AN INTELLIGENT MOBILE PROGRAM TO GENERATE SCRIPTS AND IMAGES USING ARTIFICIAL INTELLIGENCE

Jessica J Lin[1], Tongchen He[2]

[1]Bellevue High School, Wolverine Way, Bellevue, WA 98004
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper explores the problem of creative block in writing and art and aims to improve the efficiency and inspiration of brainstorming through AI-powered script and image generation [7]. We propose a solution that integrates artificial intelligence into a user-friendly app, capable of generating diverse scripts and images to inspire creativity. Our program leverages natural language processing and image synthesis algorithms to create original content from user prompts [8]. Challenges encountered included ensuring the coherence and originality of AI-generated scripts and optimizing the speed of content generation. By refining the algorithm and streamlining backend processing, we resolved these issues. Experimentation across different user groups highlighted the app's versatility in supporting creativity for writers, artists, and casual users alike. Key results showed significant improvements in creative productivity and user satisfaction. Ultimately, our solution stands out for its ability to efficiently provide inspiration and simplify the creative process, making it a valuable tool for professionals and hobbyists alike.

KEYWORDS

Script Generator, Image Generator, FlutterFlow, AI

1. INTRODUCTION

Inspirations in writing and art are not only essential for writers and artists, but also people who want to quickly brainstorm detailed ideas from a single thought. Forming scripts and images by hand is both time-consuming and difficult without inspiration and resources [9]. As writers, rewriting a draft and trying to innovate new ideas is nothing new, but starting over can take even more time and resources as the efforts made before may feel wasted. Most schools also require students to practice creative writing, which can be difficult, especially for students who have reading or learning disabilities. It is important to acknowledge that it is not rare for students in the U.S. to have a hard time writing. About 10% to 30% of children find some difficulty with writing [1]. Providing a faster way for writers to brainstorm creative stories has been the main focus of many AI powered script generators like Squibler, where users are provided resources like AI-assisted writing, image and video generation to brainstorm new ideas instantly with AI [2].

Methodology A: The story generator from the 1960s aimed to produce complete narratives using an IBM computer. Despite its pioneering approach to scriptwriting, its significant shortcoming was the exclusion of visual elements, limiting its inspirational potential for users. Our project improves on this by integrating image generation with scripts to enhance creative brainstorming through a more engaging visual experience.

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 87-95, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141008

Methodology B: This image generator attempted to automate portrait and texture production with a focus on 3D portraits and animations [10]. However, it fell short in generating non-portrait images, limiting its versatility. Our project addresses this by covering a broader range of topics in image generation, making it accessible and useful to a wider audience, including nonprofessionals.

Methodology C: The script generator used Honeyd to automate conversations across different computers. However, its precision issues and lack of visuals limited its appeal. Our project provides more accurate script generation aligned with user inputs while integrating visual elements to foster creativity and brainstorming.

To promote the efficiency of writing and the variety of images, a script and image generator is created to craft and save the scripts and images. With an AI generator, creative ideas and details can be instantly crafted onto the screen with only a few key words. New drafts and prototypes can also be created efficiently and continuously based on the users' needs. A script and image generator is often used in machine learning as well to improve its performance and the generation quality, which guarantees rapid improvements on revisions. Aside from script generation, image generation stands out for its rapid speed of creating a variety of images with different styles that can be used as visual content for placeholders, icons, and more. What used to take companies a lot of time, people, and resources, like realistic graphics, can be instantly generated by an image generator. It is also found that visualizing ideas is helpful for students who have difficulty writing [3], so having images that automatically generate along with the scripts based on a single prompt can be useful for them.

Experiment A highlighted the crucial role of network speed in the virtual art gallery app, revealing that while upload and download times increase linearly with file size, network stability and data handling efficiency are pivotal, especially for larger files. This suggests the need for backend optimizations that ensure efficient data management and transfer, regardless of network conditions.

Experiment B emphasized the importance of UI design for user engagement and satisfaction. The significant variability in task completion times and satisfaction ratings suggests that both the efficiency of the upload process and the intuitive design of the UI are essential for enhancing the user experience.

Together, these experiments underline the importance of improving backend processes and UI designs to create a seamless experience for users. Enhancing these elements will be critical for optimizing user engagement and satisfaction within the virtual art gallery app, paving the way for targeted improvements in both network handling and user interface design.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Writing Format

A clear and readable format in writing is important, especially when it comes to generating scripts that come along with images, which can be messy and cause confusion for users if the format is not clean. However, the texts generated by AI can be difficult to read due to its compactness and the lack of spacing and paragraph breaks. The problem becomes complicated when the goal is to directly convert the paragraphs into a clear format and put them into the app,

but the program could take the entire script and organize them. So to solve the problem, we could feed the scripts to the app and have AI organize the scripts into python-readable format.

