A USER-FRIENDLY AND AI-BASED MOBILE APPLICATION TO ASSIST IN TRIP PLANNING USING LARGE LANGUAGE MODELS

Ka Ling Gou[1], Tongchen He[2]

[1]Orange County School of the Arts, 1010 N Main St, Santa Ana, CA 92701
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper introduces the Utrip app, explaining various functions and the codes involved in this app. We solved problems like music recommendations and AI chatbox, making sure that they ran without getting errors [1]. So, the user can use it to enhance their travel experience. One of the problems that we encounter is that the GPT model we used does not naturally return a JSON format for our app to read [2]. Therefore, we need to engineer our prompt to specifically ask for a response in a designated JSON format, and parse the response string to convert it to JSON. In our experiment, we tested out the rate of error while using different prompt formats for the GPT model [3]. We found out that the prompt that specifies a JSON format in English words is the best. We also compared our app to ones made by others. Our app allows users to create their own travel plans, and the AI chatbox provides suitable suggestions based on real-time scenarios.

KEYWORDS

Trip Planning, Artificial Intelligence, Large Language Model (GPT-4), AI Chatbot

1. INTRODUCTION

Over 93% of Americans are planning to travel within the next 6 months. Traveling can often be a complicated process for people. It is crucial for people to make a comprehensive plan ahead of their trips in order to avoid nasty situations. We often travel with our family and friends, but planning a trip with your companions can also be hard. For example, you made a travel plan for yourself and your friends, but after you have shown it to them, they don't agree, because they all have different opinions on the plan. This is due to a lack of communication while making travel plans. Therefore, we are planning to create a mobile application to solve these problems [4].

To solve the problem, we are planning to build a mobile app that helps people create vacation plans, browse places of interest, and ask questions about a specific trip. We decided to build a mobile app because this is the most convenient way for the user, as people bring their smartphones with them all the time.

First, it helps to keep track of plans. When people are on a trip, there will be a lot of plans for them to do during the vacation, so making sure they do not miss every event they want to attend is important. To prevent the users from forgetting their plan, we are planning to create a reminder page that keeps track of all the plans that the user has made. Also, it is to set up and update your plans on a mobile device. We are focusing on making the app user-friendly and easy-to-use.

David C. Wyld et al. (Eds): DSCC, AI&FL, SESBC, CSE– 2024
pp. 105-1112, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141610

Compared to a website, the user experience on a mobile device can be a lot more responsive. It also does not require internet connection, so you can always access your plans whether or not you have a signal.

Additionally, communication takes a big part when speaking of making a peaceful and happy trip. We want to incorporate a function that allows users to add their friends and talk freely about their plans. Users who will travel with others will never feel that they are making the plan alone, they can always talk to the people who they will travel with. This ensures communication that other trip planning tools do not offer.

We designed an experiment to test out the effectiveness of different prompts for the AI to respond with the correct JSON format. In this experiment, we built three prompts, each asking the same question, but the difference is how they ask for the response format. The first one includes the JSON format in English words, the second one provides an example of how the actual JSON could look like, and the third one added a pair of triple ticks for the AI to better distinguish the code from other words. We repeatedly sent each prompt to the AI 50 times, and recorded the percentage of times where either the response contains any invalid characters, or it is in a bad format. To our surprise, we found that the first message, where we write out the format in English words, performs the best. There were only 4% of the times where the response was in a bad format, compared to the other messages which have error rates around 20%. We determined why the result occurred was because large language models, such as GPT-4 we used, understands human languages better than code [5].

2. CHALLENGES

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

2.1. Store Messages and Images

One of the challenges of making a mobile application is when users create chats with each other, we need a place to store messages and images in order to keep track of their chat history. We want to keep their information in a safe place to avoid privacy concerns. We also need to store the plans that they made as well as their reviews on a location. When they try to access these information, we also need to make sure that they can be retrieved instantly for the best user experience. To solve this problem, we could use cloud storage to keep all the data on a safe cloud platform, such as Google Firebase.

