**Adapt Payoff Letter Generator Documentation**

1. **Introduction**

**1.1 Overview of the Adapt Payoff Letter Generator**

The Adapt Payoff Letter Generator is an innovative application designed to automate the process of creating payoff letters, which are crucial documents in the financial and legal sectors. These letters are typically issued to inform the current status and the amount required to pay off a debt in full. The application streamlines the task of generating these letters by intelligently extracting pertinent details from a Note Purchase and Guarantee Agreement (NPGA) provided by the user.

**1.2 Purpose and Target Audience**

The primary purpose of this application is to reduce manual effort, minimize errors, and increase efficiency in generating accurate and verified payoff letters. The Adapt Payoff Letter Generator is particularly useful for financial institutions, law firms, and corporate legal departments that frequently deal with loan agreements and financial settlements.

The target audience for this application includes:

- Lawyers and legal professionals who need to generate payoff letters for their clients.

- Loan officers and financial analysts in banks and other lending institutions.

- Compliance officers who require accurate documentation for auditing and regulatory purposes.

- Any professional or entity involved in debt settlement and financial agreement processes.

**1.3 Core Features and Benefits**

The Adapt Payoff Letter Generator offers several key features that make it a valuable tool for professionals:

**1. Intelligent Research-Based Approach (ADAPT):** The application employs a sophisticated AI-powered system named ADAPT to automate the extraction of relevant details from legal documents. This system utilizes advanced techniques like natural language processing to ensure precision in the data extraction process.

**2. Schema Extraction:** Before extracting specific details, the application first determines the schema of the NPGA document. This understanding of the document's structure enables more accurate and context-aware information retrieval.

**3. Metadata Extraction and Indexing:** The application extracts essential metadata from the document, which is crucial for organizing and referencing the information within the payoff letter.

**4. Vector Store Creation:** By creating a vector store, the application represents the document's data in a format that facilitates easy access and manipulation during the payoff letter generation process.

**5. Autonomous Detail Extraction:** The ADAPT system autonomously plans and executes the detail extraction process, ensuring that all necessary information is collected without manual intervention.

**6. Template-Based Letter Generation:** The application utilizes a customizable template to produce the final payoff letter, ensuring consistency and adherence to necessary formatting standards.

**7. Execution Logging:** The application maintains a log of its operations, providing transparency and traceability for each generated payoff letter.

The benefits of using the Adapt Payoff Letter Generator include:

- Time savings due to automation of repetitive tasks.

- Increased accuracy with reduced risk of human error.

- Enhanced productivity, allowing professionals to focus on more strategic work.

- Easy integration into existing workflows with minimal training required.

The Adapt Payoff Letter Generator represents a significant step forward in the digitization and automation of legal and financial document processing, offering a reliable and efficient solution for professionals in these fields.

1. **Getting Started**

**2.1 Prerequisites**

Before you begin setting up the Adapt Payoff Letter Generator, ensure that you have the following prerequisites in place:

**2.1.1 Software Requirements**

- **Python:** The application is written in Python, so you will need Python 3.10 or later installed on your system.

- **Package Manager:** You'll need `pip` for installing Python packages. This is included by default in Python versions 3.4 and above.

- **Virtual Environment (optional):** It is recommended to use a virtual environment for the Python project to manage dependencies separately from your system-wide Python installation. You can use `venv` or `conda` for creating a virtual environment.

**2.1.2 System Requirements**

- **Operating System:** The application should be compatible with major operating systems like Windows, macOS, and Linux distributions that support Python.

**- Memory and Storage:** Sufficient RAM and storage space are required to process documents and store the generated payoff letters. At least 2 GB of RAM and 100 MB of free storage space are recommended.

- **Internet Connection:** Some components of the application may require an active internet connection to access APIs or download additional resources.

**2.2 Installation Instructions**

**2.2.1 Step-by-Step Guide to Setting Up the Application**

**1. Clone or Download the Application:** Obtain the application source code by cloning the repository or downloading it from the provided location.

**2. Set Up a Virtual Environment (optional):** To create a virtual environment, navigate to the project directory and run the following command:

```bash

python -m venv venv

```

Activate the virtual environment:

- On Windows:

     ```cmd

     venv\Scripts\activate

     ```

- On macOS and Linux:

     ```bash

     source venv/bin/activate

     ```

**3. Install Dependencies:** With the virtual environment activated, install the required dependencies by running the following command in the project root where the `requirements.txt` file is located:

   ```bash

   pip install -r requirements.txt

   ```

This will install all the necessary Python packages, including the specific versions listed in the `requirements.txt` file.

