Fall 2023 - Quiz App GPT Chatbot

Project Overview

Overall Problem:

Students needing additional support in understanding concepts and receiving relevant information for the course while using QuizApp.

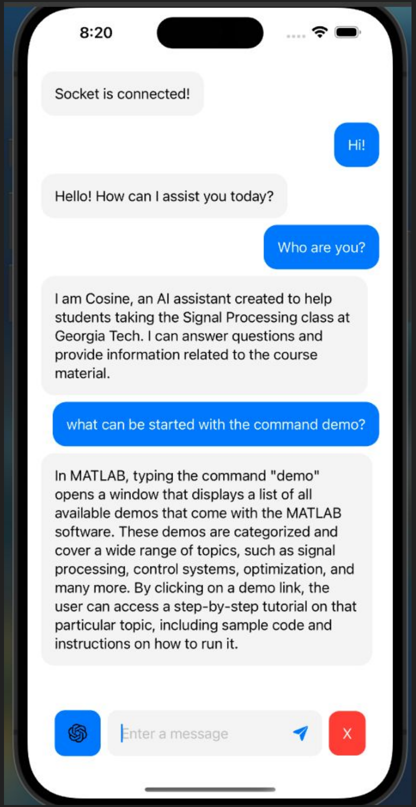
Project Overview

Semester Goals:

- Fine Tune the model with class specific information using GPT’s Function calling and model training.
- Make every chatroom for each student private. Thus students will get a personalized tutoring experience.

NLP GPT model used for the chatbot

- We are using the GPT 3.5 API provided by openAI, which is a very powerful conversational model.
- It has the ability to take an initial input, which makes it convenient to specify what types of output we'd like.



Socket is connected!

Hi!

Hello! How can I assist you today?

Who are you?

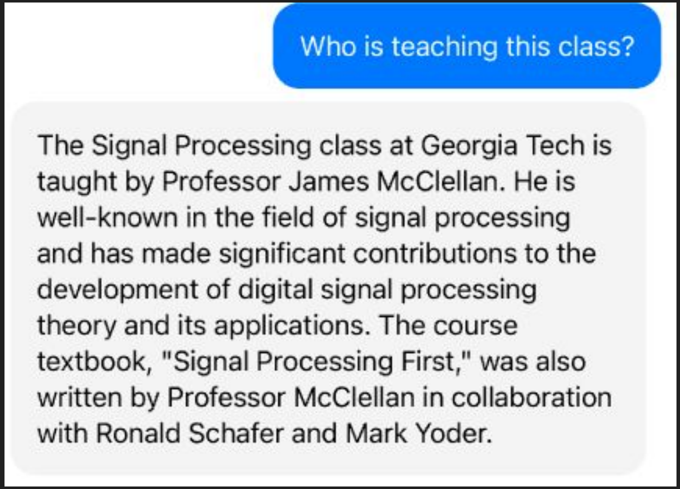
I am Cosine, an AI assistant created to help students taking the Signal Processing class at Georgia Tech. I can answer questions and provide information related to the course material.

what can be started with the command demo?

In MATLAB, typing the command "demo" opens a window that displays a list of all available demos that come with the MATLAB software. These demos are categorized and cover a wide range of topics, such as signal processing, control systems, optimization, and many more. By clicking on a demo link, the user can access a step-by-step tutorial on that particular topic, including sample code and instructions on how to run it.