2.2. Implementing the Blood Alcohol Content Estimate

When recommending music, we do not have in-depth knowledge of the culture of every location that we included in the app. This makes it challenging to manually select a list of music for each city. We could use AI to sort out the information of the popularities of different songs in each country, which allows us to give a list of popular songs to the users based on where they are traveling.

2.3. Access a list of Music Recommendations

Another problem we encounter is when we try to let the users be able to access a list of music recommendations. When users click on the button that says music recommendation, it shows a list of music and the artists next to them. However, a large language model like GPT does not naturally return responses in a list format. In order to show users lists, we could engineer the

prompt that we sent to the GPT API to have it return its response in a list format, and then retrieve the list by parsing and evaluating the response string we get.

3. SOLUTION

Our program is composed of a mobile app (UI), a backend python flask server, and a cloud database powered by Google Firebase [6].

We used FlutterFlow, an interactive web app, to build our mobile app with user interface [7]. The app contains the following pages: login, creating vacation plans, searching and viewing cities around the world with music recommendation, add and chat with friends, speech-to-text generation with translations, and real time assistant with AI. Specifically, the music recommendation, speech-to-text with translation, and real time assistant pages are supported by our Python server [8].

In Python, we connected to the OpenAI API and used generative AI to provide us with music recommendations, transcriptions of audio, and chat conversations with an AI assistant [9]. In order to connect our Python code with FlutterFlow, we built a Flask app to establish a connection. We also used the cloud database from Google Firebase to store the data and information we need. For example we store the information of different attractions and chat boxes users used to contact each other. It saves user's most recent edits, like the plans and reminders they make. We also used it for authentication purposes like creating accounts and resetting passwords.

The flow of our program will be the following. The user will first need to create an account on the mobile app, and login to get access to other features. Then, he can browse different attractions, and create vacations which then get stored in the firebase database. He can also look for music recommendations or ask the assistant a general question about his trip. This request will be sent to the Python Flask app, which connects with the OpenAI API. The Gen AI will generate a response and send it back to the mobile app.

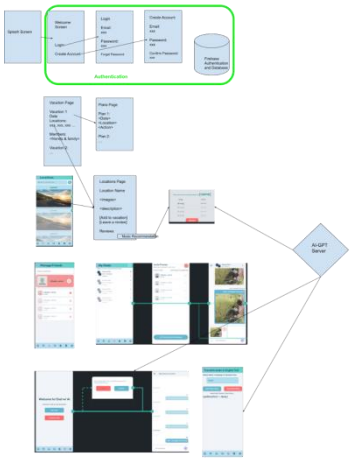

Figure 1. Overview of the solution

The first component that we established is our mobile app built on FlutterFlow.

It aims to make better and more efficient plans for travel, and it provides a platform for users to interact. For the users to log in successfully, authentication is required. The users would need to

provide an email address and create their passwords, and we used Firebase to verify authentication information.

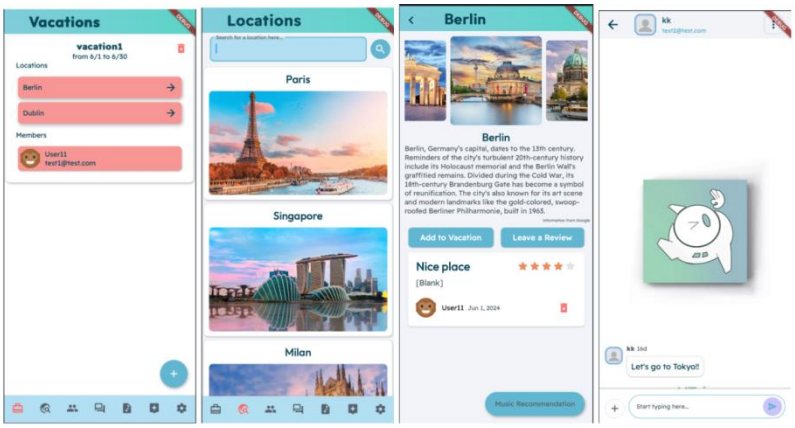

Figure 2. Screenshot of the APP

Utrip has a variety of features to enhance the user's travel experience. In the Vacation page, the users can create their own travel plan, and contact their friends. Also, users can change the background image freely and send text and photos through the chat. Next, there is a locations page, which shows the users different places around the world, and suggests where to travel. For each location, there is a brief introduction and a section where the users provide feedback on it. The users can also get a list of recommended songs by clicking the button in the right corner.

