The TD Bank Chatbot architecture consists of a backend that processes documents and a frontend that allows the user to query the processed documents. This summary focuses on the backend processing.

The new backend consists of the following pre-processing steps:

1. Converting PDF documents to text maintaining the page location of converted text
2. Classifying the converted text into a list of predetermined categories
3. "Chunking" (i.e. grouping related pages together) the text
4. Summarizing the chunked text
5. Uploading the chunked text to an annotated vector database index to support RAG queries

The frontend uses a resource augmented generation technique (vector database index in combination with an LLM) to answer user queries.

The first version of the backend produced by the previous team combined all five pre-processing steps in one method making it difficult to review and experiment with different alternatives for each step. The new backend implements each step separately to support increased flexibility. In addition, where appropriate, the backend allows different local or cloud services to be used, specifically: Ollama, Azure and OpenAI are supported.

The table below enumerates the new pre-processing modules:

Module	Purpose	Remarks
texter.py	Converts PDF documents to a JSON objects file consisting of one object per PDF page listing the page text and the page number. Extracts PDF file properties (creation date, etc.) to a separate file. Cleans converted text.	Page numbers needed to be retained to allow front-end references to be generated. Text cleansing is done via regular expressions (previous version used a slower mechanism using LLMs)
classifier.py	Classifies the JSON output of texter. The classifier is called by the pineconer.py module as the processed chunks are uploaded to the vector database index.	The first version of the classifier by the previous team used a slower version with LLMs for classifications.
chunker.py	Attempts to group related file pages (similar topics) into chunks	
summarizer.py	Summarizes each file chunk for better matching with user queries	New version supports 3 LLM environments: Ollama, Azure and OpenAI
pineconer.py	Uploads annotated and summarized chunks to a pinecone vector database index	New version supports 3 embedding environments: Ollama, Azure and OpenAI. The embedding is used to calculate a vector equivalent of chunked text.