ProtonAutoML Assignment - Chatbot Application

Overview

This project is a chatbot application developed as part of the ProtonDatalabs AI Developer Assignment. The application allows users to upload files and ask questions based on the content of the files. The chatbot retrieves relevant information from the uploaded files to answer the user's questions.

Project Structure

The project consists of two main directories:

1. **Backend**: Contains server-side Python code responsible for handling requests, processing files, and generating responses.
2. **Frontend**: Contains client-side TypeScript code for the user interface, allowing users to interact with the chatbot.

Backend Modifications

1. Implemented endpoint `/predict` in `main.py` to process file uploads and user questions.
2. Integrated file handling to extract text content from uploaded documents using the `textract` library.
3. Utilized the `spaCy` library for text processing and sentence extraction.
4. Integrated the Hugging Face `transformers` library for question-answering using the RAG (Retrieval-Augmented Generation) model.
5. Addressed warning messages related to tokenizer class consistency.

Results:

- Successfully uploaded files are saved to the server's `uploads` directory.
- File upload notifications are displayed to the user through the frontend interface.

Frontend Modifications

1. Enhanced user experience with a pop-up notification for successful file uploads using `react-toastify`.
2. Expanded file type support to include `.txt`, `.docx`, `.pdf`, and `.csv` files.
3. Added basic styling to the application for improved visual appeal.
4. Improved file input handling and validation for a smoother user experience.

Results:

- Upon successful file upload, users are notified with a pop-up message indicating the file has been uploaded.
- User-submitted questions are processed and displayed along with the chatbot's response.

Usage

1. Clone the repository and navigate to the respective directories (`backend` and `frontend`).
2. Install dependencies using `pip install -r requirements.txt` for the backend and `npm install` for the frontend.
3. Launch the backend server using `uvicorn main:app --reload` in the `backend` directory.
4. Start the frontend application with `npm start` in the `frontend` directory.
5. Access the application in your web browser at `http://localhost:3000`.

Notes

- It is recommended to use Python 3.10 or above for running the backend server.
- Ensure that all necessary dependencies are installed and compatible with your environment.
- Handle edge cases such as large file uploads and unsupported file types gracefully to enhance user experience.
- Follow best programming practices including SOLID principles, OOPs, and type hints for robust and maintainable code.

Acknowledgements

Special thanks to ProtonDatalabs for providing this assignment opportunity.