The second component that we built is our python program. It supports several features that are used in our flutterflow app, including music recommendation, speech to text translation, and real-time AI assistant. We utilized the OpenAI GPT API for these features, and built the functions on a flask API in order to connect with our FlutterFlow app.

```
// 20240622121454
// https://2242c06b-b25a-4208-ae83-c9333797e66b-00-30r01srqfu4mh.picard.replit.dev/music/tokyo/6

[
  {
    "artist": "Owl City",
    "song": "Tokyo"
  },
  {
    "artist": "Midnight Juggernauts",
    "song": "Lost in Tokyo"
  },
  {
    "artist": "Teriyaki Boyz",
    "song": "Tokyo Drift"
  },
  {
    "artist": "Madeintyo",
    "song": "Made in Tokyo"
  },
  {
    "artist": "Digital Farm Animals",
    "song": "Tokyo Nights"
  },
  {
    "artist": "Former Airline",
    "song": "To Live and Die in Tokyo"
  }
]
```

Figure 3. Screenshot of code 1

```python
@app.route('/')
def home():
  return "Welcome to my Flask app"


@app.route('/music/<dest>/<num>')
def music(dest, num):
  return jsonify(gen_music(dest, num))


@app.route('/transcription/<lang>', methods=['POST'])
def transcription(lang):
  path = request.form['recording']
  # downloading file from "path" and store it as "recording.mp3" locally
  urllib.request.urlretrieve(path, "recording.mp3")
  text = transcribe('recording.mp3', lang)
  os.remove('recording.mp3')
  return jsonify(text)


@app.route('/help', methods=['get'])
def help():
  # request.args.to_dict(): {"messages": [], "last_message": ""}
  messages = request.args.to_dict()['messages']
  last = request.args.to_dict()['last_message']
  return jsonify(realtime_assist(messages, last))
```

Figure 4. Screenshot of code 2

Our Python Flask app integrates the functions to improve user interaction with the app [10]. The "music" route allows users to receive recommended music based on location. It uses the ChatCompletion function of OpenAI API to generate a list of dictionaries with songs and their artists. The "transcription" route transcribes audio into text and translates it into English. We first retrieve the uploaded audio from a link, and then send it to OpenAI's audio translation function. The "help" route acts as a real-time assistant. It provides information and responses to users. When the user sends in a prompt, we will include both the latest message and the previous chat history for the AI to give the best possible response. Each route covers definite functionalities that deliver different content and services to users.

In order to store the information and data generated by the user, we used Google Firebase as a cloud-based database and storage. In Firebase, we created a few database collections to store data such as user information, chat history, location information, and vacation plans that users created. We also used the Authentication method to enable email-password logins.