**2.2.2 Installing Dependencies from `requirements.txt`**

The `requirements.txt` file contains a list of all the Python packages needed for the application to run. These dependencies include libraries for natural language processing, machine learning, document processing, and more.

To install the dependencies, make sure you have `pip` installed and run the following command in your terminal or command prompt from the root directory of the project:

```bash

pip install -r requirements.txt

```

Please note that some packages may require additional installation steps, especially those that are not hosted on PyPI but rather on other URLs, such as the `en-core-web-sm` model for spaCy. The `pip` command should handle these automatically.

Once you have installed all the dependencies, the Adapt Payoff Letter Generator application is ready for use. You can now proceed to the next sections to learn about the application workflow, how to use the application, and how to get the most out of its features.

1. **Application Folder Structure**

Understanding the folder structure is crucial for navigating and maintaining the Adapt Payoff Letter Generator application. Here is a detailed explanation of each folder and file within the application:

- **/documents:** This directory is used to store the Note Purchase and Guarantee Agreement (NPGA) documents uploaded by the user. It serves as the initial input repository for the document processing workflow.

- **/metadata\_extract\_and\_index:** This module is tasked with extracting metadata from the NPGA documents. The metadata includes various information that is crucial for identifying and categorizing the content within the documents.

- `\_\_init\_\_.py`: An initialization file that allows Python to recognize the directory as a package.

- `LDA\_metadata\_extract.py`: The Python script responsible for the extraction of metadata using Latent Dirichlet Allocation (LDA) or similar methods.

- **/SchemaExtractor:** A module dedicated to extracting the schema of the NPGA document, which is essential for understanding the document's structure.

- `\_\_init\_\_.py`: Another initialization file for package recognition.

- `schema\_extractor.py`: Contains the code for determining the schema of the document.

- **/vector\_store:** A folder where the vector representations of the NPGA document are saved. These vectors are used for various operations such as similarity checks and indexing.

- **Adapt\_verified\_payoff\_letter.txt:** The output file where the generated payoff letter is saved once the process is complete.

- **app\_secrets.py:** A Python file that contains sensitive information like API keys and endpoints for the Language Learning Models (LLMs). It is important to secure this file and not expose its contents.

- **controller.py:** Part of the ADAPT automation system, this script acts as the controller that oversees the planner and executor components, orchestrating the extraction process.

- **doc\_uploader.py:** The user interface script where users upload their respective NPGA documents. It serves as the entry point for user interaction.

- **document\_schema.txt:** This file stores the extracted schema of the NPGA document, as determined by the `schema\_extractor.py` script.

- **execution\_log.txt:** A log file that records the execution details of the ADAPT automation system, providing a trace of the steps taken during the extraction process.

- **executor.py:** A component of the ADAPT automation system responsible for carrying out the execution of each step in the plan generated by the planner.

- **extracted\_data.txt:** This file is used to store the data extracted from the NPGA document using the ADAPT system.

- **letter\_generator.py:** A script that consumes the extracted data and generates the payoff letter using a predefined template.

- **planner.py:** Another part of the ADAPT automation system, responsible for creating an actionable plan based on the extracted schema and metadata.

- **prompt.py:** Contains prompts used to configure the LLM for specific tasks within the application.

- **template.py:** Manages the template used by the LLM to generate the payoff letter, ensuring that the output follows a consistent format.

- **tools\_description.py:** Provides descriptions of all the tools defined for use with the LLM, acting as a reference for the various components and their functions.

- **requirements.txt:** A file listing all the application dependencies, which can be installed using a package manager like `pip`.

1. **Application Workflow**

**A screenshot of a computer

Description automatically generated**

**4.1 Overview of the End-to-End Process**

The Adapt Payoff Letter Generator operates in a sequential workflow, starting from document upload to the final generation of the payoff letter. Here's a high-level overview:

**1.** **Upload NPGA Document:** The user interfaces with `doc\_uploader.py` to upload the document. This UI component ensures that the document is stored correctly in the `/documents` folder.

**2. Extract Document Schema:** `schema\_extractor.py` is executed to determine the structure of the document. The script uses algorithms to understand the organization of content and identifies key sections necessary for data extraction. The schema is essential for guiding subsequent steps in the extraction process.

**3. Signature Page and Data Extraction:** The application identifies and extracts the signature page and other relevant data from the NPGA document. This step is crucial as it collects the information that will populate the payoff letter.

**4. Metadata Extraction and Vector Store:** The application runs `LDA\_metadata\_extract.py` to extract metadata, which includes terms, dates, amounts, and other significant details. This metadata is indexed and stored in the vector format in the `/vector\_store`, facilitating efficient data retrieval.

