REPORT

Q&A CHATBOT

1. File Loading and Processing Functions:

- **Functionality:**
  - **load_docx(file_path):** Uses `docx2txt` to extract text from DOCX files.
  - **load_excel(file_path):** Loads data from Excel files into a Pandas DataFrame and converts it to a string format.
  - **load_pdf(file_path):** Utilizes `PyMuPDF` (fitz) to extract text from PDF files.
  - **load_jpeg(file_path):** Uses `pytesseract` to perform OCR on JPEG images and extract text.
- **Design Decision:**
  - Chose specific libraries (`docx2txt`, `Pandas`, `PyMuPDF`, `pytesseract`) based on their ability to handle various file formats reliably.
  - Designed functions to catch exceptions and provide informative error messages to aid debugging.

2. Parallel File Loading:

- **Functionality:**
  - **parallel_load(file_type, file_paths):** Uses `ThreadPoolExecutor` to load files of each type (DOCX, Excel, PDF, JPEG) concurrently.
  - **load_and_cache_files():** Combines all loaded text into a single large text string.
- **Design Decision:**
  - Implemented parallel loading to improve efficiency, especially useful when dealing with multiple large files.
  - Ensured thread safety and error handling within the concurrent execution.

3. Hugging Face Transformers Integration:

- **Functionality:**
  - **query_index(question, context):** Uses the `deepset/roberta-base-squad2` model to answer questions based on the combined text.
  - **generate_text(prompt):** Uses the `gpt2` model for text generation based on the answer.
- **Design Decision:**
  - Selected models (`deepset/roberta-base-squad2` for QA, `gpt2` for text generation) known for their performance in NLP tasks.
  - Configured pipelines for ease of use, focusing on retrieving accurate answers and generating extended text.

4. Main Chatbot Functionality:

- **Functionality:**
  - **chatbot():** Implements the main loop for user interaction, allowing questions to be asked and answered interactively until the user decides to exit.
- **Design Decision:**
  - Provided a user-friendly interface with prompts and messages in both English and Hindi, enhancing accessibility.
  - Implemented a graceful exit mechanism (`'exit'` command) for user convenience.

5. Enhancements and Additional Context:

- **Functionality:**
  - Appends specific additional context (`"Generative AI, or frameworks such as Groq and Mistral, or other relevant technologies."`) to the combined text before querying.
- **Design Decision:**
  - Included relevant context to broaden the scope of potential answers and generate more informed responses.

Conclusion:

The design of this system leverages parallelism for efficient file processing, integrates advanced NLP models for accurate information retrieval and text generation, and provides a user-friendly interface for interactive querying. Error handling and informative messaging ensure robustness and usability. This approach aims to facilitate comprehensive document querying and information synthesis in a user-friendly manner.

This report outlines the thoughtful design decisions and implementation choices made to achieve the functionality of the Document Query Chatbot using Python and Hugging Face Transformers.