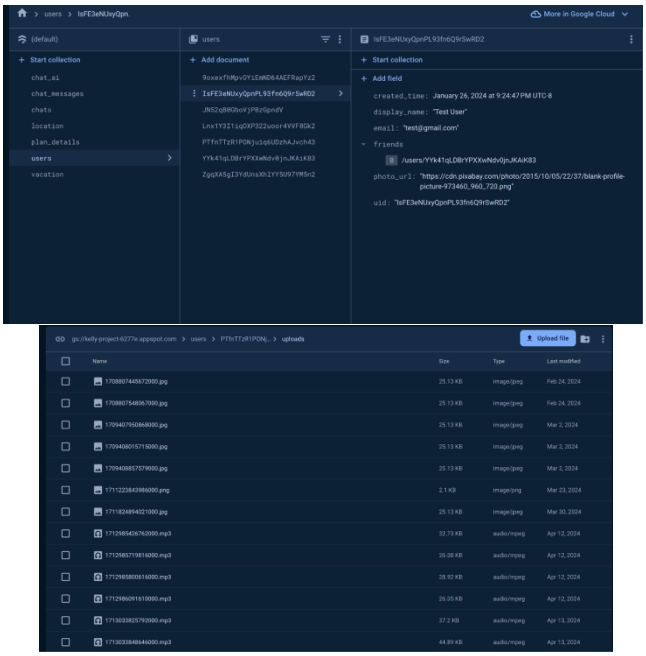

Figure 5. Screenshot of Firebase

Figure 6. Screenshot of code 3

In the first screenshot, we can have a glance at the data collections that we created in Firebase. As an example, the users collection contains each user's id, email address, friends list, profile pic url, display name, and so on. We can also see the collections for vacations, plan details, chat messages, and location information. We then set up different rules for creation, reading, writing, and deletion access for different collections, as shown in the code example screenshot. We also used the cloud storage feature to store the images (profile pic, chat images, chat background, etc.) and the audio that the user uploaded for our speech-to-text feature.

4. EXPERIMENT

One of the problems with large language models like GPT-4 is that it is sometimes inconsistent. It could provide responses in incorrect formats even if the user specifies the format in the prompt. This could be problematic because it would cause occasional errors in our flutterflow app if the response is in a bad format.

We designed an experiment that tests out a few different messages and their effectiveness of keeping the AI response in a correct format.

In our first message, we write out the format that we want to receive in words, as follows: "You need to output your answer as a list of dictionaries, where each dictionary contains the keys 'song' and 'artist'". (message 1)

In our second message, we provide an example for the correct JSON format in our prompt, as follows: "You need to output your answer as a JSON object following this structure: [{'song': <song_name>, 'artist': <artist_name>}]". (message 2)

In our third message, we added a pair of triple ticks around the JSON example, as follows: "... following this structure: ```[{'song': <song_name>, 'artist': <artist_name>}]```". (message 3)

For each message, we measure the number of times when the response contains any invalid characters, and the number of times when the response fails to contain the correct format.

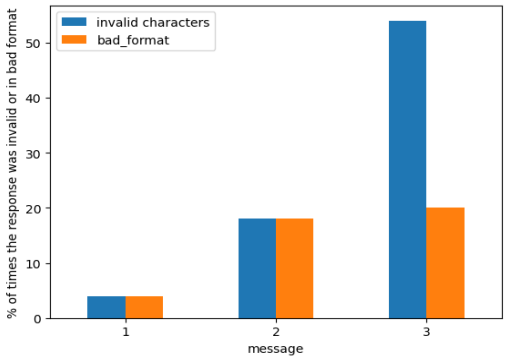

Figure 7. Figure of the experiment

Based on our experiment and visualization, message 1 was the most effective in terms of keeping the response in the correct format, with only 4% of the responses in a bad format or containing any invalid characters. The second and third message both had a similar number of badly formatted responses, 18% and 20%. However, the third message had over 50% of the responses containing invalid characters. This is likely to be the three ticks that we included in the question, but these can be removed by string parsing so it will not affect the effectiveness of our program. It is rather counterintuitive that if we included the JSON examples (in message 2 and 3), the error rate will go much higher. This might be an indication that GPT-4 understands human language better than code examples. Therefore, we decided to use the first message as our prompt for the AI model.

5. RELATED WORK

Trip-mine, created by Lu, Lin, and Tseng, provides a novel data mining-based approach, to efficiently find the optimal trip which satisfies the user's travel time constraint based on the user's location [11]. This research used machine learning techniques to create a trip plan for the user, where our Utip app lets the users create their own plan based on the locations they want to visit. The use of optimization mechanisms in Trip-mine takes travel time constraints into account, which improves the efficiency and effectiveness of trip planning, but at the same time, it ignores users' own preferences, which we highlighted in our Utrip app.

