Overview of the Application

This application is a tool designed to convert audio files (MP3 or WAV formats) into text. Once the audio is transcribed, users can download the generated transcript in either TXT or PDF format. Built using **Streamlit**, a Python framework for creating web applications, the app offers an intuitive and interactive interface. Users can easily upload audio files, convert them to text, edit the transcript if needed, and finally export the result in a user-friendly format.

The application utilizes several powerful libraries:

- **SpeechRecognition** for accurate speech-to-text conversion,
- **PyDub** for audio file processing,
- **FPDF** for generating downloadable PDF documents.

The clean, simple design ensures that users can quickly convert speech into text with minimal effort. Whether for meetings, lectures, or podcasts, this tool facilitates fast and reliable transcription. To explore how the app works or to view the source code, please visit the GitHub repository linked below.

GitHub Repo Link: Audio-to-Text

Explanation of Each Part of the Code:

1. Importing Required Libraries

```
import streamlit as st
import speech_recognition as sr
from pydub import AudioSegment
import tempfile
import os
from fpdf import FPDF
```

Libraries:

- `streamlit as st`: **Streamlit** is used to create the interactive web application interface. It allows for easy integration of components such as file uploaders, text areas, buttons, and more, creating a seamless user experience for uploading and processing audio files.
- `speech_recognition as sr`: This library is the core component for converting audio (speech) to text. It uses various speech recognition APIs, including **Google's Speech Recognition API**, to handle the transcription.

- **`pydub.AudioSegment`**: **PyDub** is an audio processing library. It is used to manipulate and convert audio files into different formats. In this case, it helps convert MP3 or other audio formats into the preferred **WAV** format, which works best with **SpeechRecognition**.
- **`tempfile`**: This module is used to create temporary files that are automatically deleted when they are no longer needed. It's helpful for managing files during the conversion and transcription process without leaving behind unnecessary files.
- **`os`**: The **OS** library is responsible for file and directory management, specifically for deleting temporary files once they are no longer required.
- **`fpdf.FPDF`**: **FPDF** is a Python library that allows for the generation of PDF documents. In this application, it's used to create a downloadable PDF version of the transcribed text.

---

2. Helper Functions

2.1. Converting Audio Files to WAV Format

```
def convert_to_wav(uploaded_file):
    """Convert all audio files to WAV format"""
    file_type = uploaded_file.name.split('.')[-1].lower()
    with tempfile.NamedTemporaryFile(delete=False, suffix=f'.
{file_type}') as tmp_file:
        tmp_file.write(uploaded_file.getvalue())
        tmp_path = tmp_file.name

    audio = AudioSegment.from_file(tmp_path)
    with tempfile.NamedTemporaryFile(suffix='.wav', delete=False) as
wav_file:
        audio.export(wav_file.name, format="wav")
        return wav_file.name
```

Purpose:
This function converts any uploaded audio file (MP3, WAV, etc.) into a **WAV** format, which is the preferred format for transcription in **SpeechRecognition**.

- **`tempfile.NamedTemporaryFile`**: This function is used to create a temporary file to store the uploaded audio file. It ensures that the original file is preserved and does not affect the processing flow.
- **`AudioSegment.from_file(tmp_path)`**: **PyDub** loads the uploaded audio file using the `from_file()` method, regardless of the format.
- **`audio.export(wav_file.name, format="wav")`**: The audio is then exported in **WAV** format to another temporary file. The path to this new file is returned for further processing.

2.2. **Transcribing Audio to Text**

```python
def transcribe_audio(file_path, language='en-US'):
    """Speech recognition process using Google API"""
    r = sr.Recognizer()
    with sr.AudioFile(file_path) as source:
        audio_data = r.record(source)
        return r.recognize_google(audio_data, language=language)
```

Purpose:
This function processes the **WAV** file using **Google's Speech Recognition API** and converts the audio into text.

- **`sr.Recognizer()`**: Initializes the speech recognizer.
- **`with sr.AudioFile(file_path) as source:`**: Opens the provided audio file for transcription.
- **`r.record(source)`**: Records the audio data.
- **`r.recognize_google(audio_data, language=language)`**: The audio is passed to **Google's Speech API**, which transcribes the speech into text. The `language` parameter defaults to **English (US)** (`'en-US'`), but this can be customized for different languages.

2.3. **Creating a PDF from Text**

```python
def create_pdf(text):
    """Create PDF document (FIXED)"""
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.multi_cell(0, 10, text)
    return bytes(pdf.output())  # Convert bytearray to bytes
```

Purpose:
This function generates a **PDF document** containing the transcribed text, which can then be downloaded by the user.

