



DATA ENGINEERING MAJOR

ADVANCED LEARNING

PROJECT REPORT

Automated Quiz Generation App: Transforming Audio Prompts into Interactive Google Form Quizzes

Author :

MARZOUG AYOUB

Advisor :

Pr. MAHMOUDI ABDELHAK

Academic Year 2022/2023

Contents

Introduction	4
1 Preamble	5
1.1 ChatGPT : Future of Conversational AI	5
1.2 OpenAI API : Unlocking the Power of AI	6
1.3 Google Forms API: Streamlining Data Collection & Automation	6
1.4 Whisper : Advancing Speech Recognition and Synthesis	7
1.5 Streamlit : Web for Data & AI Scientist	8
2 App Architecture	9
3 Application Code	10
Conclusion	16

List of Figures

1.1	OpenAI's ChatGPT	5
1.2	OpenAI's Whisper	7
3.1	Importing Libraries	10
3.2	API Key & Endpoint	11
3.3	Recording Function	11
3.4	Transcribing Function	12
3.5	Main Function	13
3.6	Main Function	14

Acronyms & Abbreviations

GPT Generative Pre-Trained Transformer

DL Deep Learning

Speech-2-Text Speech To Text

RE Regular Expressions

API Application Programming Interface

AWS Amazon Web Services

ASR Automatic Speech Recognition

EDA Exploratory Data Analysis

NLP Natural Language Processing

CSV Comma-Separated Values

ANN Artificial Neural Network

Introduction

In today's fast-paced digital era, technology has revolutionized the way we access and consume information. With the rise of voice-enabled devices and virtual assistants, audio prompts have become increasingly prevalent as a convenient means of interacting with technology. However, harnessing the potential of audio prompts to generate interactive educational content remains largely untapped. This project report introduces an innovative solution that bridges this gap by developing an app that seamlessly converts ChatGPT responses obtained from audio prompts into engaging Google Form quizzes.

This report gives an account of everything that has been achieved in the course of the project carried out as part of the "Advanced Learning" course given during our data engineering training course. The aim was to build an app that creates a Google form Quiz from a ChatGPT response obtained from an Audio prompt input.

The primary goal of this project is to leverage the power of natural language processing and machine learning to streamline the process of quiz creation, enabling educators, content creators, and learners to transform spoken questions into dynamic and interactive quizzes. By integrating cutting-edge technologies, this app aims to empower users to effortlessly generate engaging quizzes from audio prompts, thereby enhancing the accessibility and interactivity of educational content. The app's core functionality revolves around utilizing ChatGPT, a state-of-the-art language model, to process audio input, extract meaningful responses, and automatically populate a Google Form with quiz questions based on the obtained information.

The link to the Demo Video is as follows : [Demo Video](#).

Preamble

This section provides a general introduction to conversational AI techniques and the various tools used in this project.

1.1 ChatGPT : Future of Conversational AI

ChatGPT is an advanced language model developed by OpenAI. It belongs to the GPT (Generative Pre-trained Transformer) family of models and is designed to understand and generate human-like text based on the provided input. It has been trained on a vast amount of diverse and high-quality text data from the internet, enabling it to capture patterns, context, and nuances of human language.

The power of ChatGPT lies in its ability to generate coherent and contextually relevant responses in real-time. It utilizes a transformer architecture, which allows it to efficiently process and understand the relationships between words and sentences. The model breaks down the input text into smaller units called tokens and assigns each token a contextual representation based on its surrounding words. By analyzing the context of the input and leveraging the learned knowledge from the training data, ChatGPT generates meaningful and contextually appropriate responses.

The versatility of ChatGPT is another aspect that contributes to its power. It can be fine-tuned for various tasks and domains, including text completion, language translation, question-answering, and more. By fine-tuning the model on specific datasets, it can adapt to different contexts and provide more accurate and targeted responses. This flexibility allows ChatGPT to be used in a wide range of applications, from virtual assistants and chatbots to content generation and language understanding tasks.

