DEMOCRATIZING PERSONAL WEBSITE CREATION: AN AI-DRIVEN APPROACH FOR EFFORTLESS, COST-EFFICIENT, ANDHIGH-QUALITY WEB PAGE GENERATION

Jason Yang[1], Tiancheng Xu[2]

[1]Central High School, 1700 W Olney Ave, Philadelphia, PA 19141
[2]Computer Science Department, California State Polytechnic University, Pomona,
CA 91768

ABSTRACT

As the world becomes increasingly digitized, the demand for personalized websites is also gradually growing [1]. However, creating a personal website is not an easy task. Generally speaking, people can invest time and effort to learn programming languages for website development, or they can spend money to purchase services fromprofessional web development companies. Both of these methods require significant investment in terms of time or money. To democratize the creation of personal websites, we have developed a program that automatically generates personalized web pages. Users simply need to input instructions in natural language, and the program will generate a personalized website based on the information provided. Websites produced in this manner usually have high quality while simultaneously saving substantial amounts of time and money [2].

KEYWORDS

AI, ChatGPT, Website development

1. INTRODUCTION

There has always been a high barrier of entry to programming a website. With such a variety of developmental tools and languages, each taking years to learn, one finds themselves at a roadblock as to how to start learning [3]. Even when one hires professional developers to do the job for them, it costs money and so the services are inaccessible to the masses. Website builders such as Wix and Wordpress have traditionally been the tools that make the website-building experience accessible to everyone, but many of the features they offer are often locked behind paywalls and their services can quickly become complicated to use [4][5]. The website builders also may not have the ability to modify the website infrastructure or source code, which results in limited changes in functionality for the end user.

Although building a website by oneself is a good way to both learn how to code and produce customizable websites, the method takes far too much time to master and one may not know where to start with the massive amounts of information on the web.
Another way might be to use a website builder. However, this has the opposite problem of not giving as much flexibility to the end user as one might like. Eventually, the end user may have to migrate to a self-hosted or self-made website for enough flexibility.

David C. Wyld et al. (Eds): SIPR, NCWC, BIBC, SOFEA, DSML, ARIA, NLP, CSEN -2023
pp. 213-221, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131917

A last option might be to just hire a professional web designer to do the job. This may be a great choice if one has the money, as it gives a degree of flexibility over the website and possibly the maintenance too. However, to the hobbyist, this is not a great idea, as the price can be extremely expensive.

Our solution is to offer a free tool that allows users to generate a blog-style website with the backing of modern AI tools [6]. Given user input for the topic, pattern, and any specific website alterations (demands), one can generate a full html website in a matter of seconds. By creating an HTML website, and with the usage of demands, one can easily create a great website as well as watch how the HTML changes with each subsequent request [7]. This method of generation is often easier than specifying in a tool what one wishes to build, since any alterations can be explained technically or in terms that are not as technical; both go as words into an AI that will parse it into code. As such the method can tolerate both the new and advanced user, a degree of flexibility not offered in other methods. It also solves the aforementioned problem of accessibility of source code, since the HTML file is directly provided to the user. To a new user, this can be a good way to jumpstart learning how to code by seeing how a change in demand affects the end result of the website as well as the HTML changes involved.

Through development, we decided to give the AI a specific prompt to generate our initial test results, modifying code as necessary to generate a good result. Then, we gave it an unrelated prompt to test if it works on other prompts. In both trials, the GPT model generated good results, giving vibrant websites when asked for and interpreting steps perfectly [8]. This shows that the GPT model works as intended with the program to generate a website, at least for the cases we fed it.

2. CHALLENGES

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

The quality of the AI generation
One major challenge of the program is the quality of the AI generation [9]. Both the text and image generation models are using the neural networks, which is a black-box system. It means the output of the models is not explainable, which makes it hard to debug. For example, we could ask the program to output in a certain format, and we will later code based on that format. If the AI model fails to generate such a format, we will have trouble extracting information. To potentially solve this issue, we can spend more time on designing prompts, and run experiments multiple times to ensure the output is good at a high rate.

How to include images in a website
Another challenge is how to include images in a website, given that the main AI model we will use is ChatGPT, a text generation model. To solve this problem, we can incorporate another AI model, mainly for generating images. Once the images are generated, we will save the local path to those images, and send this path information to the GPT model. In that way, the GPT model can safely assume the images are at certain locations, so it can include the images when writing the website.

