**Payoff Letter Generator with Document Agent**

**Introduction**

The Payoff Letter Generator is a sophisticated application designed to streamline the process of generating payoff letters from Note Purchase and Guarantee Agreement (NPGA) documents. By leveraging an Intelligent Document Agent, the application automates the extraction of key details from the provided documents, making the creation of payoff letters efficient and less prone to human error.

**1. Overview**

**1.1 Purpose of the Application**

The Payoff Letter Generator with Document Agent is developed to address the challenges involved in manually creating payoff letters. Payoff letters are crucial documents that confirm the repayment of a loan or debt. These letters must be accurate and reflect the correct details as per the Note Purchase and Guarantee Agreement (NPGA) documents.

The application's purpose is to:

- Extract relevant information automatically from NPGA documents.

- Reduce the time and effort required to generate payoff letters.

- Minimize human error by using an Intelligent Document Agent for data extraction.

- Produce standardized and verified payoff letters that can be trusted for accuracy.

**1.2 High-Level Workflow**

At a high level, the application executes a series of steps to transform an NPGA document into a finalized payoff letter:

**1. Document Upload:** The user interacts with the `doc\_uploader.py` module to upload the NPGA document, which is then stored in the `/documents` folder.

**2. Schema Extraction:** The `schema\_extractor.py` script analyzes the uploaded NPGA document and extracts its schema. This schema is saved in the `document\_schema.txt` file.

**3. Data Extraction:** The application isolates the Signature page from the NPGA document and stores the extracted data in `extracted\_data.txt`.

**4. Metadata Extraction and Vector Store Creation:** The `LDA\_metadata\_extract.py` module takes over to extract metadata from the NPGA document and create a vector store for it. The resulting vector store is saved in the `/vector\_store` folder.

**5. Intelligent Document Agent:** The `doc\_agent.py` is invoked, and it begins the autonomous detail extraction process. The agent conducts a similarity search within the vector store to find matching details. If the search is successful, the results are updated in `extracted\_data.txt`; otherwise, the agent performs query transformation and repeats the search until all necessary details are extracted.

**6. Payoff Letter Generation:** Using the details stored in `extracted\_data.txt`, the `letter\_generator.py` module generates the payoff letter. It utilizes the `template.py` to format the letter correctly.

**7. Final Output:** The completed payoff letter is saved in the `payoff\_letter.txt` file and made available for download through the user interface.

The application's workflow is designed to be seamless and user-friendly, ensuring that users can generate payoff letters with minimal input and the highest level of accuracy.

In the following sections, we will delve deeper into the detailed application workflow, usage instructions, and concluding thoughts on the application's utility.

**2. Application Workflow**

A diagram of a diagram

Description automatically generated with medium confidence

**2.1 Detailed Workflow Description**

The following is a detailed description of the application's workflow, breaking down the steps involved in each stage of the payoff letter generation process.

**Document Upload (`doc\_uploader.py`)**

The `doc\_uploader.py` file is a Streamlit-based interface that allows users to upload NPGA documents. It saves the documents to the `/documents` directory and utilizes the `save\_uploaded\_file` function to manage file storage. It also includes a button to initiate the payoff letter generation process.

Upon clicking the "Generate" button, the script calls the `extract\_document\_schema` and `extract\_signature\_page\_data` functions from `schema\_extractor.py`. It then invokes the `process\_documents` function from `LDA\_metadata\_extract.py` to extract metadata and create a vector store.

**Schema Extraction (`schema\_extractor.py`)**

The `schema\_extractor.py` script contains functions that interface with an OpenAI-based client to process the uploaded NPGA document. It includes functionality to identify and extract the table of contents (`find\_table\_of\_contents\_range`) and signature page data (`find\_signature\_page\_range`). The extracted schema and signature data are then saved to their respective text files.

**Metadata Extraction and Vector Store Creation (`LDA\_metadata\_extract.py`)**

The `LDA\_metadata\_extract.py` script preprocesses the document text and uses Latent Dirichlet Allocation (LDA) to extract metadata in the form of keywords from the document. It also creates a vector store index using the `GPTVectorStoreIndex` class, which is then persisted within the `/vector\_store` folder.

**Intelligent Document Agent (`doc\_agent.py`)**

The `doc\_agent.py` script contains the Intelligent Document Agent logic. It uses the document schema and extracted data to generate queries for details extraction. The script performs similarity searches and query transformations by invoking Azure OpenAI's client. Extracted results are appended to the `extracted\_data.txt` file.

**Payoff Letter Generation (`letter\_generator.py`)**

The `letter\_generator.py` script generates the payoff letter sections using the `get\_section\_generator` function. It iterates over the sections defined in `template.py` and uses the previously extracted data to fill in the template placeholders. The generated sections are appended to form the complete payoff letter, which is then saved to a text file.

**3. Usage**

**3.1 How to Run the Application**

To run the Payoff Letter Generator with Document Agent, follow these steps:

**1. Start the Interface:** Navigate to the root directory of the application (`/Payoff\_Gen\_Doc\_Agent\_deployment`) in your terminal or command prompt.

**2. Activate Virtual Environment (optional):** If you are using a virtual environment, activate it with the appropriate command.

For `virtualenv`:

```bash

source /path/to/your/virtualenv/bin/activate  # On Unix or MacOS

.\path\to\your\virtualenv\Scripts\activate    # On Windows

```

For `conda` environments:

   ```bash

   conda activate your\_env\_name

   ```

**3. Run Streamlit:** Execute the Streamlit application by running the following command:

   ```bash

   streamlit run doc\_uploader.py

   ```

**4. Access the Interface:** Streamlit will start a local server, and you should see a message with the local URL, typically `http://localhost:8501`. Open this URL in your web browser to interact with the application.

**3.2 User Interaction with the System**

When you access the application through the web browser, you will be presented with the following user interface:

**1. Document Upload:** Use the provided upload field to select and upload your Note Purchase and Guarantee Agreement (NPGA) document.

**2. Upload Confirmation:** After selecting your document, click the 'Upload Documents' button to save the document to the server.

**3. Payoff Letter Generation:** Once the document is uploaded, click the 'Generate' button to start the payoff letter generation process. The application will display progress messages as it completes each step.

**4. Download Payoff Letter:** After the process is complete, the application will save the generated payoff letter to `payoff\_letter.txt`. A download link or button will be provided for you to download the final document.

**4. Conclusion**

The Payoff Letter Generator with Document Agent represents a significant step forward in automating the extraction and generation of financial documents. By combining advanced natural language processing techniques with a user-friendly interface, the application minimizes the manual effort and potential for error inherent in traditional methods.

**4.1 Final Thoughts**

The application successfully demonstrates the potential of AI-powered document agents in the realm of legal and financial document processing. Its ability to extract relevant details from complex documents and generate standardized payoff letters can save time, reduce errors, and improve overall efficiency for users.