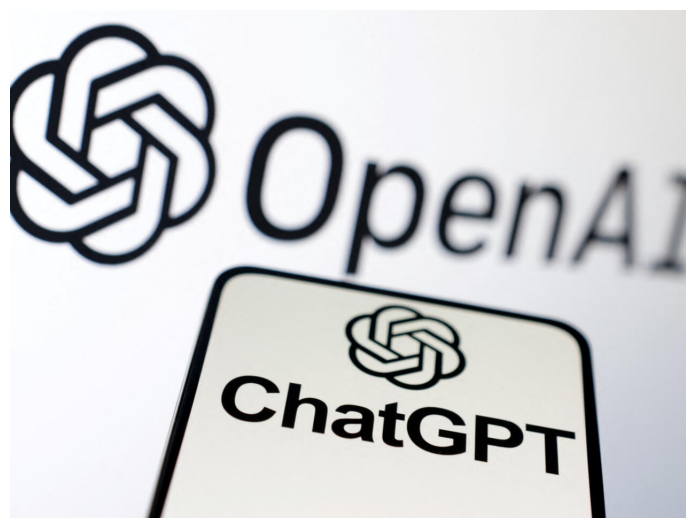


Figure 1.1: OpenAI's ChatGPT

The applications of ChatGPT are extensive and span across multiple domains. In the field of customer

service, ChatGPT can be employed to provide personalized and conversational interactions with users, addressing their queries and concerns in a human-like manner. It also finds utility in content creation, where it can assist writers by generating creative ideas, suggesting improvements, or even generating entire pieces of text. Educational platforms can leverage ChatGPT to enhance the learning experience by offering interactive tutorials, personalized feedback, and adaptive assessments. Furthermore, ChatGPT can contribute to research and development by assisting scientists in analyzing large amounts of textual data, aiding in language translation, and facilitating advancements in natural language processing. Its wide range of applications demonstrates the vast potential of ChatGPT in transforming various industries and improving human-computer interactions.

1.2 OpenAI API : Unlocking the Power of AI

OpenAI API provides developers with a comprehensive suite of tools and services that harness the power of artificial intelligence (AI). This API allows developers to integrate state-of-the-art language models, such as ChatGPT, into their applications, enabling them to create intelligent and interactive experiences. With its user-friendly interface and extensive capabilities, OpenAI API empowers developers to leverage cutting-edge AI technologies without the need for extensive machine learning expertise.

One of the key tools offered by OpenAI API is the text generation capability. Developers can use this feature to generate natural and coherent text based on prompts provided by users. Whether it's generating conversational responses, writing code snippets, or composing blog posts, OpenAI API excels in producing high-quality text that aligns with the desired context and style. The API's ability to understand and generate human-like language is a testament to the immense research and development invested in training the language models.

Another noteworthy aspect of OpenAI API is its ease of integration. The API is designed to be simple and straightforward, allowing developers to quickly integrate AI capabilities into their applications. With well-documented endpoints and comprehensive developer resources, OpenAI API streamlines the integration process, saving valuable time and effort. This accessibility empowers developers to focus on the creative aspects of their projects while leveraging the AI capabilities provided by OpenAI.

The advantages of using OpenAI API extend beyond its ease of integration. By utilizing the power of AI, developers can enhance the user experience of their applications, making them more engaging, personalized, and responsive. OpenAI API opens up new avenues for innovation, enabling developers to create virtual assistants, intelligent chatbots, content generation tools, and much more. Moreover, with ongoing advancements and updates from OpenAI, developers can benefit from the continuous improvements to the underlying models, ensuring that their applications stay at the forefront of AI technology. OpenAI API truly empowers developers to harness the potential of AI and revolutionize the way we interact with technology.

1.3 Google Forms API: Streamlining Data Collection & Automation

The Google Forms API is a powerful tool that enables developers to programmatically interact with and automate tasks related to Google Forms. With this API, developers can create, modify, and retrieve form responses, as well as manage form settings and structure. The Google Forms API offers a range of functionalities that streamline data collection, analysis, and integration, making it an essential resource for developers seeking to leverage the capabilities of Google Forms in their applications.