How to give the user enough freedom to design the website
Another challenge is how to give the user enough freedom to design the website. There is always a trade-off between automatic and freedom. If the program is absolutely automatic, then the user

has no freedom to have customized demand. If the user has all the freedom to design, then it is not close to any automatic. So we need to find a balance point when writing the program. To give the user some freedom but at the same time make the program automatic, we will ask the user for a website demand, where the user can specify what the website should look like. Without this information, the user is able to make the website more customized.

3. SOLUTION

The software starts with asking for a topic, a pattern and a number. Then it will use the given information to generate sentences and its corresponding prompt. The prompts will be used to generate AI images. The topic and sentences will be used to generate a slogan.
With the generated prompts, an AI image generation model will generate images and download the images to the local machine. We do the same to generate a background image as well.
With all images and all text contents, we send everything to the GPT model again, asking it to write an html file including all information. We save the GPT response to a local file, and that will be the website meeting all user's requirements.

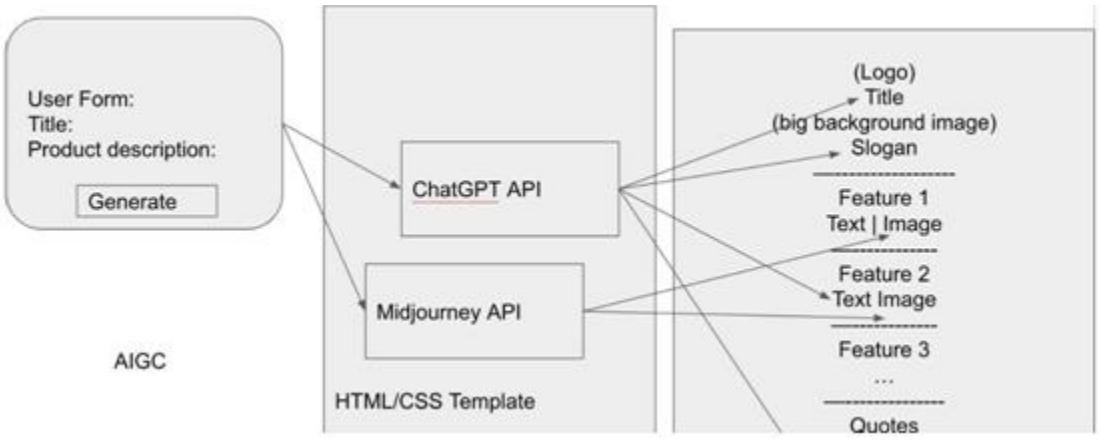

Figure 1. Overview of the solution

An UI for taking user input. The UI is developed in Python using tkinter [10]. It asks for a topic, a pattern, a number, an image style, a background image style and a website demand. Each text entry box has a hint message. A hint message usually is an example of what should be entered. It is displayed when the user opens the software. Once the user focuses on the entry box, the hint message will disappear, and the user can enter information. If the user leaves the entry box with no message inputted, then the hint message will appear again. If the user enters something, then the entry box will have the message the user just entered.

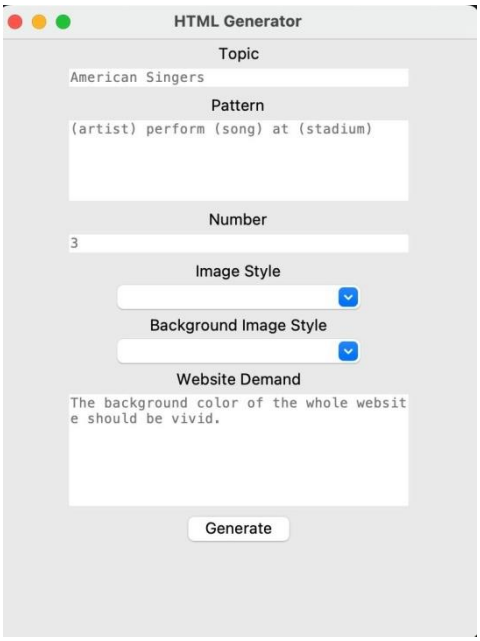

Figure 2. HTML generator

```
def make_entry(placeholder_text, entry_width, entry_height, is_number=False):
    entry = tk.Text(window, width=entry_width, height=entry_height)
    entry.insert("1.0", placeholder_text)
    entry.configure(fg="grey")
    entry.bind("<FocusIn>", lambda event: on_widget_click(event, entry, placeholder_text))
    entry.bind("<FocusOut>", lambda event: on_widget_leave(event, entry, placeholder_text))
    if is_number:
        entry.bind("<KeyRelease>", lambda event: on_key(event, entry))

    entry.pack()

    return entry
```