NLP GPT model used for the chatbot

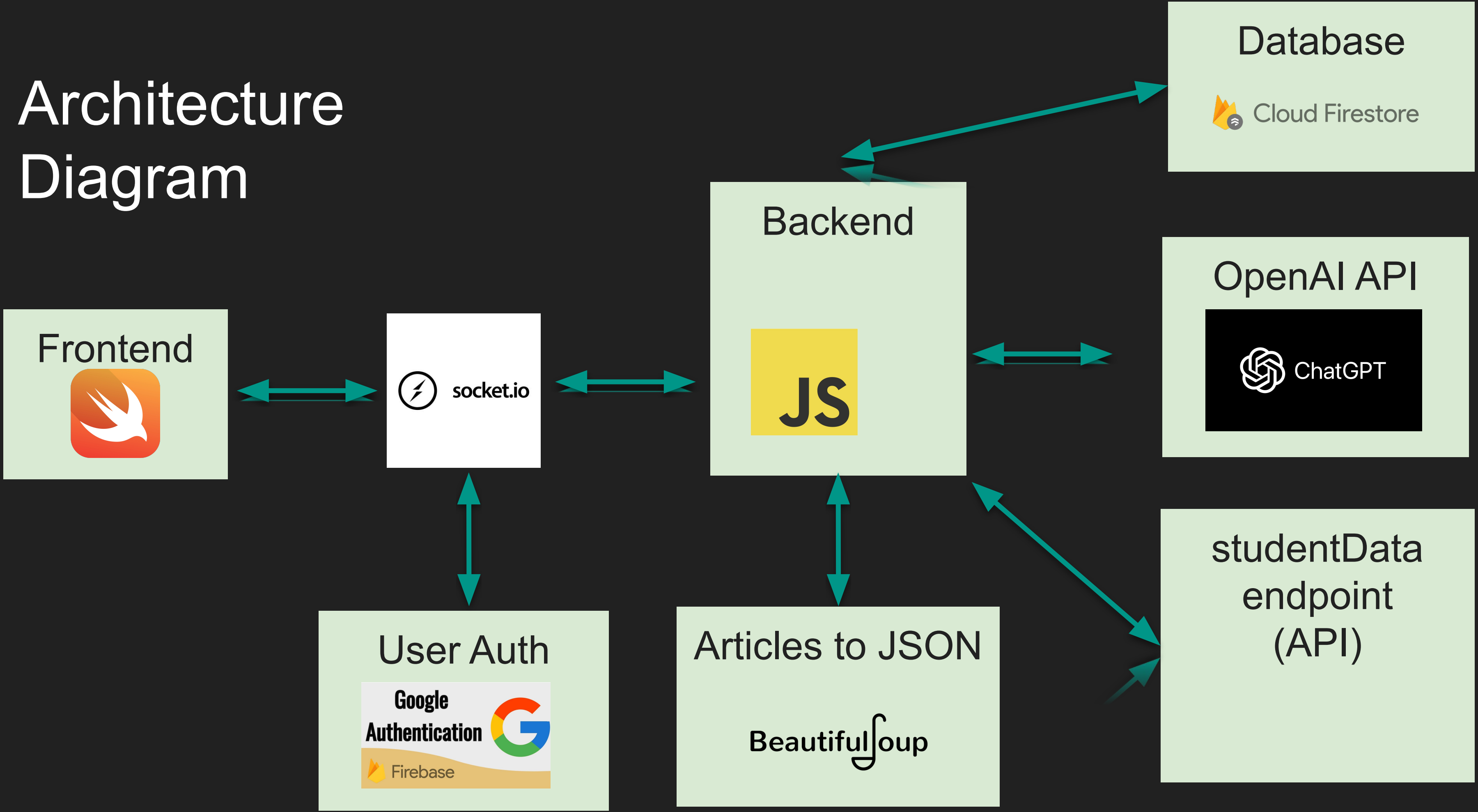
- In addition to answering questions relevant to course material, the chatbot can take additional information in the input such as course logistics and answer them as well.



Who is teaching this class?

The Signal Processing class at Georgia Tech is taught by Professor James McClellan. He is well-known in the field of signal processing and has made significant contributions to the development of digital signal processing theory and its applications. The course textbook, "Signal Processing First," was also written by Professor McClellan in collaboration with Ronald Schafer and Mark Yoder.