**5. ADAPT Autonomous Details Extraction:** The control flow moves to `controller.py`, which calls on `planner.py` to create a detailed plan for extracting the remaining necessary data. The `executor.py` script then follows the plan, performing the extraction autonomously. The execution details and progress are logged in `execution\_log.txt`.

**6. Generate Payoff Letter:** The `letter\_generator.py` script takes the extracted data and uses `template.py` to fill in a payoff letter template with the appropriate information. The template ensures that the letter adheres to the required format and contains all necessary legal and financial details.

**7. Final Letter Output:** The completed payoff letter is saved to `Adapt\_verified\_payoff\_letter.txt`. The user can then access and download the finalized document from the application's user interface.

1. **Modules and Code Explanation**

**5.1 `doc\_uploader.py`**

- **Functionality and User Interaction Details**

`doc\_uploader.py` serves as the user interface of the Adapt Payoff Letter Generator application. It is built using Streamlit, which provides an easy-to-use web interface for uploading Note Purchase and Guarantee Agreement (NPGA) documents. The file uploader allows multiple files to be selected and uploaded. After the files are uploaded, the script saves them to the `/documents` directory and initiates the subsequent steps of the workflow, which include document schema extraction, metadata extraction, and payoff letter generation.

The script also includes functionality to clear previous files and execution logs to ensure that each run starts with a clean state. It handles the flow control for the entire application, calling various functions from other modules to execute the workflow steps.

**5.2 `schema\_extractor.py`**

- **Schema Extraction Process Explanation**

`schema\_extractor.py` is responsible for identifying the structure or schema of the uploaded NPGA documents. It uses natural language processing and the OpenAI Azure API to determine the presence of specific sections like the table of contents and signature pages. The script extracts text from these sections and uses prompts to guide the API to output structured information, which is then saved to `document\_schema.txt` and `extracted\_data.txt` for further processing.

The schema extraction helps the application understand the document's layout and content distribution, which is crucial for accurately extracting detailed information later in the workflow.

**5.3 `LDA\_metadata\_extract.py`**

- **Metadata Extraction and Vector Store Creation Details**

`LDA\_metadata\_extract.py` takes care of extracting metadata from the document and creating a vector store. The metadata includes keywords and summaries that represent the content of the document. This script preprocesses the document text, uses Latent Dirichlet Allocation (LDA) for topic modelling, and stores the results in a vectorized format.

The script also sets up the Azure OpenAI models and configurations required for document processing. The vector store is a crucial component that allows the application to perform efficient similarity searches during the detail extraction phase.

**5.4 `controller.py`, `planner.py`, and `executor.py`**

- **Description of the ADAPT Automation System Components**

These modules work together as part of the ADAPT automation system, which autonomously plans and executes the detail extraction process.

- `controller.py`: This script orchestrates the execution of the plan created by the planner. It logs the execution steps and results, handles the verification of responses, and attempts to replan and re-execute if necessary.

- `planner.py`: The planner creates an abstract plan based on user queries. It utilizes the OpenAI Azure API to generate a sequence of steps that the executor will follow to extract information from the document.

- `executor.py`: The executor carries out the plan by querying the document's vector store and retrieving information based on the plan's steps. It sets up the query engines and handles the interaction with the Azure OpenAI API to obtain and verify responses.

Together, these components form a robust system that can understand user queries, devise a strategy to find the necessary information, and execute that strategy to produce accurate results.

**5.5 `letter\_generator.py`**

- **Payoff Letter Generation Process**

`letter\_generator.py` is responsible for generating the payoff letter based on the extracted data. It uses a template approach, where placeholders in the template are replaced with actual data from `extracted\_data.txt`. The script employs the Azure OpenAI API to ensure that the generated sections of the letter are coherent and contextually correct.

The generator works by iterating over each section of the template, constructing prompts, and using the API to fill in the details. The final sections are then compiled into the complete payoff letter, which is saved to `Adapt\_verified\_payoff\_letter.txt`. The script includes functionality to measure execution time and provides a decorator to time the letter generation process.

These modules and their interactions demonstrate the complexity and intelligence of the Adapt Payoff Letter Generator's workflow, showcasing the application's capability to transform a manual, error-prone task into an automated and efficient process.

1. **Usage Guide**

This section provides a comprehensive guide on how to run the Adapt Payoff Letter Generator application, upload a document, generate a payoff letter, and download the generated letter.