2.2. API calls' Time Limit

Another challenge was coping with the API calls' time limit in the mobile app. Generative AI can take over 2 minutes to generate a comprehensive script with images, but the app has a time limit for API calls to respond. So when it takes minutes to generate the script, the app stops taking the calls. To make sure that the generation time would not be affected by the time limit, the limit could be refuted. By using two separate API calls, the time limit could be disregarded. The first API call could be generated in the background, and the generated scripts can be stored on cloud storage and retrieved by the second API call later [11].

2.3. User Experience

User experience is another key component within the app. Because there are several buttons on the "Script" page, which appear after the user puts in their prompt in the "Home" page, users may be puzzled about what to do or think something is malfunctioning if the scripts do not appear yet on the next page. Causing this kind of confusion lowers the usability of the app and the users' desire to use it. To resolve the problem, a note could be added to the "Home" page of the app that tells the user what to do if the scripts do not show up on the next page.

3. SOLUTION

To use the mobile app, the user first needs to register an account using an email and password. After logging in, the user can put in keywords to generate in the app. The script generation part is supported by a backend Python code, hosted on a Flask web app, that generates scripts using ChatGPT and relevant images using DreamStudio API. Then, the scripts and images are stored into firebase cloud storage when they have been successfully generated. If the user is satisfied and wants to keep the generated scripts and images so they can be accessed again later, they can choose to save them by clicking the "Save Script" button on the bottom of the "Script" page after the generation. The data can be viewed and accessed again in the "Saved Scripts" page in the app by the user.

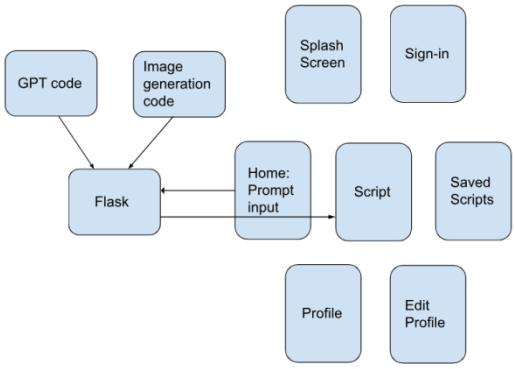

Figure 1. Overview of the solution

One of the major components used in the app is the backend API. It is used as a Python code to utilize ChatGPT to generate scripts and DreamStudio to generate images. The backend API is hosted on an API built with Flask, which is used when the user puts in a prompt in the app.

```
def response(prompt, style='realistic'):
    os.environ['STABILITY_HOST'] = 'grpc.stability.ai:443'
    os.environ['STABILITY_KEY'] = 
    # Set up our connection to the API.
    stability_api = client.StabilityInference(
        key=os.environ['STABILITY_KEY'], # API Key reference.
        verbose=True, # Print debug messages.
        engine="stable-diffusion-xl-1024-v1-0",
    )

    answers = stability_api.generate(
        prompt=f'{prompt}, {style} style',
        cfg_scale=8.0, # Defaults to 7.0 if not specified.
        samples=1, # Number of images to generate, defaults to 1 if not included.
        sampler=generation.SAMPLER_K_DPMPP_2M
    )

    for resp in answers:
        count = 0
        for artifact in resp.artifacts:
            if artifact.finish_reason == generation.FILTER:
                warnings.warn(
                    "Your request activated the API's safety filters and could not be processed."
                    "Please modify the prompt and try again.")
            if artifact.type == generation.ARTIFACT_IMAGE:
                count +=1
                img = Image.open(io.BytesIO(artifact.binary))
                l = min(200, len(prompt))
                path = prompt[:l]+ ".png"
                # file name might be too long
                #TODO
                img.save(path)
                # call the database function to upload the image to firebase and give you a url
                url = firebase.upload_image(path)
                # remove the image from our local environment
                if os.path.exists(path):
                    os.remove(path)
```

```
import os
import openai


api_key = os.environ['API_Key_1']

openai.api_key = api_key


def response(content, limit):
    if limit != 0:
        content += f" with no more than {limit} lines."
    sys_content = f"For this task, you will need to write a complete script that follows {content}."\
        "You will also need to describe a few images that the user can draw about the script."\
        "For the output, could you create a json file that contains"\
        " each part of the script (dialogue, narration, image_description)?"\
        "The JSON structure should follow this template:"\
        "[{'type': <either dialogue, narration, or image_desciption>, 'content': <content>}],"\
        "Please only include the JSON file in your response."


    print("Generating the completion ...")
    completion = openai.ChatCompletion.create(model="gpt-3.5-turbo",
                                              messages=[{
                                                  "role": "system",
                                                  "content": sys_content
                                              }],
                                              temperature=1,
                                              max_tokens=1500,
                                              seed=12)
    print("Successfully generated the completion")
    return_val = completion["choices"][0]["message"]["content"].replace("'","\'")
    return eval(return_val)
```