Architecture Diagram



Frontend

socket.io

Backend

JS

Database

Cloud Firestore

OpenAI API

ChatGPT

studentData endpoint (API)

User Auth

Google Authentication

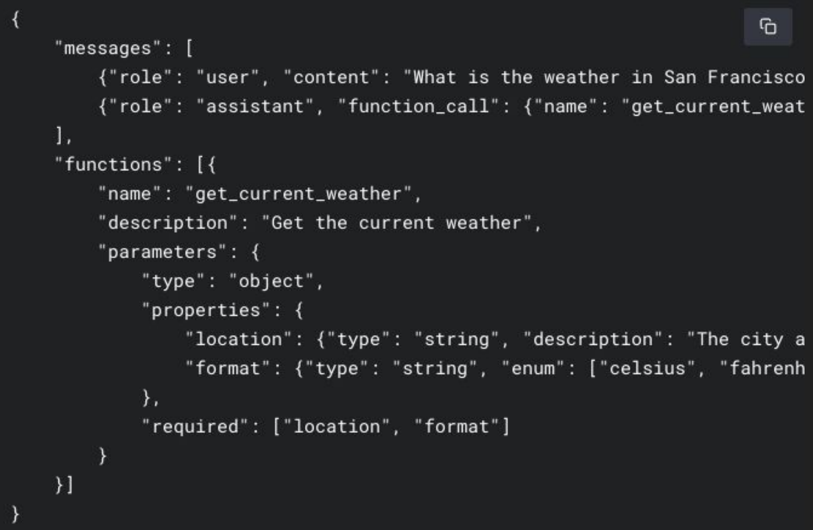
Firebase

Articles to JSON

BeautifulSoup

GPT Function calling

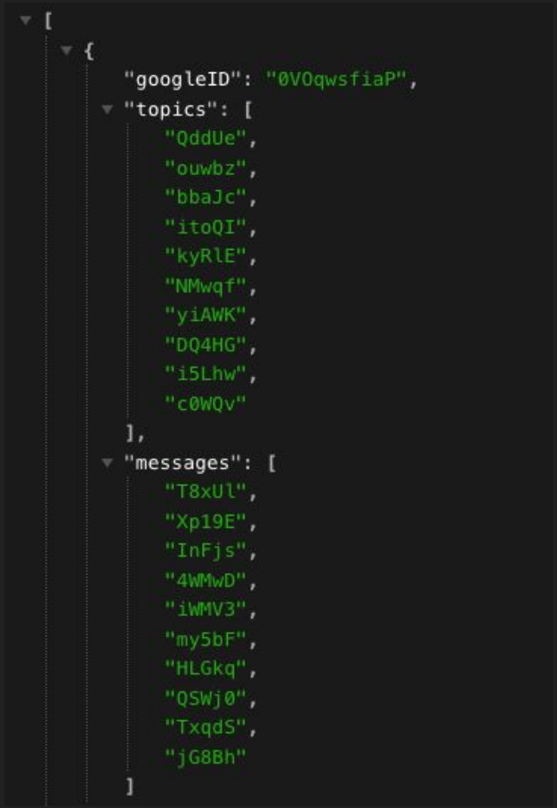
- By adding necessary functions in the format below for every API call, GPT can use it to get information that can not normally be accessed.
- We can use data such as textbook, and Piazza dataset in the function calling to serve this purpose.



```
{
    "messages": [
        {"role": "user", "content": "What is the weather in San Francisco
        {"role": "assistant", "function_call": {"name": "get_current_weat
    ],
    "functions": [{
        "name": "get_current_weather",
        "description": "Get the current weather",
        "parameters": {
            "type": "object",
            "properties": {
                "location": {"type": "string", "description": "The city a
                "format": {"type": "string", "enum": ["celsius", "fahrenh
            },
            "required": ["location", "format"]
        }
    }]
}
```

