KCET Assistant Project Documentation

Overview

KCET Assistant is an intelligent FAQ chatbot application for Kamaraj College of Engineering and Technology built using Streamlit.

It enables students or visitors to ask questions related to the college and receive relevant answers using NLP (TF-IDF + cosine similarity).

It also features text-to-speech, chat export, and email integration.

Project Structure

kcet_chatbot/

```
 kcet.csv              # Dataset containing Questions and Answers
 kcet_logo.png         # College logo image (optional)
 vectorized.pkl        # Pickled TF-IDF vectorizer and dataset cache
 chatbot_app.py        # Main Streamlit application file
```

Features

Chatbot Features

- Accepts user questions and responds using a similarity-based NLP engine.
- Handles fuzzy/partial matches using cosine similarity.
- Uses TF-IDF vectorization for text preprocessing.
- Displays personalized chat bubbles for User and Assistant.

Voice Output

- Uses gTTS (Google Text-to-Speech) to convert assistant replies into speech.

- Audio plays automatically after the bot responds.

UI Customization

- Light/Dark theme toggle.
- Custom bubble and text color options.
- Dynamic banners, logos, and styled headers.

Chat Export & Email

- Export chat log as:
  - PDF
  - DOC
  - TXT
- Send PDF directly via email (Gmail SMTP).
- Includes KCET logo and branding in exported PDFs.

Chat Management

- Clear chat button to reset conversation.
- Maintains state with st.session_state.

Dependencies

Install the following libraries:

```
pip install streamlit pandas scikit-learn gTTS fpdf
```

Setup and Configuration

1. Email Configuration:

   Set your Gmail address and App Password:

```
SENDER_EMAIL = "your-email@gmail.com"
SENDER_PASSWORD = "your-app-password"
```

2. Data Preparation:

   Ensure kcet.csv contains the following columns:

```
Question,Answer
"Where is KCET located?","It is in Virudhunagar, Tamil Nadu."
```

3. Run the App:

```
streamlit run chatbot_app.py
```

How it Works

Vectorization & Matching

```
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(df['Question'])
```

When a user asks a question:

```
vec = vectorizer.transform([user_input])
similarity = cosine_similarity(vec, vectors)
```

Returns the answer with the highest similarity score if above threshold (0.6).

Text-to-Speech

```
tts = gTTS(text=response, lang='en')
```

Audio is encoded in base64 and played using HTML audio tags.

Email with Attachment

```
with smtplib.SMTP_SSL("smtp.gmail.com", 465) as smtp:
    smtp.login(SENDER_EMAIL, SENDER_PASSWORD)
    smtp.send_message(msg)
```

Only PDF export is attached to email.

Security Considerations

- Use App Passwords for Gmail SMTP.
- Avoid storing plain-text passwords in code for production.
- Consider storing credentials in a .env file and load them securely using dotenv.

Future Improvements

- Use LLMs (e.g., GPT-4) for more robust Q&A.
- Add voice input using SpeechRecognition.
- Support for image or document queries.
- Add admin portal to update Q&A dynamically.

Author

Developed by Shakthivel as a smart assistant for Kamaraj College of Engineering and Technology.