Photo2Trip, created by Xin Lu, Changhu Wang, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang [12]. It suggests customized travel routes using geo-tagged photos from www.panoramio.com. It also creates this based on users' preferences. This feature of the app improves the efficiency and accuracy of planning a travel guide, which saves a lot of time for users. However, using Utrip, the user is able to create their travel plan themselves and there are more functions in the app other than travel planning. Our app provides translation and chat box with AI between users.

Tourist Trip Design Problem, created by Tommaso Adamo, Lucio Colizzi, Giovanni Dimauro, GianpaoloGhiani, and Emaneula Guerriero [13]. It provides the opening duration of attractions and road network. It enhances efficiency using iterated local search, and it optimizes tourist satisfaction. The app may not work well since the environment is unpredictable in real situations, which can make travel plans less accurate. Our app provides a chatbot with AI that may be helpful in various real-time scenarios, therefore suitable for unpredictable situations.

6. CONCLUSIONS

The limitations of this project are that it may run into errors when using the AI chatbox. Also, it only works when it's connected to the Internet, so the user cannot access it anywhere. It does not

offer personalized routes for the user, which also may be a problem when users travel in unfamiliar places. Currently, we are using the regular GPT model, but one improvement could be building a customized GPT with enhanced capability of performing as a travel guide [14]. If I were to improve this, I would create a page that allows the user to make their own travel route, and they may ask AI to create a travel route for them by listing out their needs.

This travel app is designed to make trip planning more convenient and more fun [15]. With feature like event reminders, it reminds the user of every activity, making sure that no activity is missed. The AI chatbox also provide real-time assistance and suggestions based on their questions.

REFERENCES

[1] Battineni, Gopi, Nalini Chintalapudi, and Francesco Amenta. "AI chatbot design during an epidemic like the novel coronavirus." Healthcare. Vol. 8. No. 2. MDPI, 2020.
[2] Peng, Dunlu, Lidong Cao, and Wenjie Xu. "Using JSON for data exchanging in web service applications." Journal of Computational Information Systems 7.16 (2011): 5883-5890.
[3] Hendy, Amr, et al. "How good are gpt models at machine translation? a comprehensive evaluation." arXiv preprint arXiv:2302.09210 (2023).
[4] Islam, Rashedul, Rofiqul Islam, and Tohidul Mazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.
[5] Nori, Harsha, et al. "Capabilities of gpt-4 on medical challenge problems." arXiv preprint arXiv:2303.13375 (2023).
[6] Neil, Theresa. Mobile design pattern gallery: UI patterns for smartphone apps. " O'Reilly Media, Inc.", 2014.
[7] Fadaee, Mohsen. "Building the foundation for an ai-integrated career coaching application with flutterflow." (2024).
[8] Taneja, Sheetal, and Pratibha R. Gupta. "Python as a tool for web server application development." JIMS8I-International Journal of Information Communication and Computing Technology 2.1 (2014): 77-83.
[9] Tingiris, Steve, and Bret Kinsella. Exploring GPT-3: An unofficial first look at the general-purpose language processing API from OpenAI. Packt Publishing Ltd, 2021.
[10] Chan, Jack, Ray Chung, and Jack Huang. Python API Development Fundamentals: Develop a full-stack web application with Python and Flask. Packt Publishing Ltd, 2019.
[11] Lu, Eric Hsueh-Chan, Chih-Yuan Lin, and Vincent S. Tseng. "Trip-mine: An efficient trip planning approach with travel time constraints." 2011 IEEE 12th International Conference on Mobile Data Management. Vol. 1. IEEE, 2011.
[12] Lu, Xin, et al. "Photo2trip: generating travel routes from geo-tagged photos for trip planning." Proceedings of the 18th ACM international conference on Multimedia. 2010.
[13] Ruiz-Meza, José, and Jairo R. Montoya-Torres. "A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines." Operations Research Perspectives 9 (2022): 100228.
[14] Zhang, Min, and Juntao Li. "A commentary of GPT-3 in MIT Technology Review 2021." Fundamental Research 1.6 (2021): 831-833.
[15] Dickinson, Janet E., et al. "Fundamental challenges in designing a collaborative travel app." Transport Policy 44 (2015): 28-36.