Access to topics of conversation through endpoint

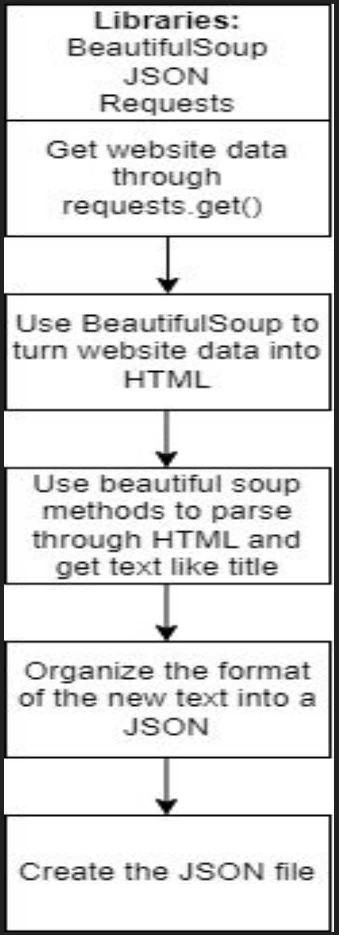
- We can configure the backend to constantly append to the list of topics the student is currently talking about in the chat.
- The instructors can access this information through the “/studentData” endpoint like the example shown on the right.



```
[
  {
    "googleID": "0VOqwsfiaP",
    "topics": [
      "QddUe",
      "ouwbz",
      "bbaJc",
      "itoQI",
      "kyRlE",
      "NMwqf",
      "yiAWK",
      "DQ4HG",
      "i5Lhw",
      "c0WQv"
    ],
    "messages": [
      "T8xUl",
      "Xp19E",
      "InFjs",
      "4WMwD",
      "iWMV3",
      "my5bF",
      "HLGkq",
      "QSWj0",
      "TxqdS",
      "jG8Bh"
    ]
```

Creating JSON files from articles

- This code can be used to parse through articles and convert them into JSON files
- Sections are assigned numbers based on appearance in article
- The example is used is the provided ChatGPT article
- Within the chatBot, there is a function that will call upon the parsed file to quote the article.



Libraries:
BeautifulSoup
JSON
Requests

Get website data through requests.get()

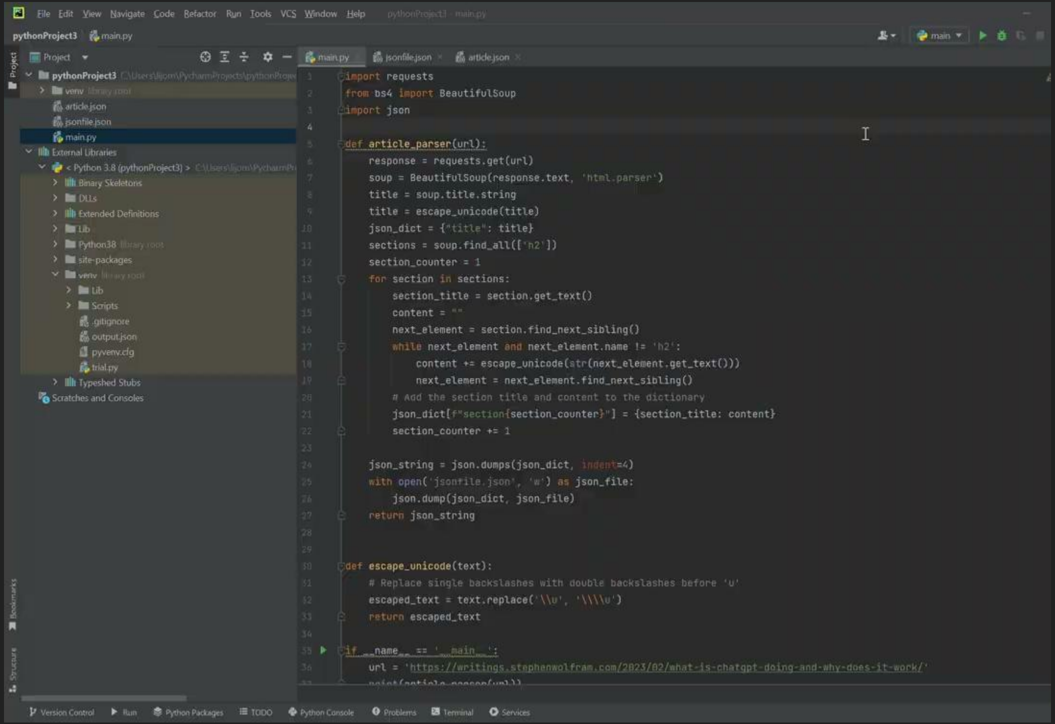
Use BeautifulSoup to turn website data into HTML