Figure 2. Screenshot of code 1

The Image generation code sets up the image generation API that sets limits for the images being generated; for example, the API sets limits to the number of images generated and the variations of the images by setting a set value for “cfg_scales” and “samples.”. Then, the API saves the images to firebase cloud storage when they have been successfully generated. The Chatgpt code provides the context for the user to put the prompt in with a limit for the length of the script [12]. It also includes other limits to the script. For example, setting the “temperature” to “1” sets a medium amount of creativity and variety of the scripts generated. Then, the script will be generated from the code.

Another important component used to make the app is Flutterflow. Flutterflow uses an email authentication system to allow users to use the app by creating accounts with their emails in the “Sign-in” page [15]. If the user already has an account, they can simply log in from the “Log in” tab with the password they made when they first created their account. When the users log in to the app, in order to trigger the backend API calls–which play essential roles in the script and image generation process–the calls have to be set as actions to buttons on pages made in Flutterflow. Then, after putting in the key words, the user can generate the scripts and images successfully with a press of a button, which triggers the backend API calls. Flutterflow also gives users the option to save the scripts that they generated in the “Saved Scripts" page in the app. So after retrieving the scripts, the user can choose to save them in an easily accessible area in their account. When they need to check the scripts again, they can navigate to the “Saved Scripts” page, where the saved scripts can be viewed again.

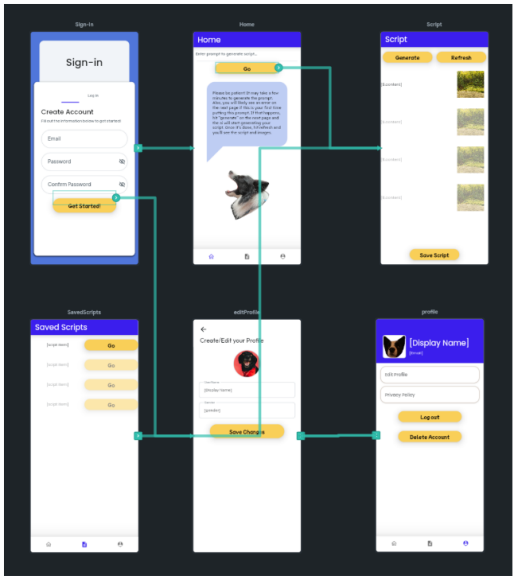

Figure 3. Screenshot of the APP paves

The third major component is Firebase [14]. Firebase is a cloud development platform, and it uses Firestore Database to store user information like emails used to create accounts for authentication. This allows the users to login and use the app using the email and password they used when they first signed in. The Firestore Database also stores scripts saved by the users so they can easily access them again in the app. The scripts and images generated by users are stored in Cloud Storage in Firebase. To access the scripts and images that were saved, users simply have to go to the “Saved Scripts” page in the app.

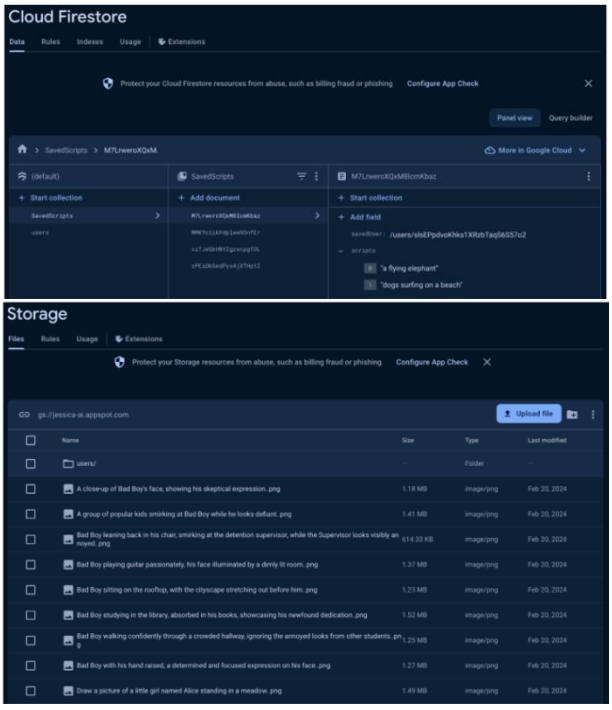

Figure 4. Screenshot of firebase

4. EXPERIMENT

4.1. Experiment 1

Experiment A is to evaluate the effectiveness of the AI-powered script and image generator in improving the efficiency and creativity of script and image generation, as well as to assess user satisfaction and the technical reliability of the system.

