Overview of the Application

This application is designed to convert audio files (MP3 or WAV formats) into text and allows users to download the generated transcript in either TXT or PDF format. Built with **Streamlit**, this web application provides an intuitive and interactive interface that lets users upload audio files, transcribe them into text, and then edit and export the result in a convenient format. The user-friendly design ensures a smooth experience for speech-to-text conversion.

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 build the web application interface. It allows easy creation of interactive applications with components like file uploaders, text areas, and download buttons.
- **`speech_recognition as sr`**: This library is used for converting speech into text. It supports different speech recognition APIs, including Google's Speech Recognition API.
- **`pydub.AudioSegment`**: PyDub is used for audio processing. The `AudioSegment` class handles audio file manipulation, including conversion between different formats.
- **`tempfile`**: This module allows for creating temporary files, which are automatically deleted when they are no longer needed. It is used here to store the uploaded audio file and the converted `.wav` file.
- **`os`**: This library is used for file and directory management tasks, such as deleting temporary files.
- **`fpdf.FPDF`**: FPDF is a library for generating PDF documents programmatically. This is used to create a PDF document 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"""
    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 into `.wav` format, as the `speech_recognition` library works best with WAV files.
- **`tempfile.NamedTemporaryFile`**: Creates a temporary file to store the uploaded audio content.
- **`AudioSegment.from_file(tmp_path)`**: Uses `PyDub` to load the uploaded audio file.
- **`audio.export(wav_file.name, format="wav")`**: Converts the audio into the `.wav` format and saves it to another temporary file. The path to the new `.wav` file is returned.

2.2. Transcribing Audio to Text

```
def transcribe_audio(file_path, language='en-US'):
    """Speech recognition process"""
    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 converts the audio content into text using Google's speech recognition API.
- **`sr.Recognizer()`**: Initializes the recognizer to process the audio.
- **`with sr.AudioFile(file_path) as source`**: Opens the audio file.
- **`r.record(source)`**: Records the audio from the file and processes it.
- **`r.recognize_google(audio_data, language=language)`**: This is the core speech-to-text function, which converts the recorded audio into text using Google's API.

The `language` parameter is set to `'en-US'` by default but can be changed for other languages.

2.3. Creating a PDF from Text

```
def create_pdf(text):
    """Create PDF"""
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.multi_cell(0, 10, text)
    return pdf.output()
```

- **Purpose**: This function creates a PDF document from the transcribed text.
- **`FPDF()`**: Initializes a new FPDF object used to create the PDF.
- **`pdf.add_page()`**: Adds a new page to the PDF document.
- **`pdf.set_font("Arial", size=12)`**: Sets the font to Arial with a size of 12.
- **`pdf.multi_cell(0, 10, text)`**: Adds the transcribed text to the PDF. The `multi_cell` method allows automatic text wrapping.
- **`return pdf.output()`**: Generates and returns the content of the PDF.

---

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
    """)
```

- **`st.title`**: Sets the title of the web page as "Audio-Text Converter" with some emojis for visual appeal.
- **`st.expander`**: Creates a collapsible section with additional information about the application. When expanded, it explains the tool's functionality, such as converting audio to text, editing transcripts, and exporting them 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:
    file_type = uploaded_file.name.split('.')[-1].lower()
    audio_path = convert_to_wav(uploaded_file)
```

- **`st.file_uploader`**: This Streamlit component allows users to upload an audio file. It supports both MP3 and WAV file formats.
- The code then checks if a file is uploaded. If so, it extracts the file type (MP3 or WAV) and converts the file to `.wav` format using the `convert_to_wav` function.

3.3. Transcribing the Audio and Displaying Results

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

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

- If the `.wav` file is successfully created, it is passed to the `transcribe_audio` function for speech recognition.
- **`st.subheader`**: Displays a subheading "Transcript" to label this section.
- **`st.text_area`**: Displays the transcribed text in a text box, allowing the user to edit the text before saving or exporting.

3.4. Download Buttons for TXT and PDF

```
# 4. Download Buttons
col1, col2 = st.columns(2)

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

with col2:
    pdf = create_pdf(edited_text)
```

```
    st.download_button(
        "📄 Download PDF",
        bytes(pdf),
        "transcript.pdf",
        "application/pdf"
    )
```

- The app provides two download options for the user:
  - **Download TXT**: The user can download the edited transcript as a `.txt` file.
  - **Download PDF**: The user can download the transcript as a `.pdf` file. The edited text is passed to the `create_pdf` function to generate the PDF.
- **`st.download_button`**: This Streamlit component creates a button for downloading the text or PDF file.

3.5. Error Handling and Cleanup

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

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

- **Error Handling**: If any exception occurs during processing (such as an issue with speech recognition or file conversion), an error message is displayed to the user.
- **Cleanup**: The `finally` block ensures that the temporary `.wav` file is deleted after processing, freeing up resources.

Summary of the Workflow:

1. **File Upload**: The user uploads an audio file (MP3 or WAV).
2. **Conversion to WAV**: If the uploaded file is not in WAV format, it is converted to `.wav`.
3. **Transcription**: The audio is processed using Google's Speech Recognition API to convert the speech into text.
4. **Editing**: The transcribed text is displayed in an editable text area, allowing the user to make changes.
5. **Download Options**: The user can download the final transcript as a TXT or PDF file.
6. **Cleanup**: Temporary files are deleted after processing to free up system resources.

Conclusion:

This application combines several libraries (**Streamlit** for the UI, **PyDub** for audio manipulation, **SpeechRecognition** for transcription, and **FPDF** for PDF generation) to create an interactive tool for converting audio into text. The user can upload an audio file, edit the transcript, and download it in a convenient format, all within a smooth and easy-to-use interface.