Figure 3. Screenshot of code 1

In the code, we make a function called make_entry, which takes some parameters and returns the tkinter entry widget. In the function, we use a tkinter widget called Text, which is an entry box with arbitrary size. For the entry box, we defined two functions, on_widget_click and on_widget_leave. Those two functions will be called when the FocusIn and FocusOut events happen. This is where we implement the feature of displaying the hint message.

A function for generating sentences and prompts given a topic and pattern. A pattern will be something like “(artist) perform (song) at (stadium)”. The GPT model will fill in the value in parentheses, generating the sentences. For each sentence, it will generate an AI image generation prompt.

```
def get_sentences_and_prompts(model, topic, pattern, number):
  system_message = f"""
  Your job is to generate {number} sentences by filling out the parentheses in the pattern. \
  The information you generate should be following the topic. \
  For each sentence, you should also write a prompt for generating the corresponding AI
images. \
  To write the prompt, you should follow the following guidelines:
  1 - Be specific and detailed about the image you want the AI to generate.
  2 - Use sensory language to describe textures, sounds, smells, and other sensory details.
  3 - Consider composition and perspective by providing guidelines on camera angles,
framing, and object arrangement.
  4 - Set the mood and atmosphere by describing the desired emotional context of the image.
  5 - Incorporate storytelling elements to give the AI image a sense of narrative or drama.
  6 - Reference existing art or images as inspiration or guidance for the desired aesthetic. \
  You should output a JSON array of objects, where each object corresponds to a sentence
and its prompt.\
  Each object should have its sentence with key "sentence" and its prompt with key
"prompt".
  """

  user_message = f"""
  The topic is {topic}. \
  The pattern is delimited by <>
  <{pattern}>
  """
                response = openai.ChatCompletion.create(
                  model=model,
                  messages=[
                    {"role": "system", "content": system_message},
                    {"role": "user", "content": user_message},
                  ],
                  temperature=0,
                )

                contents = response['choices'][0]['message']['content']
                return contents
```

Figure 4. Screenshot of code 2

In the code, we have a function generating sentences and prompts by taking user's inputs, including topic, pattern and number. The function will ask the GPT model to fill in the missing parts in the pattern following the given topic. On the other hand, it also asks the GPT model to write a prompt for each completed pattern. The prompts are mainly for generating an AI image, which is known for having a good quality if the prompts are detailed enough. To achieve that, we develop an AI image generation prompt guideline. The guideline is detailed and specific for generating high quality AI images.

A function for generating images. It takes prompts and an art style. In this function, we will use an AI image generation model called DreamStudio. The specific engine we use is stable-diffusion-xl-beta-v2-2-2. We set the inference step to 30 for a high quality generation. At the end, we load the generated image using PIL, and save the image and its path to the local machine.

```
def gen_images(prompts, image_style):
  stability_api = client.StabilityInference(
      key=os.environ['STABILITY_KEY'],  # API Key reference.
      verbose=True,  # Print debug messages.
      engine= "stable-diffusion-xl-beta-v2-2-2",  # Set the engine to use for generation.
      # Available engines: stable-diffusion-v1 stable-diffusion-v1-5 stable-diffusion-512-v2-0
stable-diffusion-768-v2-0
      # stable-diffusion-512-v2-1 stable-diffusion-768-v2-1 stable-diffusion-xl-beta-v2-2-2
stable-inpainting-v1-0 stable-inpainting-512-v2-0
  )

  images = []

  for i, prompt in enumerate(prompts):
    answers = stability_api.generate(
      prompt=prompt,
      style_preset = image_style,
      seed=992446758,  # If a seed is provided, the resulting generated image will be

      deterministic.
          # What this means is that as long as all generation parameters remain the same, you can
      always recall the same image simply by generating it again.
          # Note: This isn't quite the case for CLIP Guided generations, which we tackle in the CLIP
      Guidance documentation.
          steps=30,  # Amount of inference steps performed on image generation. Defaults to 30.
          cfg_scale=8.0,  # Influences how strongly your generation is guided to match your prompt.
          # Setting this value higher increases the strength in which it tries to match your prompt.
          # Defaults to 7.0 if not specified.
          width=512,  # Generation width, defaults to 512 if not included.
          height=512,  # Generation height, defaults to 512 if not included.
          samples=1,  # Number of images to generate, defaults to 1 if not included.
          sampler=generation.SAMPLER_K_DPMPP_2M  # Choose which sampler we want to
      denoise our generation with.
          # Defaults to k_dpmpp_2m if not specified. Clip Guidance only supports ancestral
      samplers.
          # (Available Samplers: ddim, plms, k_euler, k_euler_ancestral, k_heun, k_dpm_2,
      k_dpm_2_ancestral, k_dpmpp_2s_ancestral, k_lms, k_dpmpp_2m, k_dpmpp_sde)
        )

      for resp in answers:
        for artifact in resp.artifacts:
          if artifact.finish_reason == generation.FILTER:
            print("Generation failed.")
          if artifact.type == generation.ARTIFACT_IMAGE:
            img = Image.open(io.BytesIO(artifact.binary))
            img.save(str(i)+".png")
            images.append(str(i)+".png")

    return images
```