Use beautiful soup methods to parse through HTML and get text like title

Organize the format of the new text into a JSON

Create the JSON file

Demo



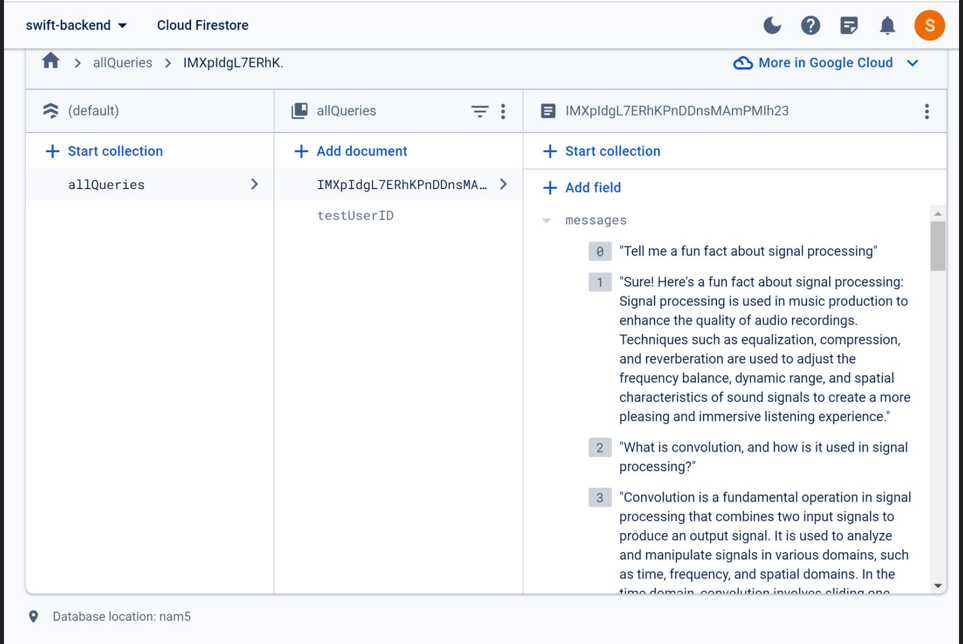
```python
import requests
from bs4 import BeautifulSoup
import json

def article_parser(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    title = soup.title.string
    title = escape_unicode(title)
    json_dict = {"title": title}
    sections = soup.find_all(['h2'])
    section_counter = 1
    for section in sections:
        section_title = section.get_text()
        content = ""
        next_element = section.find_next_sibling()
        while next_element and next_element.name != 'h2':
            content += escape_unicode(str(next_element.get_text()))
            next_element = next_element.find_next_sibling()
        # Add the section title and content to the dictionary
        json_dict[f"section{section_counter}"] = {section_title: content}
        section_counter += 1

    json_string = json.dumps(json_dict, indent=4)
    with open('jsonfile.json', 'w') as json_file:
        json.dump(json_dict, json_file)
    return json_string


def escape_unicode(text):
    # Replace single backslashes with double backslashes before 'u'
    escaped_text = text.replace('\\u', '\\\\u')
    return escaped_text

if __name__ == '__main__':
    url = 'https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/'
    print(article_parser(url))
```

Why chat privatization?

- Private chats is an industry-standard feature for chat apps
- Students can tell which past questions were written by them
- Students can read the answers to their past questions
- Better user experience
- Opportunities for future development

User Chat History

- Users can login using Google identity provider. Thru Firebase Authentication, we can see all registered users, each with a User UID
- In Google Firestore, we save a user profile per UID, containing their previous messages
- Can be useful for detecting student-specific areas of improvement



messages

0 "Tell me a fun fact about signal processing"