One of the key features of the Google Forms API is its ability to create and customize forms programmatically. Developers can dynamically generate forms, define question types, and set validation rules, allowing for tailored and interactive form experiences. By using the API, developers can seamlessly integrate form creation into their applications, automating the process and reducing the need for manual form setup. This capability is particularly beneficial for applications that require user input or feedback, surveys, or data collection.

Additionally, the Google Forms API offers robust functionality for managing and retrieving form responses. Developers can programmatically fetch form responses, apply filters and sorting criteria, and perform data analysis on the collected information. This feature enables the automation of data processing and analysis, making it easier to extract meaningful insights from large volumes of form data. By leveraging the API, developers can build applications that streamline data workflows, facilitate reporting, and enable real-time data analysis.

The advantages of using the Google Forms API extend beyond data collection and analysis. Integration with other Google services, such as Google Sheets and Google Drive, allows developers to seamlessly store and manipulate form responses, making it easier to manage and share collected data. Furthermore, the API provides robust security features, ensuring that data is transmitted and stored securely. The Google Forms API empowers developers to create powerful applications that harness the capabilities of Google Forms, facilitating efficient data collection, analysis, and integration within their software ecosystems.

1.4 Whisper : Advancing Speech Recognition and Synthesis

The Whisper model, developed by OpenAI, is an innovative deep learning model designed to tackle speech recognition and synthesis tasks. Whisper utilizes a combination of techniques, including unsupervised learning and transfer learning, to achieve impressive performance in converting spoken language into written text and vice versa. With its remarkable accuracy and adaptability, the Whisper model holds great promise for various applications in speech technology.

One of the key capabilities of the Whisper model is automatic speech recognition (ASR). ASR involves converting spoken words into written text and plays a crucial role in applications like transcription services, voice assistants, and voice-controlled systems. Whisper's ASR capability demonstrates exceptional accuracy, allowing it to accurately transcribe spoken content, even in challenging acoustic environments. The model's ability to handle different accents, languages, and speech variations makes it a powerful tool for improving the accessibility and usability of speech-driven applications.



Figure 1.2: OpenAI's Whisper

In addition to ASR, the Whisper model excels in speech synthesis, which involves converting written text into natural-sounding speech. Whisper's speech synthesis capability enables the generation of high-quality and human-like speech, which finds applications in voice-over services, assistive technologies, and interactive voice response systems. By leveraging Whisper's ability to synthesize speech, developers can create engaging and lifelike conversational experiences, enhancing the user interaction with voice-based applications.

The applications of the Whisper model span various domains. In healthcare, Whisper can be utilized to transcribe patient-doctor interactions, aiding in medical documentation and facilitating more efficient and accurate

healthcare services. In the education sector, the model can provide real-time transcription and closed captioning, enhancing accessibility for students with hearing impairments. Furthermore, in the entertainment industry, Whisper's speech synthesis capability can be employed for voice acting, dubbing, and creating realistic virtual characters.

Overall, the Whisper model represents a significant advancement in speech recognition and synthesis, offering developers powerful tools to transform spoken language into written text and vice versa. With its exceptional accuracy, adaptability, and versatility, the Whisper model paves the way for improved accessibility, enhanced user experiences, and innovative applications in various fields that rely on speech technology.

1.5 Streamlit : Web for Data & AI Scientist

Streamlit is a powerful Python library specifically designed for creating interactive and customizable web applications with ease. It simplifies the process of building data-driven applications, including those that utilize AI models, by providing a straightforward and intuitive framework. With Streamlit, developers can effortlessly transform their AI projects into user-friendly web interfaces, allowing them to showcase their work and engage users in a seamless and interactive manner.