Figure 5. Screenshot of code 3

In the code above, it has a function named gen_images. It takes prompts and an image style, and returns images and their paths. The number of images is the same as the number of prompts. In the function, we first set up the model using the stability API, which is the API for the image generation model [15]. Then we send the prompts and the image style to the model, asking it to generate images. At the end, we save each image to the local system, and return the paths as the result.

4. EXPERIMENT

4.1. Experiment

In order to ensure that the program initially worked well, we decided to give it a test prompt tobe used throughout development.

Our initial test prompts included the topic of “American Singers”, with a pattern of “(artist) performs (song) at (stadium).”, and various image styles with three images and patterns to generate. We queried the GPT model with a temperature of 0 to discourage any changes in output during development, although that method did not prevent small changes through rounding errors. After our initial successful attempt, we decided to add a website demand to ensure that the color of the website was vivid.

Figure 8. Figure of experiment 1

As shown from the website generated, the images were high quality and the text were representative of the prompts and patterns given. The slogan was well authored and the website was overall well designed. The program listened to the website demand to make it vivid and the topic was generated as well. The slogan was easy to see, and so was the other text on the website. Any other fonts could be easily added through the website demand. It was surprising how the GPT model was able to organize the images well, but interesting how it filled in some blind spots and assumed that the images should be arranged in a row, rather than in a column. There was also an issue with the dividers for each image not being affected by the color of the website.

4.2. EXPERIMENT

Next, we decided to test on a different prompt to ensure that the network was not just trained to answer one type of prompt. We decided on a different number of images this time to ensure variety, but used the same GPT engine.
For the next test, we decided to use the prompt of "Hollywood Movies", with a pattern of "(actor)shoots (movie) with (actress)" and two images instead of three. This time, we specified explicitly that the images should be on the same row, instead of letting the model infer. However, we still used a temperature of 0 to ensure predictability.

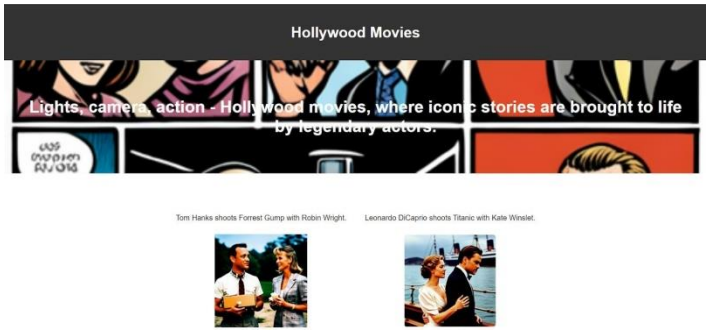

Figure 9. Figure of experiment 2

Interestingly, the model was able to fill in the gaps and create a nice UI for the website. It featured the same banner with two images. However, the slogan text was not as nicely displayed

this time as it was last time, due to the choice of image. This might be a problem since there is no way to determine the text color before the image is generated. One could change the text color, but there is no guarantee that the text will have a solid color that distinguishes itself from the banner image. The model did listen to the request to be on the same row, but it is no longer as vivid as the previous attempt.