- **`FPDF()`**: Initializes the PDF document.
- **`pdf.add_page()`**: Adds a new page to the PDF document.
- **`pdf.set_font("Arial", size=12)`**: Sets the font to Arial, size 12, for better readability.
- **`pdf.multi_cell(0, 10, text)`**: This method automatically wraps the text within the PDF, ensuring that long sentences or paragraphs don't get cut off.
- **`return bytes(pdf.output())`**: The PDF is then converted into bytes, which can be returned and provided for download.

---

3. Streamlit Interface

3.1. Title and Information Panel

```
st.title("□□ Audio-Text Converter")

# Information Window
with st.expander("ℹ About This Application"):
    st.markdown("""
    **Audio-Text Converter** is a speech-to-text transcription tool
that allows you to:
    - □ Convert MP3/WAV audio files to text
    - □ Edit the generated transcript
    - □ Export results in TXT or PDF formats
    """)
```

Explanation:

- **`st.title`**: Displays the title of the application with emojis for a more engaging presentation.
- **`st.expander`**: This creates a collapsible section, which provides users with more information about the app when expanded. It describes its features such as converting audio files to text, editing the text, and exporting the results in TXT or PDF formats.

3.2. File Upload Section

```
uploaded_file = st.file_uploader("Upload MP3/WAV file", type=["wav",
"mp3"])
audio_path = None

if uploaded_file:
    audio_path = convert_to_wav(uploaded_file)
```

Explanation:

- **`st.file_uploader`**: This component allows users to upload an audio file in either MP3 or WAV format.
- Once the file is uploaded, it's passed to the `convert_to_wav` function to ensure it is in **WAV** format, as that format is most compatible with the transcription process.

3.3. Transcribing the Audio and Displaying Results

```
if audio_path:
    try:
        # Convert to Text
        text = transcribe_audio(audio_path)
```

```python
        # Display Results
        st.subheader("Transcript")
        edited_text = st.text_area("Edit Text", text, height=250)
```

Explanation:

- If the **WAV** file exists, it is passed to **Google's API** for transcription using the `transcribe_audio` function.
- **`st.subheader`**: A subheading labeled "Transcript" is displayed to introduce the transcribed text.
- **`st.text_area`**: Displays the transcribed text in an editable text box, allowing users to modify the text before downloading it.

3.4. **Download Buttons for TXT and PDF**

```python
col1, col2 = st.columns(2)

with col1:
    st.download_button(
        "□ Download TXT",
        edited_text,
        "transcript.txt",
        "text/plain"
    )

with col2:
    pdf_data = create_pdf(edited_text)
    st.download_button(
        "□ Download PDF",
        pdf_data,
        "transcript.pdf",
        "application/pdf"
    )
```

Explanation:

- The application provides two download options:
  - **TXT**: The user can download the transcript as a **.txt** file.
  - **PDF**: The user can download the transcript as a **.pdf** file, with the edited text passed to the `create_pdf` function for PDF creation.
- **`st.download_button`**: A button is created that allows the user to download the final transcript in their desired format.

3.5. **Error Handling and Cleanup**

```python
except Exception as e:
    st.error(f"An error occurred: {str(e)}")

finally:
    # Cleanup
    if os.path.exists(audio_path):
        os.unlink(audio_path)
```

Explanation:

- **Error Handling**: If any exception occurs during the process (e.g., during file conversion or transcription), an error message is shown to the user.
- **Cleanup**: The **temporary files** are deleted in the `finally` block to ensure no unnecessary files are left behind, freeing up system resources.

Summary of the Workflow:

1. **File Upload:** Users upload an audio file (MP3 or WAV).
2. **Conversion to WAV:** If the uploaded file is not in **WAV** format, it is converted.
3. **Transcription:** The audio file is processed by **Google's Speech Recognition API** to convert the speech into text.
4. **Editing:** Users can edit the transcribed text directly within the app.
5. **Download Options:** Users can download the final transcript as either a **TXT** or **PDF** file.
6. **Cleanup:** Temporary files are deleted after processing to free up resources.

Conclusion:

This application effectively combines several popular Python libraries (**Streamlit**, **SpeechRecognition**, **PyDub**, and **FPDF**) to create an interactive, user-friendly tool for converting audio into text. Users can upload audio, edit the transcription, and easily download the result in TXT or PDF formats. The smooth interface and robust functionality make this tool ideal for anyone needing to convert speech to text quickly and efficiently.