Streamlit excels in its ability to streamline the development process for AI developers. It offers a simple and intuitive syntax that allows developers to quickly prototype and iterate on their applications. The library provides a wide range of pre-built components, such as sliders, buttons, and plots, that can be easily integrated into the user interface. This extensive set of components empowers AI developers to present their models and results effectively, enhancing the user experience and understanding.

Another significant advantage of Streamlit is its ability to automatically update the app in real-time as developers make changes to the code. This live-reloading feature saves valuable development time by eliminating the need for manual app restarts. It enables developers to make changes and immediately visualize the effects, making the development process more interactive and efficient.

Additionally, Streamlit seamlessly integrates with popular data science and machine learning libraries, such as Pandas, NumPy, and TensorFlow. This integration simplifies the process of incorporating AI models and data processing functionalities into the app. AI developers can leverage Streamlit's capabilities to build data visualization dashboards, interactive model demos, and even deploy AI-powered web applications, all without the need for extensive web development expertise.

In summary, Streamlit provides AI developers with a powerful tool for rapidly prototyping, visualizing, and deploying AI applications. Its simplicity, real-time updates, and seamless integration with popular libraries make it an invaluable asset for building intuitive and interactive web interfaces. Streamlit empowers AI developers to showcase their work, share insights, and engage users in a more immersive and user-friendly way.

App Architecture

The audio-to-quiz generation app is designed to seamlessly convert audio recordings into Google Forms quizzes by integrating the Whisper model for transcription and the ChatGPT model for quiz generation. The architecture of this app can be divided into several key components, including the streamlit web interface, the audio transcription module, and the Google Forms quiz generation module.

The streamlit web interface serves as the user-facing component of the app. It provides a user-friendly and intuitive interface where users can upload their audio recordings and initiate the conversion process. The web interface allows users to interact with the app and receive the generated Google Forms quiz URL once the conversion is complete. Streamlit, a popular Python library for building interactive web applications, can be utilized to develop this interface.

The audio transcription module leverages the Whisper model for accurate and efficient transcription of the uploaded audio recording. This module processes the audio input using the Whisper model, which converts the spoken content into written text. Whisper's powerful automatic speech recognition (ASR) capabilities enable it to handle various accents, languages, and environmental conditions, ensuring accurate transcription results. The module then outputs the transcribed text, which serves as the basis for generating the quiz questions.

The Google Forms quiz generation module utilizes the transcribed text and integrates the ChatGPT model to automatically create a quiz. This module takes the transcribed text as input and utilizes the powerful natural language processing capabilities of ChatGPT to generate a set of engaging and relevant quiz questions. The module can extract key information from the transcribed text and formulate questions based on that information. It also generates answer options for each question, providing a comprehensive and interactive quiz experience. Finally, the module generates a Google Forms quiz and provides the URL to the user via the web interface.

The app architecture also includes appropriate data flow and communication channels between these components. The streamlit web interface communicates with the audio transcription module to receive the transcribed text. This transcribed text is then passed to the Google Forms quiz generation module, which uses it as a basis for creating the quiz questions. The generated quiz, along with the quiz URL, is sent back to the web interface for display and access by the user. This seamless integration ensures a smooth and efficient user experience throughout the audio-to-quiz conversion process.

Overall, the architecture of the audio-to-quiz generation app combines the power of the Whisper model for accurate transcription and the ChatGPT model for dynamic quiz generation. By utilizing a Streamlit web interface and integrating these components effectively, the app enables users to easily convert audio recordings into interactive Google Forms quizzes, providing a novel and convenient approach to content creation and educational engagement.

Application Code

First we start by importing the various libraries and modules necessary for the functionality of the app.