5. RELATED WORK

One other way one would build a website would be to learn it themselves [11]. Although one could create a better website this way than using an AI, it is a time-consuming and labor-intensive task, as there would be too much to learn and one would not know where to start. Using a GPT model to generate a website relieves the burden of having to make a website, and allows potential web developers to have a concrete starting point on what to learn. By changing website demands, one would be able see how changes in code affects changes in the website.

Another method of building a website would be using a website builder such as Wix or Wordpress [12]. Although excellent solutions, they fall short in their customizability and accessibility. Most do not allow their internal files to be edited, instead opting for a design that promotes editing of the view. This restricts the advanced user from making any changes they like. Again, using GPT models allows one to make any changes they wish, as the file is on their computer.

Hiring a professional developer to create a website may also be a good alternative to using a GPT model [13]. It gives a lot of flexibility to the user and allows them any level of control they wish. However, this method can be extremely costly, which makes it inaccessible to the hobbyist or person that wishes to learn to build a website. GPT models, on the other hand, promote a similar level of flexibility through the “website demand” input while keeping the price minimal.

6. CONCLUSIONS

Although using a GPT model to generate a website may be a good way for aspiring web designers to learn to code and build a website at the same time, it may not be as great for them later on. GPT models have some inherent limitations thus far that cannot be overcome without more breakthroughs in the field. Websites made by a GPT model can only become so complex before failure. The application is also prone to failure by invalid output from the model, especially with a temperature higher than 0. This could be fixed through trial and error and feeding stronger prompts to the model. Setting the temperature to 0 is also not a wise option since there is much less variation that way. To relieve this, one could “tune” the temperatures throughout each component and see the success rate to find a “sweet spot” for website generation. Though using a GPT model to generate a website has its downsides, there is much to be gained by using a tool like this [14]. As models grow in size and complexity, so does the ability to generate a website. This makes GPT models an adaptive and ever-growing tool to build websites.

7. REFERENCES

[1] Thurman, Neil, and Steve Schifferes. "The future of personalization at news websites: Lessons from a longitudinal study." Journalism studies 13.5-6 (2012): 775-790.

[2] Tang, Yayuan, et al. "Relevant feedback based accurate and intelligent retrieval on capturing user intention for personalized websites." IEEE Access 6 (2018): 24239-24248.

[3] McNamara, Harold M., Beatrice Ramm, and Jared E. Toettcher. "Synthetic developmental biology:

New tools to deconstruct and rebuild developmental systems." Seminars in Cell & Developmental Biology. Vol. 141. Academic Press, 2023.

[4] Williams, Brad, David Damstra, and Hal Stern. Professional WordPress: design and development. John Wiley & Sons, 2015.

[5] Cyr, Dianne, et al. "Exploring human images in website design: a multi-method approach." MIS quarterly (2009): 539-566.

[6] Pimenov, Danil Yu, et al. "Artificial intelligence systems for tool condition monitoring in machining: Analysis and critical review." Journal of Intelligent Manufacturing 34.5 (2023): 2079-2121.

[7] Levering, Ryan, and Michal Cutler. "The portrait of a common HTML web page." Proceedings of the 2006 ACM symposium on Document engineering. 2006.

[8] Floridi, Luciano, and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences." Minds and Machines 30 (2020): 681-694.

[9] Wu, Fei, et al. "Towards a new generation of artificial intelligence in China." Nature Machine Intelligence 2.6 (2020): 312-316.

[10] Lundh, Fredrik. "An introduction to tkinter." URL: www. pythonware. com/library/tkinter/introduction/index. htm (1999).

[11] Cao, Xiang, and Wenhua Yu. "Using content management system joomla! to build a website for research institute needs." 2010 International Conference on Management and Service Science. IEEE, 2010.

[12] Arafin, Md Shamsul, and Yi Jiang. "Developing a dynamic website using the online website builder Weebly for Viking Fortune Oy." (2017).

[13] Zhang, Min, and Juntao Li. "A commentary of GPT-3 in MIT Technology Review 2021." Fundamental Research 1.6 (2021): 831-833.

[14] Hendy, Amr, et al. "How good are gpt models at machine translation? a comprehensive evaluation." arXiv preprint arXiv:2302.09210 (2023).

[15] McDonnell, Tyler, Baishakhi Ray, and Miryung Kim. "An empirical study of api stability and adoption in the android ecosystem." 2013 IEEE International Conference on Software Maintenance. IEEE, 2013.