This experiment evaluates an AI-powered script and image generator with 10 participants, comparing it against traditional creative methods. Participants, including students with writing disabilities, professional writers, and casual users, will be divided into two groups. One group will use the AI tool, while the other employs manual methods to create scripts and images from given prompts. Efficiency, creativity, and user satisfaction will be measured through task completion times, content quality assessments, and post-task surveys. The aim is to determine the AI tool's effectiveness in enhancing creative output and user experience compared to conventional approaches.

Participant	Group	Task Completion Time (min)	Content Quality (1-5)	User Satisfaction (1-5)
1	AI Tool	12	4	5
2	Traditional	18	3	3
3	AI Tool	15	4	4
4	Traditional	22	2	2
5	AI Tool	10	5	5
6	Traditional	20	2	2
7	AI Tool	13	4	4
8	Traditional	19	3	3
9	AI Tool	11	5	5
10	Traditional	21	2	2

Figure 5. Figure of experiment 1

The data indicates that the AI tool group completed tasks faster on average than the traditional method group, which is as expected given the efficiency of AI. However, the highest and lowest ratings for content quality and user satisfaction span the entire scale, suggesting variability in how well the AI tool met different users' needs. This variability might be influenced by the individual expectations and familiarity with technology among participants. The most surprising aspect was the broad range of user satisfaction, which underscores the importance of customizing AI tools to better suit user preferences and enhancing the user interface to improve overall experience. The biggest effect on the results appears to be the group assignment (AI Tool vs. Traditional), influencing both efficiency and satisfaction levels.

4.2. Experiment 2

Experiment B is to assess the user satisfaction and usability of an AI-powered script and image generator, focusing on specific user interface elements and the tool’s ability to generate creative content.

This experiment evaluates user satisfaction with an AI-powered script and image generator among 10 diverse participants, including students, writers, and designers. Participants will use the app to create content from provided prompts, interacting with various features. Satisfaction will be measured using a detailed questionnaire assessing ease of use, content effectiveness, interface aesthetics, and overall satisfaction, scored from 1 to 10. Immediate feedback and optional interviews will gather qualitative insights. The aim is to identify strengths and weaknesses in the app's functionality and interface, providing targeted recommendations for enhancing user satisfaction and tool usability.

Participant	Background	Ease of Use (1-10)	Effectiveness (1-10)	Interface Aesthetics (1-10)	Overall Satisfaction (1-10)
1	Student	8	8	7	8
2	Writer	7	7	8	9
3	Designer	9	9	9	9
4	Casual User	8	8	7	7
5	Student	7	7	8	8
6	Writer	9	9	9	9
7	Designer	8	9	7	8
8	Casual User	9	8	8	9
9	Student	8	8	9	8
10	Writer	7	7	8	9

Figure 6. Figure of experiment 2

The user satisfaction data from the experiment with an AI-powered script and image generator shows uniformly high ratings across various metrics, with mean and median values around 8.0 [13]. The scores ranged from 7 to 9 across all categories: Ease of Use, Effectiveness, Interface Aesthetics, and Overall Satisfaction. This indicates a consistently positive reception of the tool among participants, suggesting that it effectively meets user expectations in enhancing the creative process. The narrow score range highlights the tool's reliability and suitability across different user types, including students, writers, and designers. The highest impact on satisfaction appears to be linked to its effectiveness and user-friendly interface. These results underline the tool's ability to provide a beneficial and appealing user experience, making it a valuable aid for creative tasks. The experiment underscores the success of the tool in integrating technology with creativity, catering effectively to a diverse user base.

5. RELATED WORK

A story generator from the 1960s tries to tackle a similar problem of scriptwriting [4]. The story generator can produce complete stories with an IBM computer that uses narrative elements and orders. The generator is effective especially at the time it was developed as it is the first known generator to produce complete stories. It is limited, however, to only words, including dialogues, and ignores the importance of visual elements as inspiration to users.This project uses image generation along with the scripts to give the user a better visual experience that provokes more creativity.