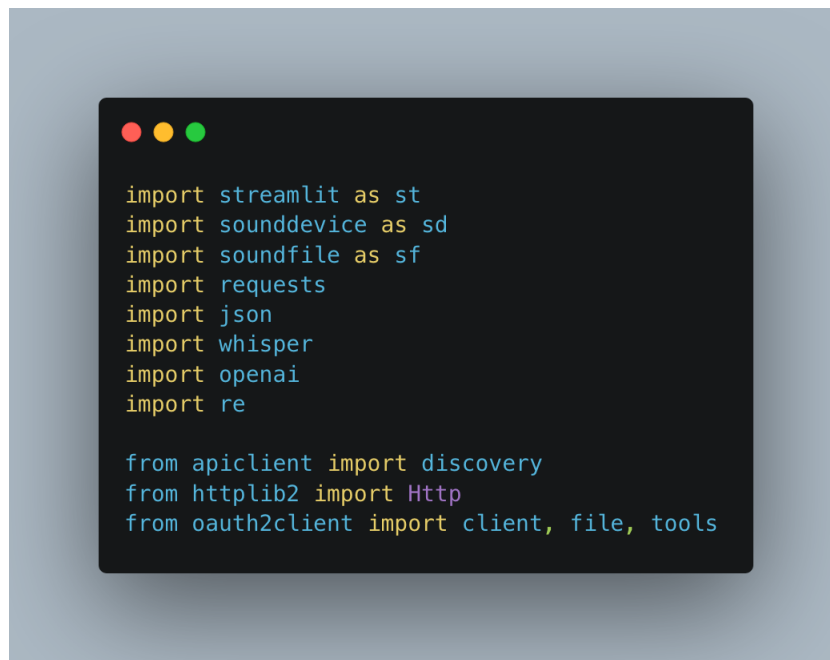



Figure 3.1: Importing Libraries

- **streamlit** (imported as **st**): Streamlit is a Python library used for building interactive web applications. It provides an intuitive and straightforward framework for creating user interfaces.
- **sounddevice** (imported as **sd**): Sounddevice is a Python library for recording and playing audio. It allows for capturing audio from the microphone or playing audio files.
- **soundfile** (imported as **sf**): Soundfile is a Python library for reading and writing sound files. It provides a convenient interface for handling audio files in various formats.
- **requests**: The **requests** library enables making HTTP requests to external APIs. It is used for sending and receiving data over the web.
- **json**: The **json** module provides functionality for working with JSON (JavaScript Object Notation) data, including encoding and decoding JSON objects.
- **whisper**: The **whisper** module likely refers to a custom module or package specific to the application. Its purpose may be related to audio processing or speech recognition.
- **openai**: The **openai** module is the Python library provided by OpenAI. It allows developers to interact with OpenAI's models and services, such as ChatGPT.
- **re**: The **re** module provides regular expression matching operations. It is commonly used for pattern matching and text processing tasks.

These import statements ensure that the necessary libraries and modules are available for the app to perform audio recording, file handling, API communication, natural language processing, and other relevant tasks.

Next we specify the API key and the endpoint URL for the Whisper model's completions API. This allows us to make API calls to interact with the Whisper model and utilize its capabilities for speech recognition or other related tasks.

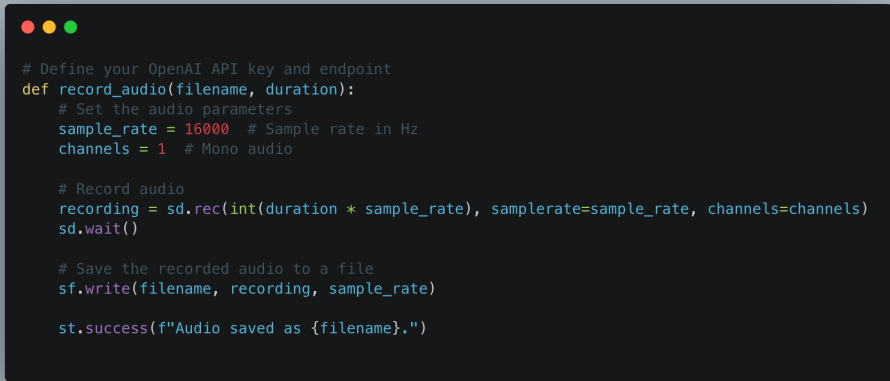


```
# Define your OpenAI API key and endpoint
API_KEY = "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"
API_ENDPOINT = "https://api.openai.com/v1/engines/whisper/betas/0.3.0/completions"
```