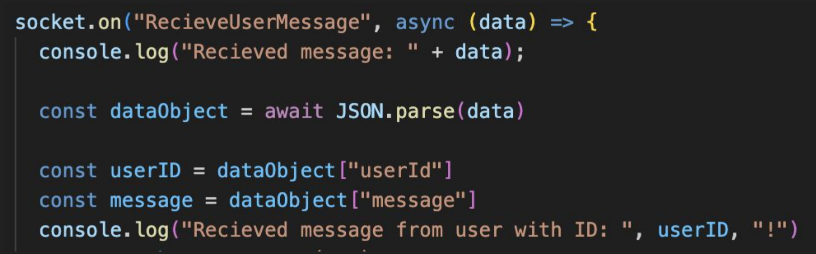
1 "Sure! Here's a fun fact about signal processing: Signal processing is used in music production to enhance the quality of audio recordings. Techniques such as equalization, compression, and reverberation are used to adjust the frequency balance, dynamic range, and spatial characteristics of sound signals to create a more pleasing and immersive listening experience."

2 "What is convolution, and how is it used in signal processing?"

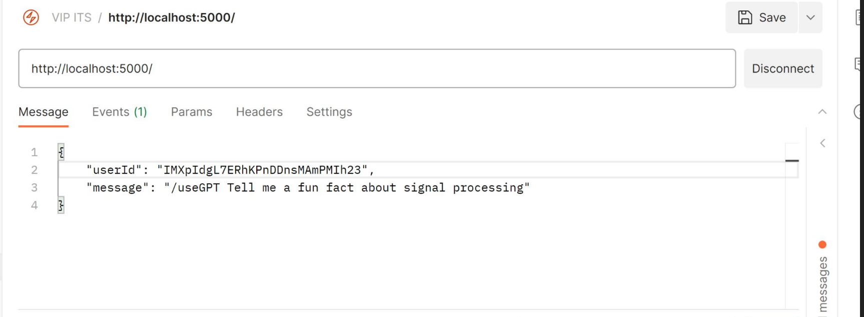
3 "Convolution is a fundamental operation in signal processing that combines two input signals to produce an output signal. It is used to analyze and manipulate signals in various domains, such as time, frequency, and spatial domains. In the

Privatizing Users with userID

- When a message is sent, it’s sent as a JSON
- JSON is parsed
- Variables will be processed in backend



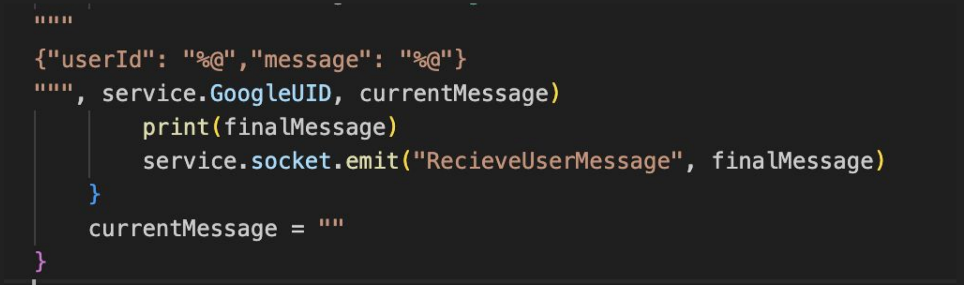
```
socket.on("RecieveUserMessage", async (data) => {
  console.log("Recieved message: " + data);

  const dataObject = await JSON.parse(data)

  const userID = dataObject["userId"]
  const message = dataObject["message"]
  console.log("Recieved message from user with ID: ", userID, "!")
```



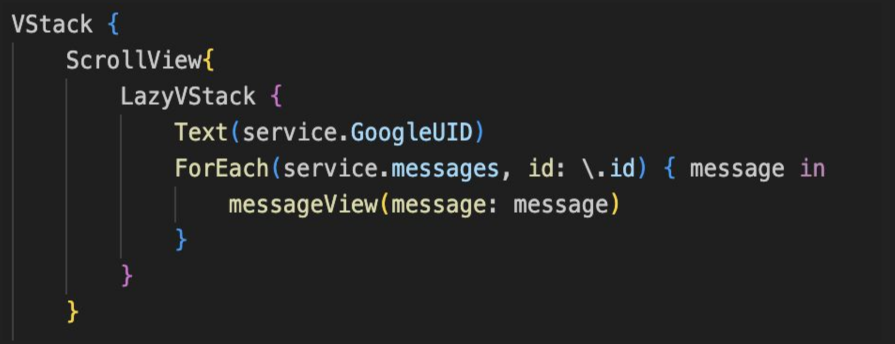
```
{
    "userId": "IMXpIdgL7ERhKPnDDnsMAmPMIh23",
    "message": "/useGPT Tell me a fun fact about signal processing"
}
```

