline access and in-app editing add further uniqueness.

CONCLUSION

We successfully built a mobile app that people can use to create images from text. WITH Replicate.com, we were able to combine powerful diffusion models and use cloud-native tools to make creative content generation simple. In the future, it would be useful to add real-time model fine-tuning, user authentication and enable the use of models tailored to each user.

FUTURE WORK

Enhancements planned in the future hope to change the app into a full creative package. Among the new features are AI image enhancement for greater resolution, a background remover to use images freely and built-in editing options.

REFERENCES

1. C. Bodnar, "Text to Image Synthesis Using GANs," *arXiv preprint arXiv:1803.06176*, 2018.
2. S. Patil, "Text to Image Using Deep Learning," *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, no. 2, pp. 232–238, 2022.
3. S. Ramzan, "Text-to-Image Generation Using Deep Learning," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 6, pp. 101–108, 2022.
4. R. Nanda Kumar, "Text-to-Image Generation Using AI in Android Application," *International Journal of Research in Engineering, Science and Management*, vol. 6, no. 3, pp. 85–90, 2023.
5. L. Sudha, "Semantic Image Synthesis from Text - A Review of GANs and Transformer Models," *IEEE International Conference on Recent Advances in Computer Science*, pp. 231–236, 2024.
6. T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks," *Proc. IEEE CVPR*, pp. 1316–1324, 2018.
7. H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, and D. Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," *Proc. IEEE ICCV*, pp. 5907–5915, 2017.
8. H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, and D. Metaxas, "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1947–1962, 2019.
9. J. Zhu, Z. Zhang, C. Zhang, and B. Zhang, "DM-GAN: Dynamic Memory Generative Adversarial Networks for Text-to-Image Synthesis," *Proc. IEEE CVPR*, pp. 5802–5810, 2019.
10. M. Tao, H. Zhang, X. Huang, and J. Xu, "DF-GAN: Deep Fusion Generative Adversarial Networks for Text-to-Image Synthesis," *arXiv preprint arXiv:2008.05865*, 2020.

11. R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *arXiv preprint arXiv:2112.10752*, 2022.
12. A. Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," *arXiv preprint arXiv:2205.11487*, 2022.
13. J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *arXiv preprint arXiv:2006.11239*, 2020.
14. A. Ramesh et al., "Zero-Shot Text-to-Image Generation," *Proc. 38th International Conference on Machine Learning (ICML)*, vol. 139, pp. 8821–8831, 2021. (DALL·E)
15. B. Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models," *arXiv preprint arXiv:2112.10741*, 2021.
16. P. Esser, R. Rombach, and B. Ommer, "Taming Transformers for High-Resolution Image Synthesis," *Proc. IEEE CVPR*, pp. 12873–12883, 2021.
17. L. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating Images from Captions with Attention," *arXiv preprint arXiv:1511.02793*, 2015.
18. S. Reed et al., "Generative Adversarial Text to Image Synthesis," *Proc. 33rd International Conference on Machine Learning (ICML)*, pp. 1060–1069, 2016.
19. Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes in the Wild," *Proc. IEEE ICCV*, pp. 3730–3738, 2015. (used in T2I datasets)
20. C. Hong, C. Yan, W. Tao, Y. Liu, and Y. Zhao, "Text-to-Image Synthesis via Adversarial Training," *Neurocomputing*, vol. 396, pp. 125–135, 2020