Figure 3.2: API Key & Endpoint

- **API_KEY** : This variable stores the OpenAI API key. The API key is a unique identifier that grants access to the OpenAI API services. It is used to authenticate and authorize your requests to the OpenAI API.
- **API_ENDPOINT** : This variable stores the endpoint URL for the specific API service we want to use. The endpoint represents the specific location or URL where we send your API requests to interact with the OpenAI Whisper model. It indicates the desired operation or functionality provided by the API, such as generating completions in this case.

Next, the following code snippet defines a function that records audio from the microphone and saves it to a file.



```
# Define your OpenAI API key and endpoint
def record_audio(filename, duration):
    # Set the audio parameters
    sample_rate = 16000 # Sample rate in Hz
    channels = 1 # Mono audio

    # Record audio
    recording = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=channels)
    sd.wait()

    # Save the recorded audio to a file
    sf.write(filename, recording, sample_rate)

    st.success(f"Audio saved as {filename}.")
```

Figure 3.3: Recording Function

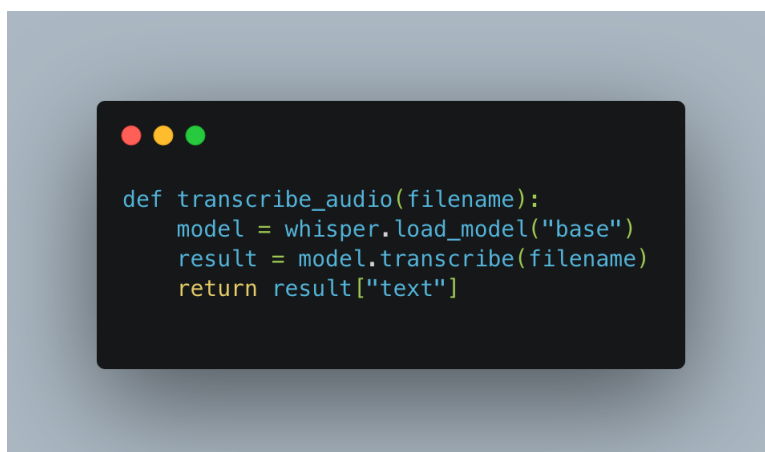
- **filename** : This parameter represents the desired filename or path where the recorded audio will be saved.
- **duration** : This parameter specifies the duration of the audio recording in seconds.

Inside the function:

1. The sample rate and number of channels for the audio recording are set. In this case, the sample rate is 16000 Hz, and the audio is recorded in mono (one channel).
2. The `sd.rec` function from the sounddevice library is used to record the audio. The number of frames to be recorded is calculated based on the desired duration and sample rate. The recorded audio is stored in the recording variable.
3. The `sd.wait()` function is called to ensure that the recording is complete before proceeding.
4. The recorded audio is saved to a file using the `sf.write` function from the soundfile library. The filename, recorded audio data (recording), and sample rate are provided as parameters.
5. Finally, a success message is displayed using `st.success` (likely from the Streamlit library) to indicate that the audio has been successfully saved with the specified filename.

By calling this function with appropriate arguments, we can record audio for the specified duration and save it to the desired file location.

After that, we defined a function that providing the filename of an audio file, generates the transcribed text from that audio file using the Whisper model. The Whisper model is specifically designed for automatic speech recognition (ASR) tasks and can convert spoken content into written text.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code inside the editor is as follows:

```
def transcribe_audio(filename):  
    model = whisper.load_model("base")  
    result = model.transcribe(filename)  
    return result["text"]
```

Figure 3.4: Transcribing Function

Inside the function:

1. The Whisper model is loaded using the `whisper.load_model` function. The argument `base` specifies the pre-trained base model for speech recognition. This step initializes the model and prepares it for transcription.
2. The `transcribe` method of the loaded model is called with the `filename` parameter. This method transcribes the audio content in the specified file.
3. The transcription result is stored in the `result` variable, which likely contains additional metadata along with the transcribed text.
4. The transcribed text is extracted from the `result` dictionary using the key `text`.