Privatizing Users with userID

- After the ID is isolated, the variable gets sent to the backend
- ID is displayed



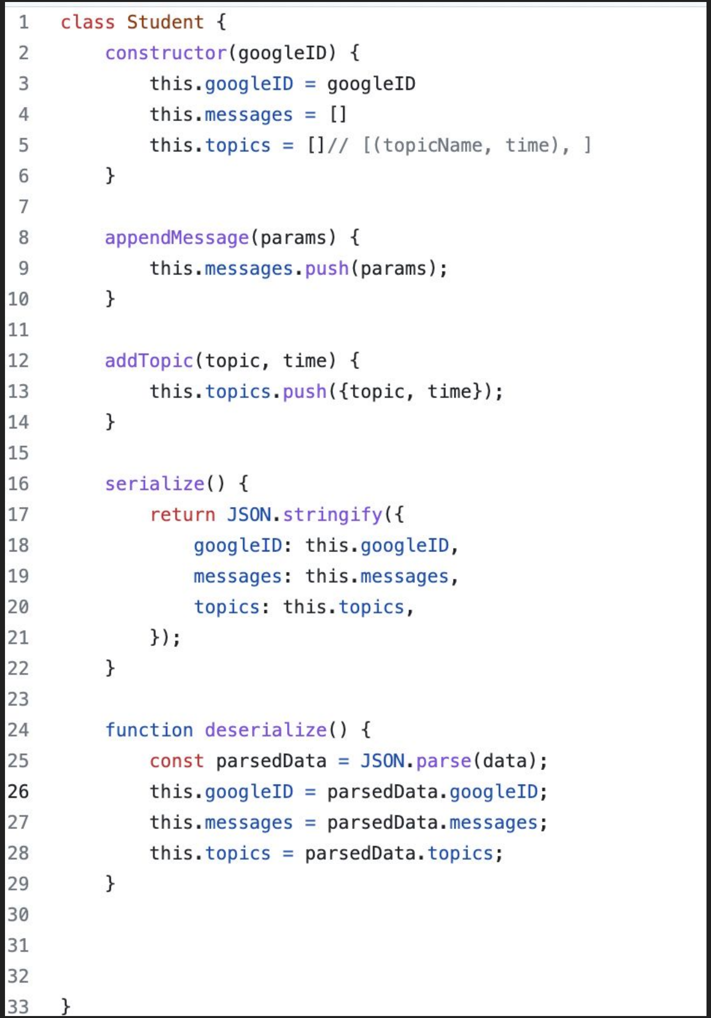
```
"""
{"userId": "%@","message": "%@"}
""", service.GoogleUID, currentMessage)
            print(finalMessage)
            service.socket.emit("RecieveUserMessage", finalMessage)
    }
    currentMessage = ""
}
```



```
VStack {
    ScrollView{
        LazyVStack {
            Text(service.GoogleUID)
            ForEach(service.messages, id: \.id) { message in
                messageView(message: message)
            }
        }
    }
```