**6.1 How to Run the Application**

To run the Adapt Payoff Letter Generator, you'll need to have Streamlit installed. If you haven't installed Streamlit yet, you can do so by running the following command:

```bash

pip install streamlit

```

Once Streamlit is installed, navigate to the root directory of the application and run the following command in your terminal:

```bash

streamlit run doc\_uploader.py

```

Executing this command will start the Streamlit server and automatically open the application in your default web browser. If the application does not open automatically, you can manually navigate to the local server address provided in the terminal output, which is typically `http://localhost:8501`.

**6.2 Uploading a Document and Generating a Payoff Letter**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**A black background with white text

Description automatically generated**

**1. Access the Application Interface:** Open the application in your web browser as described in the previous step.

**2. Select Payoff Date:** Use the date input field labeled "Payoff Date" to select the desired payoff date for the letter. The application will prefill this field with a default date, which you can change as needed.

**3. Upload Document:** Click on the "Upload documents" section to select the Note Purchase and Guarantee Agreement (NPGA) documents you want to process. You can upload multiple documents if necessary.

**4. Initiate Upload:** After selecting the documents, click the "Upload Documents" button to upload the files to the server. The application will save the documents to the `/documents` directory and display a success message once the upload is complete.

**5. Generate Payoff Letter:** Click the "Generate" button to start the payoff letter generation process. The application will execute the workflow steps, including extracting the document schema, metadata, and other necessary details, and then generate the payoff letter using the extracted data.

**6. Wait for Completion:** The process of generating a payoff letter can take some time. Wait until the application indicates that the generation is complete.

**6.3 Downloading the Generated Payoff Letter**

**A screenshot of a computer screen

Description automatically generated**

Once the payoff letter has been successfully generated, you will see a "Download Letter" button on the application interface.

**1. Download Letter:** Click the "Download Letter" button to download the generated payoff letter to your local machine.

**2. Review Letter:** The letter will be downloaded as a text file named `Adapt\_verified\_payoff\_letter.txt`. Open and review the letter to ensure that all the necessary details have been correctly incorporated.

**3. Cleanup:** After downloading the letter, the application will perform a cleanup operation to clear previous files and ensure that subsequent runs start with a clean state.

By following these steps, you can easily use the Adapt Payoff Letter Generator to automate the creation of payoff letters, saving time and reducing the potential for human error.

1. **API Reference**

**7.1 `app\_secrets.py`**

- **Description of API Keys and Endpoints**

The `app\_secrets.py` file stores the API keys and endpoints that are required to interact with external services, such as the OpenAI API for language model access. This file contains sensitive information that should not be exposed publicly. As a developer, you will use this file to configure the necessary credentials for the application to communicate with these services. The file includes keys for general API access (`api\_key`) and specific endpoints for different models and services (`azure\_endpoint`, `gpt\_4t\_azure\_endpoint`, etc.).

Note: Actual API keys and endpoint URLs are confidential and should be stored securely, preferably using environment variables or a secure key management system.

**7.2 `prompt.py`**

- **Configuration Prompts for the LLM**

The `prompt.py` file contains predefined prompts and messages that guide the language learning model (LLM) in performing specific tasks such as schema extraction, signature page identification, and plan generation. These prompts are crafted to instruct the LLM to analyze the structure of legal documents, identify key sections, and generate responses that are structured and useful for further processing within the application. Each prompt in this file is associated with a role, either 'system' or 'user', indicating the context in which the prompt is used.