Finally, the transcribed text is returned by the function.

```

def main():
    st.title("Audio ChatGPT To Quiz Form")

    duration = st.slider("Recording Duration (seconds)", min_value=1, max_value=10, value=5)
    n = st.slider("Number of Quiz Questions ", min_value=1, max_value=20, value=10)
    filename = st.text_input("Enter the filename to save the audio", "audio.wav")

    if st.button("Record"):
        record_audio(filename, duration)
        st.info("Recording complete.")

    if st.button("Transcribe"):
        st.info("Transcribing audio...")
        global transcription
        transcription = transcribe_audio(filename)
        st.success("Transcription:")
        st.write(transcription)
        st.info("Generating form...")
        Link = form_generator(transcription,n)
        st.success("Link to the form:")
        st.write(Link)

```

Figure 3.5: Main Function

The provided code defines a function `main()` that serves as the entry point for the Streamlit application. Let's break down the functionality of this function:

```

def form_generator(text,n):
    openai.api_key = API_KEY
    content = text + f""" The quiz should contain {n} questions with multiple choice answers in the
    same format as the following, and in the end write finished :
        Title of quiz : "quiz on ..."
        Question 1 : ....
            a.
            b.
            c.
            d.
        Question 2 : ....
            a.
            b.
            c.
            d.
    """

    completion = openai.ChatCompletion.create(model = "gpt-3.5-turbo", messages = [{"role": "user",
    "content": content }])
    requests = completion.choices[0].message.content

    # Parsing the result

    Title = re.findall(r'Quiz on (.*)\n', requests)
    questions = re.findall(r'Question \d+: (.*)\n', requests)

    a_answers = re.findall(r'a\.(.*)\n', requests)
    b_answers = re.findall(r'b\.(.*)\n', requests)
    c_answers = re.findall(r'c\.(.*)\n', requests)
    d_answers = re.findall(r'd\.(.*)\n', requests)

    q = []

    for i in range(n):
        q.append({
            "createItem": {
                "item": {
                    "title": questions[i],
                    "questionItem": {
                        "question": {
                            "required": True,
                            "choiceQuestion": {
                                "type": "RADIO",
                                "options": [
                                    {"value": a_answers[i]},
                                    {"value": b_answers[i]},
                                    {"value": c_answers[i]},
                                    {"value": d_answers[i]}
                                ],
                                "shuffle": True
                            }
                        }
                    }
                },
                "location": {
                    "index": i
                }
            }
        })

    SCOPES = "https://www.googleapis.com/auth/forms.body"
    DISCOVERY_DOC = "https://forms.googleapis.com/$discovery/rest?version=v1"

    store = file.Storage('token.json')
    creds = None
    if not creds or creds.invalid:
        flow = client.flow_from_clientsecrets('key.json', SCOPES)
        creds = tools.run_flow(flow, store)

    form_service = discovery.build('forms', 'v1', http=creds.authorize(
        Http()), discoveryServiceUrl=DISCOVERY_DOC, static_discovery=False)

    # Request body for creating a form
    NEW_FORM = {
        "info": {
            "title": Title[0],
        }
    }

    # Request body to add a multiple-choice question
    NEW_QUESTION = {
        "requests": q
    }

    # Creates the initial form
    result = form_service.forms().create(body=NEW_FORM).execute()

    # Adds the question to the form
    question_setting = form_service.forms().batchUpdate(formId=result["formId"],
    body=NEW_QUESTION).execute()

    # Prints the result to show the question has been added
    get_result = form_service.forms().get(formId=result["formId"]).execute()
    return get_result["responderUri"]

```

The provided code defines a function `form_generator` that generates a Google Form quiz based on a given text and the number of questions (`n`). Let's break down the functionality of this function:

- **text:** The parameter represents the transcribed text used to generate the quiz form.
- **n:** The parameter represents the number of quiz questions to include in the form.