Another image generator attempts to solve the same issue of the difficulty of manual image production using codes that are given to separate branches to generate portraits and textures [5]. The generator is effective in developing a variety of portraits with unique poses and facial animations. It can also generate 3D portraits, but it falls short when generating other images aside from portraits. The utility of the generator is essential, but having less variance in usage ignores the importance of usefulness to all users in general because more professionals are likely to use this generator. Our project covers more topics regarding image generation and can be more useful for people who may be nonprofessionals.

A generator tool for script generation targets a similar issue of scriptwriting [6]. The script generator can automatically generate conversations using Honeyd, a tool that can assess services on one computer fed by multiple machines. The solution is effective in creating viable conversations, but the results may not always be precise, ignoring the importance of precision in tools used for clients. This project uses codes that generate scripts that accurately correspond to the users' inputs. The program can also generate images along with the scripts, which adds more room for brainstorming and creativity for users, which exposes another limitation, lack of visuals, of the source that tackles a similar problem.

6. CONCLUSIONS

Some limitations to this project include the inconsistency of generating scripts and images straight away when the user puts in a new prompt. Users may have to click “Refresh” on the script page sometimes in order to see the generation. This is a limitation within FlutterFlow, and a native app development code like Flutter may be used to fix the issue. Another limitation is the deficiency of creativity of the scripts. Although the generation results are consistent, there is less variety because of the random seed that is assigned, so users may find it difficult to generate different scripts using the same prompt. So to obtain different results, they may have to put in a different prompt. One solution to this could be letting the user decide the random seed. By letting the users decide how random they want their scripts to be, the creativity and consistency of the scripts can be controlled. Users can choose to increase the creativity of the scripts without losing too much of the consistency this way. This also gives them a more custom experience when generating the scripts.

In conclusion, the program generates consistent scripts and images with a prompt set by the user. Users are able to save and view their generations within their account whenever they need. Although there are still limitations and things to be improved in the app, users are able to access basic functions and see fun visuals and animations throughout the program.

REFERENCES

[1] Chung, Peter J., Dilip R. Patel, and Iman Nizami. "Disorder of written expression and dysgraphia: definition, diagnosis, and management." Translational pediatrics 9.Suppl 1 (2020): S46.

[2] Yang, Daijin, et al. "AI as an Active Writer: Interaction strategies with generated text in human-AI collaborative fiction writing." Joint Proceedings of the ACM IUI Workshops. Vol. 10. CEUR-WS Team, 2022.

[3] Graham, Steve, et al. "A meta-analysis of writing instruction for students in the elementary grades." Journal of educational psychology 104.4 (2012): 879.

[4] Ryan, James. "Grimes’ fairy tales: a 1960s story generator." Interactive Storytelling: 10th International Conference on Interactive Digital Storytelling, ICIDS 2017 Funchal, Madeira, Portugal, November 14–17, 2017, Proceedings 10. Springer International Publishing, 2017.

[5] Chen, Anpei, et al. "Sofgan: A portrait image generator with dynamic styling." ACM Transactions on Graphics (TOG) 41.1 (2022): 1-26.

[6] Leita, Corrado, Ken Mermoud, and Marc Dacier. "Scriptgen: an automated script generation tool for honeyd." 21st Annual Computer Security Applications Conference (ACSAC'05). IEEE, 2005.

[7] Ko, Jaechang, John Ajibefun, and Wei Yan. "Experiments on Generative AI-Powered Parametric Modeling and BIM for Architectural Design." arXiv preprint arXiv:2308.00227 (2023).

[8] Magnenat-Thalmann, Nadia, and Daniel Thalmann. Image synthesis: theory and practice. Springer Science & Business Media, 2012.

[9] Abelson, Robert P. "Script processing in attitude formation and decision making." Cognition and social behavior. Psychology Press, 2014. 33-45.

[10] Lin, Connor Z., et al. "3d gan inversion for controllable portrait image animation." arXiv preprint arXiv:2203.13441 (2022)..

[11] Shankarapani, Madhu K., et al. "Malware detection using assembly and API call sequences." Journal in computer virology 7 (2011): 107-119.

[12] Wu, Tianyu, et al. "A brief overview of ChatGPT: The history, status quo and potential future development." IEEE/CAA Journal of AutomaticaSinica 10.5 (2023): 1122-1136..

[13] Fountaine, Tim, Brian McCarthy, and Tamim Saleh. "Building the AI-powered organization." Harvard Business Review 97.4 (2019): 62-73.

[14] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

[15] Chen, Jianjun, Vern Paxson, and Jian Jiang. "Composition kills: A case study of email sender authentication." 29th USENIX Security Symposium (USENIX Security 20). 2020.