Created Student Class

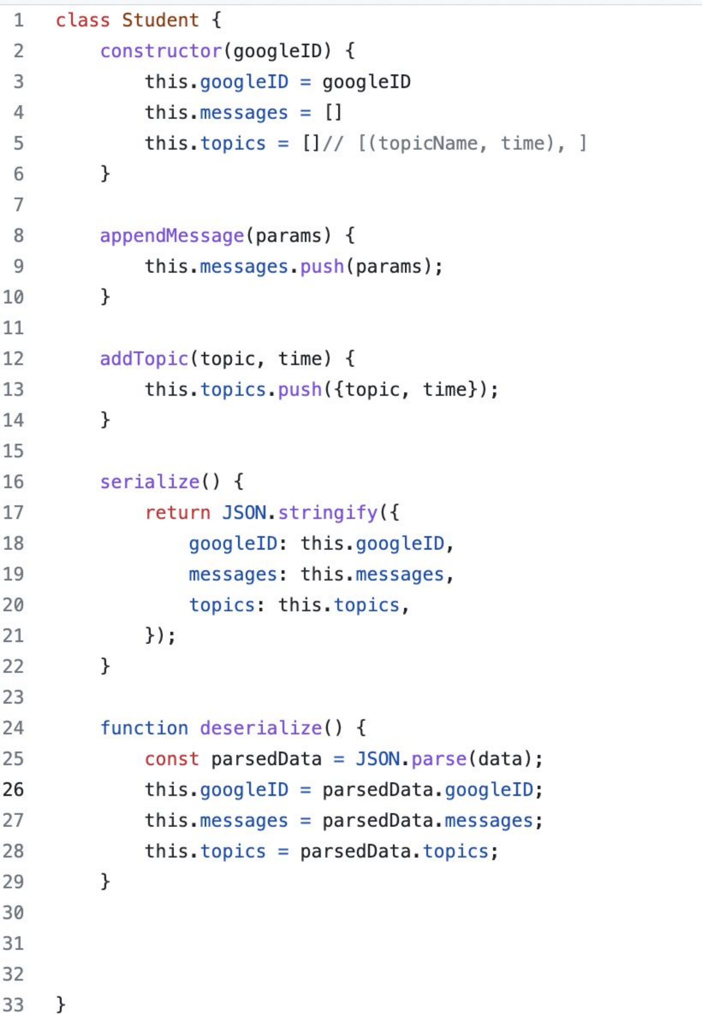
- Class Student:
    - Represents a student with a unique Google ID, a list of messages, and a list of topics.
    - Each message is an object containing sender information, message content, and a timestamp.
    - Each topic is represented as an object with a topic name and a timestamp.
- Method appendMessage(sender, content, time):
    - Adds a new message to the messages array for the student.



```
class Student {
    constructor(googleID) {
        this.googleID = googleID
        this.messages = []
        this.topics = []// [(topicName, time), ]
    }

    appendMessage(params) {
        this.messages.push(params);
    }

    addTopic(topic, time) {
        this.topics.push({topic, time});
    }

    serialize() {
        return JSON.stringify({
            googleID: this.googleID,
            messages: this.messages,
            topics: this.topics,
        });
    }

    function deserialize() {
        const parsedData = JSON.parse(data);
        this.googleID = parsedData.googleID;
        this.messages = parsedData.messages;
        this.topics = parsedData.topics;
    }



}
```

Created Student Class

- Method addTopic(topicName, time):
    - Adds a new topic to the topics array for the student.
- Method serialize():
    - Converts the Student object into a JSON string.
    - Returns a JSON representation of the student, including Google ID, messages, and topics.
- Method deserialize(data):
    - Takes a JSON string as a parameter and updates the Student object with the information from the string.
    - Parses the JSON string to extract Google ID, messages, and topics, then updates the corresponding properties of the student object.



```
class Student {
    constructor(googleID) {
        this.googleID = googleID
        this.messages = []
        this.topics = []// [(topicName, time), ]
    }

    appendMessage(params) {
        this.messages.push(params);
    }

    addTopic(topic, time) {
        this.topics.push({topic, time});
    }

    serialize() {
        return JSON.stringify({
            googleID: this.googleID,
            messages: this.messages,
            topics: this.topics,
        });
    }

    function deserialize() {
        const parsedData = JSON.parse(data);
        this.googleID = parsedData.googleID;
        this.messages = parsedData.messages;
        this.topics = parsedData.topics;
    }



}
```

Future improvements

- Allow users to use both a private and public chat
- Allow students to create public and private chats
- Analyze topics and messages by student