Inside the function:

1. The OpenAI API key is set to `API_KEY` to authenticate and authorize API requests.
2. The content of the quiz form is constructed by concatenating the given text with additional formatting. The content includes instructions for the quiz format, the desired title of the quiz, and placeholders for the questions and answer choices.
3. The constructed content is used in an API call to OpenAI's GPT-3.5 Turbo model for chat-based completions. The `openai.ChatCompletion.create` method is called with the model specified as `gpt-3.5-turbo` and the content provided as a user message.
4. The response from the API call is stored in the `completion` variable, which contains the generated quiz form as a chat response.
5. The generated form response is parsed using regular expressions (`re`) to extract the title, questions, and answer choices from the response text.
6. An empty list `q` is created to store the formatted quiz questions and answer choices.
7. A loop is executed `n` times to iterate through each question and answer choice. Each question is formatted as an item in the required structure for the Google Forms API.
8. The necessary authentication and authorization steps are performed using the Google Forms API client. This involves handling authentication credentials, creating an instance of the forms service, and authorizing the HTTP client.
9. The request body for creating the form and adding the multiple-choice questions are defined.
8. The initial form is created using `form_service.forms().create` and the response is stored in the `result` variable.
9. The generated questions are added to the form using `form_service.forms().batchUpdate` with the form ID and the request body.
10. The final form URL (responder URI) is retrieved using `form_service.forms().get` and returned as the result.

By calling this function with the transcribed text and the desired number of questions, you can generate a quiz form that follows the specified format and contains the questions and answer choices extracted from the text. The form is created using the Google Forms API, and the resulting form URL is returned for further use or display.

General Conclusion

In conclusion, the project report has explored the development of an innovative app that leverages the power of AI to seamlessly convert audio recordings into interactive Google Forms quizzes. By integrating the Whisper model for accurate transcription and the ChatGPT model for dynamic quiz generation, the app offers a comprehensive solution for transforming spoken content into engaging educational experiences.

The app's architecture, comprising the streamlit web interface, audio transcription module, and Google Forms quiz generation module, enables users to easily upload audio recordings, obtain accurate transcriptions, and generate customized quizzes with just a few simple steps. The user-friendly interface and seamless integration of advanced AI models simplify the process, making it accessible to both developers and end users.

The Whisper model, with its exceptional automatic speech recognition capabilities, provides accurate transcriptions even in challenging acoustic environments. This ensures the reliability and quality of the transcribed text, which serves as the foundation for generating the quiz questions. The ChatGPT model further enhances the app's functionality by dynamically formulating relevant quiz questions based on the transcribed text, creating an interactive and personalized learning experience.

By utilizing Streamlit, the app simplifies the development process for AI developers, enabling them to create interactive web interfaces without extensive web development expertise. The library's simplicity, real-time updates, and integration with popular data science libraries make it a powerful tool for showcasing AI models and engaging users in a visually appealing and intuitive manner.

Overall, the audio-to-quiz generation app represents a significant advancement in educational technology, combining the power of speech recognition, natural language processing, and web interface development. By bridging the gap between audio prompts, transcription, and quiz generation, the app empowers educators, content creators, and learners to easily convert spoken content into interactive quizzes, enhancing accessibility, engagement, and interactivity in the learning process.

Bibliography

- [1] OpenAI API Documentation : "<https://platform.openai.com/docs/>". 2023.
- [2] Google Forms API Documentation : "<https://developers.google.com/forms/api/reference/rest?hl=fr>". 2023.
- [3] OpenAI ChatGPT : "<https://chat.openai.com/>". 2023.
- [4] OpenAI Whisper : "<https://openai.com/research/whisper>". 2023.
- [5] Streamlit Documentation : "<https://docs.streamlit.io/>". 2023.