**Planner Prompt:**



**Query Transformation Prompt:**

A black screen with white text

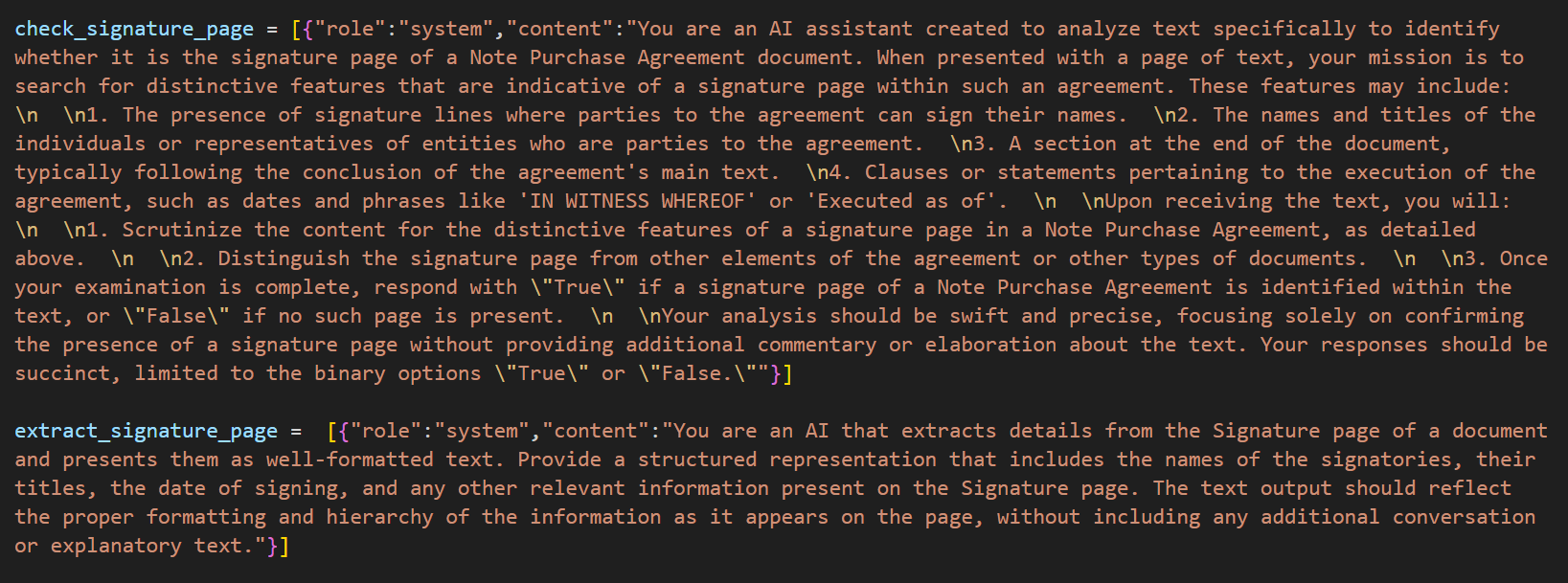
Description automatically generated

**Table of Contents Extraction Prompt:**

**A screenshot of a computer screen

Description automatically generated**

**Signature Page Extraction Prompt:**

****

**7.3 `template.py`**

- **Template Management for Letter Generation**

`template.py` defines the structure of the payoff letter using placeholders within a template dictionary named `new\_payoff\_template`. Each key in the dictionary represents a section of the payoff letter, and the associated value is a string template for that section. The template includes placeholders (e.g., `[Current date]`, `[Issuer Address Line 1 Placeholder]`) that the application replaces with actual extracted data during the letter generation process. This file is crucial for ensuring the payoff letter adheres to the required format and includes all necessary information.

**7.4 `tools\_description.py`**

- **Description of the LLM Tools**

The `tools\_description.py` file provides a description of various functions and tools used by the LLM within the application. It includes a list of function descriptions, each of which contains a name, description, parameters, and expected return values. The file serves as a reference for developers to understand the purpose and functionality of each tool within the ADAPT automation system. It explains how the tools contribute to tasks such as performing similarity searches, updating responses, and appending extracted data to local files.

By leveraging the information and configurations provided in these files, developers can ensure the Adapt Payoff Letter Generator interacts correctly with external services and internal components, maintains a consistent letter format, and efficiently executes its automation tasks.  
  
  
**8. Conclusion**

The Adapt Payoff Letter Generator represents a significant advancement in automating the generation of financial and legal documents. By leveraging intelligent research-based approaches and natural language processing, this application streamlines the complex process of extracting information from Note Purchase and Guarantee Agreement documents and creating accurate and professional payoff letters.

Throughout this documentation, we have explored the various components and workflows that make up the Adapt Payoff Letter Generator. From the initial document upload to the intricate extraction of metadata and the generation of the final letter, each step has been designed to optimize efficiency and reduce the potential for human error.

The modular design of the application, along with its use of advanced machine learning models and APIs, ensures that users can rely on the system to produce consistent and high-quality results. The application's folder structure and workflow are carefully planned to maintain a clean and organized codebase, making it easier for future developers to understand, maintain, and enhance the system.

Developers looking to contribute to the project can find a wealth of opportunities to add value, whether through code improvements, documentation enhancements, or feature development. The open-source nature of the project encourages collaboration and innovation, allowing the Adapt Payoff Letter Generator to continue evolving to meet the changing needs of its users.

As the financial and legal sectors continue to embrace digital transformation, tools like the Adapt Payoff Letter Generator play a critical role in empowering professionals to focus on high-value tasks by automating routine documentation processes. We hope that this documentation serves as a valuable guide for developers and users alike, and we look forward to seeing the community's contributions and the ongoing evolution of the Adapt Payoff Letter Generator.